

USER GUIDE

Essential Studio

for EJ2 TypeScript

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

Syncfusion JavaScript (ES5) UI Controls (Essential JS 2)	72
Components list	72
table	72
How to best read this user guide	74
Getting help	74
See also	74
Browser support	75
Required polyfills	75
Using CDN	75
Node.js	75
Getting started	76
Getting Started with Syncfusion JavaScript (ES5) UI control in a quickstart application	76
Prerequisites	76
Set up development environment	76
Add Syncfusion resources	76
Add Syncfusion control to the application.....	82
Run the application	83
See also	85
Compatibility with Syncfusion JavaScript (Essential JS 1)	85
Prerequisites	85
Creating JavaScript application with Syncfusion JavaScript (Essential JS 2) control.....	85
Adding Syncfusion JavaScript (Essential JS 1) control in the application.....	86
Installation and upgrade	88
Installation	89
Install by using npm CLI.....	89
Install by using package.json.....	89
Download JavaScript – EJ2 Installer	89
Download the Trial Version	89
Installation using Web Installer	92
Overview	92
Installation	93
Uninstallation.....	99
Installation using Offline Installer	104
Installing with UI	104
Installing in silent mode	112

Installing Syncfusion JavaScript – EJ2 Mac Installer.....	113
Steps to resolve the warning message in Catalina OS or later	113
Step-by-Step Installation.....	114
License key registration in samples	118
Linux Installer	118
Download Syncfusion JavaScript Linux Installer	118
Installing Syncfusion JavaScript Linux installer	122
Common Installation Errors	123
Unlocking the license installer using the trial key.....	123
License has expired	123
Unable to find a valid license or trial	124
Unable to install because of another installation.....	125
Unable to install due to controlled folder access	126
Upgrading Syncfusion JavaScript (Essential JS2).....	128
Upgrading to the latest version	128
Upgrade from trial version to license version.....	128
Licensing.....	128
Overview in EJ2 JavaScript Licensing control.....	128
Difference between unlock key and license key.....	129
Registering Syncfusion license keys in Build server	129
See Also	129
License key generation in EJ2 JavaScript Licensing control	129
Claim License key	130
See Also	131
License key registration in EJ2 JavaScript Licensing control	131
Javascript es5	131
See also	132
Licensing errors in EJ2 JavaScript Licensing control.....	132
Licensing errors.....	132
Licensing errors from version 16.2.0* to 20.3.0*	134
Licensing troubleshoot in EJ2 JavaScript Licensing control	138
Is an internet connection required for license validation.....	138
Upgrade from the trial version after purchasing a license	138
Where can I get a license key.....	138
Refer EJ2 scripts without registering the license key	138

Will the registered license key expire	140
When to generate new license key while upgrading.....	140
License registration for multiple developers on your project	140
Can I use the same key for all the web apps under the project	140
Does the license registration access any resources or data	140
License & Downloads shows the "Essential Studio Enterprise Edition Binary with Test Studio" and the "Project License". Which license to use.....	140
Appearance	141
Theme in EJ2 JavaScript controls	141
Reference themes in the application.....	142
NPM packages.....	142
CDN reference.....	143
Change theme dynamically	144
Common variables	146
Size modes Size Mode for Syncfusion EJ2 JavaScript Controls	162
Size mode for application	162
Size mode for a control	163
Change size mode for application at runtime.....	163
Change size mode for a control at runtime	165
See also	166
Icons Library	166
Referring icons in JavaScript application	166
Steps to use icons library	167
Available icons	170
Overview	171
Customizing theme color from theme studio.....	171
Import previously changed settings into the theme studio	180
Material 3 Theme.....	182
Syncfusion Material 3 Theme	182
What are CSS Variables?	183
Dark mode support	184
Common.....	187
Accessibility in Syncfusion EJ2 JavaScript controls	187
Accessibility overview	187
Accessibility standards	187

Accessibility compliance	188
Ensuring accessibility	189
Accessibility support for specific controls.....	189
table	189
Animation in EJ2 JavaScript	191
Animation effects.....	191
Animation duration.....	192
Animation delay	193
Enable or disable animation globally	194
Right-To-Left support in Syncfusion JavaScript Controls	194
Enable RTL for all controls.....	194
Enable RTL for an individual control	195
State Persistence in Syncfusion JavaScript controls	196
State Persistence supported controls and properties	198
Template Engine	200
Compiling	200
Available template syntax.....	201
Custom helper.....	201
Template in Syncfusion EJ2 JavaScript controls.....	202
Types of templates.....	202
String template	203
Script template.....	204
Function template.....	206
Getting started with Localization.....	210
Loading translations.....	210
Changing current locale	212
Internationalization.....	212
Loading culture data	212
Changing global culture and currency code.....	213
Manipulating numbers.....	213
Manipulating dateTime.....	218
Drag and drop in EJ2 JavaScript	224
Draggable	224
Droppable	227
See also	228

Troubleshoot.....	228
Content Security Policy	228
How To	229
Updating Syncfusion npm packages	229
How to load culture files in Essential JS 2 using Ajax.....	230
How to load the culture file in Essential JS 2	233
How to resolve Content Security Policy (CSP) errors.....	234
3D Chart	235
Working with data in EJ2 JavaScript 3D Chart control.....	235
Local data	235
Remote data.....	236
Binding data using ODataAdaptor	237
Empty points	238
Dimensions in EJ2 JavaScript 3D Chart control.....	241
Size for container	241
Size for chart	242
Category axis in EJ2 JavaScript 3D Chart control	245
Labels placement	246
Range	247
Indexed category axis.....	249
Numeric axis in EJ2 JavaScript 3D Chart control.....	250
Range	251
Range padding	252
Label format.....	259
Grouping separator	260
Custom label format	262
DateTime axis in EJ2 JavaScript 3D Chart control.....	263
DateTime axis.....	263
DateTime category axis.....	264
Label format.....	272
Custom label format	273
Logarithmic axis in EJ2 JavaScript 3D Chart control.....	275
Range	276
Logarithmic base	277
Logarithmic interval	279

Axis labels in EJ2 JavaScript 3D Chart control	280
Smart axis labels.....	280
Edge label placement.....	284
Maximum labels.....	285
Axis customization in EJ2 JavaScript 3D Chart control	287
Title	287
Title rotation	288
Tick lines customization	289
Grid lines customization	291
Multiple axis.....	292
Inversed axis.....	293
Opposed position	295
Multiple panes in EJ2 JavaScript 3D Chart control	296
Rows.....	296
Columns	300
Chart Types	303
Column Chart in EJ2 JavaScript 3D Chart control	303
Stacked column chart in EJ2 JavaScript 3D Chart control.....	309
100% Stacked column chart in EJ2 JavaScript 3D Chart control	315
Bar Chart in EJ2 JavaScript 3D Chart control.....	319
Stacked bar chart in EJ2 JavaScript 3D Chart control	326
100% Stacked bar chart in EJ2 JavaScript 3D Chart control.....	333
Data labels in EJ2 JavaScript 3D Chart control.....	337
Position	338
Template	340
Text mapping	341
Format.....	342
Margin.....	344
Customization	345
Customizing specific label	346
Legend in EJ2 JavaScript 3D Chart control.....	348
Position and alignment	348
Legend customization	354
Series selection through legend.....	362
Collapsing legend item.....	363

Legend title	365
Arrow page navigation	366
Legend item padding	368
Tooltip in EJ2 JavaScript 3D Chart control	369
Default tooltip	369
Fixed tooltip	371
Format the tooltip	372
Tooltip template	373
Customize the appearance of tooltip	375
Selection in EJ2 JavaScript 3D Chart control	376
Point	376
Series	378
Cluster	379
Selection type	380
Selection during initial loading	382
Selection through legend	383
Print and Export in EJ2 JavaScript 3D Chart control	385
Print	385
Export	386
Appearance in EJ2 JavaScript 3D Chart control	387
Custom color palette	387
Data point customization	389
Point level customization	390
Chart area customization	391
Animation	394
Chart rotation	395
Title	396
Accessibility in EJ2 JavaScript 3D Chart control	403
WAI-ARIA	403
Keyboard navigation	403
Accordion	403
Getting started in EJ2 JavaScript Accordion control	403
Component Initialization	403
Initialize the Accordion using JSON items collection	405
Initialize the Accordion using HTML elements	407

See Also	409
Expand mode in EJ2 JavaScript Accordion control	409
Single	409
Multiple	410
See Also	411
Accessibility in EJ2 JavaScript Accordion control	411
ARIA attributes	412
Keyboard interaction	413
Ensuring accessibility	413
See also	413
Style in EJ2 JavaScript Accordion control.....	414
Customizing Accordion	414
Customizing the list items	414
Customizing Accordion's header.....	414
Customizing Accordion's expand and collapse icons.....	414
Customizing the hover state of Accordion control	415
Customizing selected item of Accordion control	415
How To	415
Set the nested accordion in EJ2 JavaScript Accordion control	415
Load content through post in EJ2 JavaScript Accordion control	417
Set custom animation in EJ2 JavaScript Accordion control	419
To keep single pane open always in EJ2 JavaScript Accordion control.....	421
Create wizard using accordion in EJ2 JavaScript Accordion control	422
Load accordion with data source in EJ2 JavaScript Accordion control	428
Load accordion items dynamically in EJ2 JavaScript Accordion control	429
Add icon to accordion header in EJ2 JavaScript Accordion control	431
Add font awesome in EJ2 JavaScript Accordion control	437
Customize expand collapse actions in EJ2 JavaScript Accordion control	439
Integrate treeview inside the accordion in EJ2 JavaScript Accordion control	441
Ej1 api migration in EJ2 JavaScript Accordion control	443
Accessibility	443
AjaxSettings.....	443
Animation.....	444
Items	445
Common.....	447

AccumulationChart	448
Getting started in EJ2 JavaScript Accumulation chart control.....	448
Dependencies.....	448
Installation and Configuration	449
Add Accumulation Chart to the Project	450
Pie dough nut in EJ2 JavaScript Accumulation chart control.....	452
Pie Chart.....	452
Radius Customization.....	453
Pie Center.....	454
Various Radius Pie Chart	455
Doughnut Chart.....	456
Start and End angles	457
Color & Text Mapping	458
Customization	459
Hide pie or doughnut border	460
Multi-level pie chart.....	461
See Also	465
Pyramid in EJ2 JavaScript Accumulation chart control	465
Mode.....	466
Size	467
Gap Between the Segments.....	468
Explode.....	468
Customization	469
See Also	471
Funnel in EJ2 JavaScript Accumulation chart control	471
Size	472
Neck Size	472
Gap Between the Segments.....	473
Explode.....	474
Smart Data Label.....	475
Customization	477
See Also	478
Data label in EJ2 JavaScript Accumulation chart control	478
Positioning.....	479
Smart labels.....	480

Data Label Template	481
Connector Line	482
Text Mapping	484
Format.....	484
Customization	486
Text wrap	487
Show percentages in data labels of pie chart	488
Grouping in EJ2 JavaScript Accumulation chart control	490
Group Mode.....	491
Customization	493
Empty points in EJ2 JavaScript Accumulation chart control	494
Customization	495
Annotation in EJ2 JavaScript Accumulation chart control	496
Region	497
Co-ordinate Units.....	498
Alignment.....	499
Tooltip in EJ2 JavaScript Accumulation chart control	500
Header.....	501
Format.....	502
Tooltip format	503
Fixed tooltip	504
Customization	505
To customize individual tooltip.....	506
Legend in EJ2 JavaScript Accumulation chart control.....	507
Position and Alignment.....	508
Legend Reverse	509
Legend Shape	510
Legend Size.....	511
Legend Item Size	512
Paging for Legend.....	513
Legend Text Wrap	514
Legend Title.....	515
Arrow Page Navigation	516
Legend Item Padding	517
Center label in EJ2 JavaScript Accumulation chart control.....	518

Hover text	519
Customization	520
Title and sub title in EJ2 JavaScript Accumulation chart control	522
Title Customization	523
SubTitle	524
SubTitle Customization	525
Chart print in EJ2 JavaScript Accumulation chart control.....	527
Print.....	527
Export.....	528
Accessibility in EJ2 JavaScript Accumulation chart control.....	529
WAI-ARIA attributes.....	530
Keyboard interaction	530
Ensuring accessibility	530
See also	530
Ej1 api migration in EJ2 JavaScript Accumulation chart control.....	531
Accumulation Chart	531
Annotation	533
Series.....	535
DataLabel	539
Legend.....	541
Methods.....	544
Events.....	544
AppBar.....	548
Size and color in EJ2 JavaScript AppBar control.....	548
Size	548
Color.....	552
Accessibility in EJ2 JavaScript AppBar control	557
Keyboard interaction	558
Ensuring accessibility	558
See also	558
Position in EJ2 JavaScript AppBar control	558
Top AppBar	558
Bottom AppBar	560
Sticky AppBar	562
Design in EJ2 JavaScript AppBar control	563

Spacer.....	563
Separator.....	565
Media Query	567
Designing AppBar with Menu	568
Designing AppBar with Buttons	570
Designing AppBar with SideBar.....	572
Style and appearance in EJ2 JavaScript Appbar control	574
CssClass	575
HtmlAttributes	576
AutoComplete	577
Data binding in EJ2 JavaScript Auto complete control	577
Bind to local data	577
Bind to remote data	581
See Also	583
Templates in EJ2 JavaScript Auto complete control	583
Item template	583
Group template.....	584
Header template	585
Footer template	587
No records template	588
Action failure template	589
See Also	590
Grouping in EJ2 JavaScript Auto complete control.....	590
Customization	592
See Also	592
Filtering in EJ2 JavaScript Auto complete control.....	592
Change the filter type	592
Filter item count.....	593
Limit the minimum filter character.....	595
Case sensitive filtering	596
Diacritics Filtering.....	597
See Also	598
Virtualization in AutoComplete Component	599
Binding local data.....	599
Binding Remote data.....	600

Grouping with Virtualization.....	602
Localization in EJ2 JavaScript Auto complete control.....	603
Loading translations.....	603
See Also.....	605
Style in EJ2 JavaScript Auto complete control	605
Customizing the appearance of wrapper element	605
Customizing the dropdown icon's color	605
Customizing the focus color.....	606
Customizing the outline theme's focus color	606
Customizing the disabled component's text color	606
Customizing the float label element's focusing color	606
Customizing the color of the placeholder text	607
Customizing the text selection color.....	607
Customizing the background color of focus, hover, and active item's	607
Customizing the appearance of pop-up element	607
Adding mandatory asterisk to placeholder and float label.....	608
Accessibility in EJ2 JavaScript Auto complete control	609
WAI-ARIA attributes.....	610
Keyboard interaction	610
Ensuring accessibility	612
See also	612
How To	612
Autofill in EJ2 JavaScript Auto complete control	612
Icon support in EJ2 JavaScript Auto complete control	614
Custom search in EJ2 JavaScript Auto complete control	615
Filter in EJ2 JavaScript Auto complete control.....	616
Achieve virtual scrolling in EJ2 JavaScript Auto complete control.....	618
Ej1 api migration in EJ2 JavaScript Auto complete control.....	620
DataBinding.....	620
Filtering	621
Placeholder	621
Popup	621
CSS.....	623
Grouping	623
Localization	623

Template	623
Sorting	624
Accessibility	624
Selection.....	625
Miscellaneous	625
Common.....	625
Avatar	627
Types in EJ2 JavaScript Avatar control	627
Avatar size	627
Avatar types	628
How To	630
Avatar customization in EJ2 JavaScript Avatar control	630
Integrate avatar into listview in EJ2 JavaScript Avatar control.....	634
Integrate avatar into badge in EJ2 JavaScript Avatar control	635
Badge	639
Types in EJ2 JavaScript Badge control	639
Badge styles	639
Badge types.....	641
How To	650
Badge customization in EJ2 JavaScript Badge control	650
Integrate badge into listview in EJ2 JavaScript Badge control.....	653
Dynamic badge content in EJ2 JavaScript Badge control.....	655
Barcode	657
Getting started in EJ2 JavaScript Barcode control	657
Dependencies.....	657
Setup for local development.....	657
Configuring system JS	657
Adding CSS reference.....	658
Adding Barcode Generator control.....	659
Adding QR Generator control	660
Adding Datamatrix Generator control	661
BarcodeGenerator in EJ2 JavaScript Barcode control.....	662
Code39	662
Code39 Extended	663
Code 11	664

Codabar	665
Code 32	666
Code 93	668
Code 93 Extended	669
Code 128	669
Customizing the Barcode color	670
Customizing the Barcode dimension	671
Customizing the text	672
Qrcodegenerator in EJ2 JavaScript Barcode control.....	673
QR Code	673
Customizing the Barcode color	674
Customizing the Barcode dimension	675
Customizing the text	676
Datamatrixgenerator in EJ2 JavaScript Barcode control	677
Data Matrix	677
Customizing the Barcode color	678
Customizing the Barcode dimension	679
Customizing the text	680
Export in EJ2 JavaScript Barcode control	681
Export.....	681
Breadcrumb	682
Data binding in EJ2 JavaScript Breadcrumb control	682
Items based on current Url	683
Absolute Url	684
Customize text when generated items using Url.....	685
Icons in EJ2 JavaScript Breadcrumb control.....	686
Icon in Breadcrumb item	686
Icon Position.....	690
Icon Only	691
Show icon only for first item.....	693
Navigation in EJ2 JavaScript Breadcrumb control.....	694
URL	694
Enable navigation for last Breadcrumb item	697
Open URL in new page or tab	698
Templates in EJ2 JavaScript Breadcrumb control	700

Item Template.....	700
Separator Template	702
Customize Specific Item Template.....	703
Overflow in EJ2 JavaScript Breadcrumb control	704
Collapsed.....	706
Menu.....	708
Wrap.....	709
Scroll.....	711
Hidden.....	712
None.....	714
Accessibility in EJ2 JavaScript Breadcrumb control	714
WAI-ARIA attributes.....	715
Keyboard interaction	715
Ensuring accessibility	715
See also	716
Bullet Chart	716
Bullet chart dimensions in EJ2 JavaScript Bullet chart control	716
Size for Container.....	716
Size for Bullet Chart.....	717
Axis customization in EJ2 JavaScript Bullet chart control	719
MajorTickLines and MinorTickLines Customization.....	719
Tick Placement	720
Label Format	721
GroupingSeparator	722
Custom Label Format.....	723
Label Placement.....	724
Opposed Position	725
Category Label	726
Category Label Customization	727
Data binding in EJ2 JavaScript Bullet chart control.....	728
Ranges in EJ2 JavaScript Bullet chart control.....	729
Color Customization.....	730
Value bar in EJ2 JavaScript Bullet chart control.....	731
Types of actual bar	732
Actual bar customization	733

Comparative bar in EJ2 JavaScript Bullet chart control	735
Types of target bar	736
Target bar customization	737
Title in EJ2 JavaScript Bullet chart control	738
Title	738
Subtitle	739
Title and SubTitle Position	740
Title Customization	744
SubTitle Customization	745
Customization in EJ2 JavaScript Bullet chart control	746
Orientation	746
Right-to-left (RTL)	747
Animation	748
Theme	749
Data label in EJ2 JavaScript Bullet chart control	750
Data Label Customization	752
Tool tip in EJ2 JavaScript Bullet chart control	753
Default tooltip	753
Tooltip template	755
Customization of the appearance of tooltip	756
Accessibility in EJ2 JavaScript Bullet chart control	757
WAI-ARIA attributes	758
Keyboard interaction	759
Ensuring accessibility	759
See also	759
ButtonGroup	759
Getting started in EJ2 JavaScript Button group control	759
Dependencies	759
Set up development environment	759
Add Syncfusion JavaScript packages	760
Import the Syncfusion CSS styles	760
Add ButtonGroup to the project	760
Run the application	761
Orientation	762
See Also	763

Types and styles in EJ2 JavaScript Button group control	763
ButtonGroup types	763
ButtonGroup styles	765
See Also	766
Selection in EJ2 JavaScript Button group control	766
Selection.....	766
Nesting	769
See Also	772
Accessibility in EJ2 JavaScript Button group control.....	772
Keyboard interaction	773
Ensuring accessibility	774
See also	774
How To	774
Create buttongroup with icons in EJ2 JavaScript Button group control.....	774
Create buttongroup with rounded corner in EJ2 JavaScript Button group control.....	775
Disable in EJ2 JavaScript Button group control.....	777
Enable ripple in EJ2 JavaScript Button group control	778
Enable rtl in EJ2 JavaScript Button group control	779
Form submit in EJ2 JavaScript Button group control.....	781
Initialize buttongroup using util function in EJ2 JavaScript Button group control	782
Show buttongroup selected state on initial render in EJ2 JavaScript Button group control.....	784
Button	786
Types and styles in EJ2 JavaScript Button control	786
Button styles	786
Button types.....	788
Icons	794
Button size	797
See Also	799
Accessibility in EJ2 JavaScript Button component	799
WAI-ARIA attributes.....	800
Keyboard interaction	800
Ensuring accessibility	800
See also	800
How To	800
Add link to a button in EJ2 JavaScript Button control.....	800

Create a block button in EJ2 JavaScript Button control	801
Customize button appearance in EJ2 JavaScript Button control	803
Customize input and anchor elements in EJ2 JavaScript Button control	805
Repeat button in EJ2 JavaScript Button control	806
Right to left in EJ2 JavaScript Button control.....	809
Set the disabled state in EJ2 JavaScript Button control.....	811
Tooltip for button in EJ2 JavaScript Button control.....	813
Ej1 api migration in EJ2 JavaScript Button control	814
Properties.....	814
Methods.....	816
Events.....	816
Calendar	816
Date range in EJ2 JavaScript Calendar control.....	816
Multi select in EJ2 JavaScript Calendar control	818
See Also	819
Globalization in EJ2 JavaScript Calendar control	819
Right-to-left.....	823
Customization in EJ2 JavaScript Calendar control	825
Disable weekends	826
Day cell format.....	827
Highlight weekends.....	829
See Also	830
Calendar views in EJ2 JavaScript Calendar control	830
View restriction.....	831
Accessibility in EJ2 JavaScript Calendar control.....	832
WAI-ARIA attributes.....	833
Keyboard interaction	834
Ensuring accessibility	835
See also	836
Islamic calendar in EJ2 JavaScript Calendar control	836
Style appearance in EJ2 JavaScript Calendar control.....	837
Customizing the background color for the Calendar	838
Customizing the Calendar date elements on hovering.....	838
Customizing the border of date cell grid	838
Customizing the Calendar title.....	838

Customizing the previous and next icon	839
Customizing the footer button	839
Customizing the selected date cell grid	839
Customizing the content header in Calendar	839
How To	840
Set clear button in calendar in EJ2 JavaScript Calendar control	840
Show dates of other months in EJ2 JavaScript Calendar control	841
Select a sequence of dates in calendar in EJ2 JavaScript Calendar control	843
Skip a month in calendar in EJ2 JavaScript Calendar control	844
Render the calendar with week numbers in EJ2 JavaScript Calendar control	846
Change the first day of week in EJ2 JavaScript Calendar control	846
Customize the calendar day header in EJ2 JavaScript Calendar control	848
Card	849
Getting started in EJ2 JavaScript Card control	849
Component initialization	849
Adding a header and content	852
See Also	853
Header content in EJ2 JavaScript Card control	853
Header	853
Content	854
Card image in EJ2 JavaScript Card control	855
Images	855
Divider	857
See Also	858
Action buttons in EJ2 JavaScript Card control	858
Vertical	858
See Also	860
Horizontal in EJ2 JavaScript Card control	860
Stacked cards	860
Style in EJ2 JavaScript Card control	861
Customizing the card	861
Customizing the Header element	862
Customizing the card content	862
Divider used to separate the elements inside the card	862
Including image within card element	862

Including a title or caption for the image	863
To include heading image within the header	863
Customizing the Header main title	863
Customizing the Header subtitle.....	863
Including action buttons or anchor tags	864
To align card elements horizontally	864
To align elements vertically within the horizontal layout.....	864
How To	864
Customize the card image title position in EJ2 JavaScript Card control	864
Integrate other component inside the card in EJ2 JavaScript Card control	866
Carousel	867
Populating items in EJ2 JavaScript Carousel control.....	867
Populating items using carousel item	867
Populating items using data source	868
Selection.....	870
Partial visible slides	873
See Also	876
Navigators and indicators in EJ2 JavaScript Carousel control.....	876
Navigators	876
Indicators	881
Play button.....	891
Animations and transitions in EJ2 JavaScript Carousel control	894
Animations	894
Intervals between slides	897
Auto play slides	899
Pause on hover.....	900
Looping slides.....	902
Slide changing events.....	903
Disable touch swiping	905
Swipe Modes.....	906
Accessibility in EJ2 JavaScript Carousel control	908
ARIA attributes.....	909
Keyboard interaction	910
Ensuring accessibility	910
See also	910

Styles and appearance in EJ2 JavaScript Carousel control	910
CSS Structure in JavaScript Carousel Control.....	910
Customizing the indicators	911
Customizing the navigators.....	914
Customizing partial slides size	917
Chart.....	918
Working with data in EJ2 JavaScript Chart control	918
Local Data.....	918
Remote Data	919
Binding Data Using ODataAdaptor.....	920
Lazy loading.....	921
Empty points	923
Chart dimensions in EJ2 JavaScript Chart control.....	926
Size for Container.....	926
Size for Chart.....	927
Category axis in EJ2 JavaScript Chart control.....	930
Labels Placement	931
Range	932
Indexed category axis.....	933
Numeric axis in EJ2 JavaScript Chart control	934
Range	935
Range Padding.....	936
Label Format	941
GroupingSeparator	942
Custom Label Format.....	943
Date time axis in EJ2 JavaScript Chart control	944
DateTime Axis	944
DateTimeCategory Axis.....	945
Label Format	952
Custom Label Format.....	953
Logarithmic axis in EJ2 JavaScript Chart control	954
Range	955
Logarithmic Base.....	956
Logarithmic Interval	957
Axis labels in EJ2 JavaScript Chart control	959

Smart Axis Labels	959
Axis Labels Positioning	962
Multilevel Labels	963
Edge Label Placement	968
Sorting	969
Labels Customization	970
Customizing Specific Point	971
Line break support	972
Maximum Labels	974
Axis customization in EJ2 JavaScript Chart control.....	975
Axis Crossing	975
Title	976
Title Rotation.....	977
Tick Lines Customization	978
Grid Lines Customization	980
Multiple Axis	981
Inversed Axis	982
Opposed Position	983
Strip line in EJ2 JavaScript Chart control.....	985
Horizontal Strip lines.....	985
Vertical Striplines	986
Customize the strip line	987
Customize the stripline text.....	988
See Also	989
Multiple panes in EJ2 JavaScript Chart control.....	989
Rows.....	989
Columns	992
Chart Types	996
Line Chart in EJ2 JavaScript control	996
Step line Chart in EJ2 JavaScript control	999
Stack line Chart in EJ2 JavaScript control.....	1001
Stacked line Chart in EJ2 JavaScript control.....	1004
Spline Chart in EJ2 JavaScript control	1007
Area Chart in EJ2 JavaScript control.....	1010
Range area Chart in EJ2 JavaScript control	1014

Range step area Chart in EJ2 JavaScript control	1016
Spline range area Chart in EJ2 JavaScript control	1019
Stack area Chart in EJ2 JavaScript control	1021
Stacked area Chart in EJ2 JavaScript control	1024
Stacked step area Chart in EJ2 JavaScript control.....	1027
Step area Chart in EJ2 JavaScript control.....	1029
Spline area Chart in EJ2 JavaScript control	1031
Column Chart in EJ2 JavaScript control.....	1034
Range column Chart in EJ2 JavaScript control	1039
Stack column Chart in EJ2 JavaScript control.....	1042
Stacked column Chart in EJ2 JavaScript control.....	1047
Bar Chart in EJ2 JavaScript control.....	1050
Stack bar Chart in EJ2 JavaScript control	1056
Stacked bar Chart in EJ2 JavaScript control	1060
Scatter Chart in EJ2 JavaScript control.....	1063
Bubble Chart in EJ2 JavaScript control.....	1067
Polar Chart in EJ2 JavaScript control.....	1070
Radar Chart in EJ2 JavaScript control.....	1080
High low Chart in EJ2 JavaScript control	1083
High low open close Chart in EJ2 JavaScript control.....	1086
Candle Chart in EJ2 JavaScript control	1088
Box whisker Chart in EJ2 JavaScript control.....	1091
Waterfall Chart in EJ2 JavaScript control	1095
Histogram Chart in EJ2 JavaScript control	1097
Error bar Chart in EJ2 JavaScript control.....	1099
Vertical Chart in EJ2 JavaScript control.....	1105
Pare to Chart in EJ2 JavaScript control	1106
Polar radar in EJ2 JavaScript Chart control	1110
Polar Chart	1110
Radar Chart	1118
See Also	1121
Chart series in EJ2 JavaScript Chart control	1121
Multiple Series	1121
Enable Complex Property in Series	1124
Technical indicators in EJ2 JavaScript Chart control	1125

Accumulation Distribution	1126
Average True Range (ATR)	1129
Bollinger Band	1133
Exponential Moving Average (EMA)	1140
Momentum	1143
Moving Average Convergence Divergence (MACD)	1150
Relative Strength Index (RSI).....	1158
Simple Moving Average (SMA)	1161
Stochastic	1165
Triangular Moving Average (TMA)	1172
Trend lines in EJ2 JavaScript Chart control	1182
Linear.....	1182
Exponential	1184
Logarithmic	1185
Polynomial	1187
Power	1188
Moving Average	1189
Forecasting.....	1192
Show or hide a trendline.....	1195
Data markers in EJ2 JavaScript Chart control	1196
Marker.....	1196
Shape.....	1197
Images	1198
Customization	1199
Customizing specific point	1200
Fill marker with series color	1201
See also	1202
Data labels in EJ2 JavaScript Chart control	1202
Position	1203
Data Label Template	1205
Text Mapping	1206
Format.....	1207
Margin.....	1209
Customization	1210
Customizing Specific Point	1211

Show percentage based on each series points	1213
See Also	1215
Chart annotations in EJ2 JavaScript Chart control	1215
Region	1216
Co-ordinate Units	1217
Alignment	1219
Adding y-axis sub title through on annotation	1220
See Also	1221
Legend in EJ2 JavaScript Chart control	1221
Position and Alignment	1221
Customization	1227
Series selection on Legend	1236
Collapsing Legend Item	1237
Legend Title	1238
Arrow Page Navigation	1240
Legend Item Padding	1242
See Also	1244
Tooltip in EJ2 JavaScript Chart control	1244
Default tooltip	1244
Fixed tooltip	1245
Format the tooltip	1246
Tooltip template	1248
Customize the appearance of tooltip	1249
See also	1250
Zooming in EJ2 JavaScript Chart control	1250
Enable zooming	1250
Modes	1252
Toolbar	1253
Enable pan	1255
Enable scrollbar	1256
Auto interval on zooming	1258
Data editing in EJ2 JavaScript Chart control	1259
Enable Data Editing	1259
Cross hair and track ball in EJ2 JavaScript Chart control	1261
Tooltip for axis	1263

Customization	1264
Trackball	1265
Synchronized Charts in EJ2 JavaScript Chart control	1267
Tooltip synchronization.....	1267
Crosshair synchronization	1270
Zooming synchronization.....	1273
Selection synchronization	1276
Selection in EJ2 JavaScript Chart control	1280
Point	1280
Series.....	1281
Cluster	1282
Rectangular selection.....	1284
Selection type.....	1285
Selection on load.....	1286
Selection through on legend.....	1288
Customization for selection	1289
See Also	1290
Chart print in EJ2 JavaScript Chart control	1291
Print.....	1291
Export.....	1292
Multiple chart export.....	1298
Exporting chart using base64 string.....	1300
Chart appearance in EJ2 JavaScript Chart control	1302
Custom color palette.....	1302
Data point customization.....	1303
Point level customization.....	1306
Chart area customization.....	1308
Animation.....	1312
Chart title	1315
See also	1323
Render methods in EJ2 JavaScript Chart control	1323
SVG	1324
Canvas	1324
Accessibility in EJ2 JavaScript Chart control	1324
WAI-ARIA attributes.....	1325

Keyboard interaction	1325
Ensuring accessibility	1326
See also	1326
Internationalization in EJ2 JavaScript Chart control	1326
Localization in EJ2 JavaScript Chart control	1328
Ej1 api migration in EJ2 JavaScript Chart control.....	1330
Annotations.....	1330
CommonSeriesOptions	1332
Crosshair	1332
3D chart.....	1444
Canvas rendering	1445
Indicators	1445
Legend.....	1448
primaryXAxis	1451
primaryYAxis	1630
Axes	1639
Rows.....	1752
Series.....	1752
marker.....	1759
TrendLines.....	1763
StripLines.....	1765
Multilevel Labels	1766
Methods.....	1768
Events.....	1768
How To	1775
Live chart in EJ2 JavaScript Chart control	1775
Prevent data label in EJ2 JavaScript Chart control.....	1777
Tool tip format in EJ2 JavaScript Chart control.....	1779
Add series in EJ2 JavaScript Chart control	1781
Points customization in EJ2 JavaScript Chart control	1783
Stacking total in EJ2 JavaScript Chart control	1785
Selected data grid in EJ2 JavaScript Chart control	1787
Marker customization in EJ2 JavaScript Chart control.....	1789
Legend customization in EJ2 JavaScript Chart control.....	1791
Tool tip table in EJ2 JavaScript Chart control.....	1793

Footer in EJ2 JavaScript Chart control	1795
Threshold in EJ2 JavaScript Chart control	1797
Grid data chart in EJ2 JavaScript Chart control.....	1799
Percentage tool tip in EJ2 JavaScript Chart control	1801
Data label template in EJ2 JavaScript Chart control	1803
Hide tool tip in EJ2 JavaScript Chart control	1804
Dotted line in EJ2 JavaScript Chart control	1807
Grid data pie in EJ2 JavaScript Chart control	1808
Down sampling in EJ2 JavaScript Chart control	1810
Clicked data in EJ2 JavaScript Chart control	1813
Initial scrollbar in EJ2 JavaScript Chart control	1815
Legend in table in EJ2 JavaScript Chart control	1817
Syn pan in EJ2 JavaScript Chart control	1821
Axis hide in EJ2 JavaScript Chart control.....	1824
Exact date in EJ2 JavaScript Chart control	1827
Dialog chart in EJ2 JavaScript Chart control.....	1830
Series visible in EJ2 JavaScript Chart control	1833
Dynamic chart in EJ2 JavaScript Chart control.....	1834
CheckBox.....	1836
Label and size in EJ2 JavaScript Check box control.....	1836
Label.....	1836
Size	1837
See Also	1839
Accessibility in EJ2 JavaScript Check box control.....	1839
WAI-ARIA attributes.....	1840
Keyboard interaction	1840
Ensuring accessibility	1840
See also	1840
How To	1840
Customized checkbox in EJ2 JavaScript Check box control	1840
Name and value in form submit in EJ2 JavaScript Check box control	1848
Right to left in EJ2 JavaScript Check box control	1849
Ej1 api migration in EJ2 JavaScript Check box control	1850
Properties.....	1851
Methods.....	1852

Events.....	1852
Chips.....	1853
Getting started in EJ2 JavaScript Chips control.....	1853
control Initialization	1853
Types in EJ2 JavaScript Chips control.....	1856
Input Chip.....	1856
Choice Chip	1857
Filter Chip	1858
Action Chip	1859
Customization in EJ2 JavaScript Chips control	1862
Styles	1862
Leading Icon	1864
Avatar.....	1865
Avatar Content.....	1867
Trailing Icon.....	1868
Outline Chip	1870
Style in EJ2 JavaScript Chips control	1871
Customizing the chip text	1871
Customizing the chip icon	1872
Customizing the chip delete button.....	1872
Customizing the chip outline	1872
Customizing the chip on selection	1872
Customizing the chip avatar text	1873
Accessibility in EJ2 JavaScript Chips component	1873
WAI-ARIA attributes.....	1874
Keyboard interaction	1874
Ensuring accessibility	1875
See also	1875
Circular Gauge.....	1875
Gauge dimensions in EJ2 JavaScript Circular gauge control	1875
Size for Container.....	1875
Size for Circular Gauge	1876
Gauge axes in EJ2 JavaScript Circular gauge control	1877
Axis Customization.....	1877
Angles and Direction	1878

Axis Radius	1880
Ticks.....	1882
Labels	1884
Minimum and Maximum	1892
Multiple Axes	1893
Gauge ranges in EJ2 JavaScript Circular gauge control.....	1894
Start and End.....	1894
Customization	1895
Radius.....	1896
Dragging Range	1898
Multiple Ranges	1899
Rounded corner radius	1900
Gradient Color.....	1901
See also	1907
Gauge pointers in EJ2 JavaScript Circular gauge control	1907
Needle Pointers.....	1908
RangeBar Pointer	1911
Rounded corner for range bar pointer	1913
Marker Pointer.....	1914
Dragging pointer	1917
Multiple Pointers	1917
Animation.....	1919
Gradient Color.....	1920
Gauge annotations in EJ2 JavaScript Circular gauge control	1924
Content	1924
Position	1925
Sub Gauge	1926
See also	1929
Animation in EJ2 JavaScript Circular Gauge control	1929
Gauge legend in EJ2 JavaScript Circular gauge control.....	1932
Legend customization	1932
Toggle option in legend	1935
Paging support in legend	1936
Legend text customization.....	1937
Gauge user interaction in EJ2 JavaScript Circular gauge control	1939

Tooltip for pointers	1939
Tooltip for ranges	1942
Tooltip for annotation	1942
Pointer Drag	1944
Gauge print and export in EJ2 JavaScript Circular gauge control	1945
Print	1945
Export	1946
Gauge appearance in EJ2 JavaScript Circular gauge control	1949
Gauge Title	1949
Gauge Position	1950
Area Customization	1952
Radius calculation based on angles	1954
Accessibility in EJ2 JavaScript Circular gauge control	1955
WAI-ARIA attributes	1955
Screen reading in Circular Gauge	1956
Ensuring accessibility	1956
See also	1956
Internationalization in EJ2 JavaScript Circular gauge control	1956
Globalization	1956
Right-to-left	1957
Ej1 api migration in EJ2 JavaScript Circular gauge control	1959
Circular gauge dimensions	1960
Axis Line	1960
Ticks	1961
Labels	1962
Ranges	1963
Needle Pointer	1965
Marker Pointer	1966
Rangebar Pointer	1966
Annotations	1967
Appearance	1967
Events	1968
How To	1969
Gauge range in EJ2 JavaScript Circular gauge control	1969
Color Picker	1972

Mode and value in EJ2 JavaScript Color picker control	1972
Rendering palette at initial load	1972
Color value	1973
See Also	1975
Localization in EJ2 JavaScript Color picker control	1975
Localization	1975
Right to Left - RTL	1977
See Also	1978
Accessibility in EJ2 JavaScript Color picker control	1979
WAI-ARIA attributes	1980
Keyboard interaction	1980
Ensuring accessibility	1980
See also	1980
Style and appearance in EJ2 JavaScript Color picker control	1981
How To	1981
Hide control buttons in EJ2 JavaScript Color picker control	1981
Render palette alone in EJ2 JavaScript Color picker control	1982
Colorpicker in dropdownbutton in EJ2 JavaScript Color picker control	1984
Customize colorpicker in EJ2 JavaScript Color picker control	1986
Handle no color support in EJ2 JavaScript Color picker control	2000
Disabled in EJ2 JavaScript Color picker control	2006
Ej1 api migration in EJ2 JavaScript Color picker control	2007
Properties	2007
Methods	2009
Events	2011
ComboBox	2011
Tags in EJ2 JavaScript Combo box control	2011
Select element	2012
UL element	2013
Input element	2014
Data binding in EJ2 JavaScript Combo box control	2014
Binding local data	2014
Binding remote data	2018
See Also	2019
Templates in EJ2 JavaScript Combo box control	2019

Item template	2019
Group template.....	2021
Header template	2022
Footer template	2023
No records template	2024
Action failure template	2026
See Also	2027
Grouping in EJ2 JavaScript Combo box control	2027
HTML select.....	2028
Customization	2030
See Also	2030
Filtering in EJ2 JavaScript Combo box control	2030
Limit the minimum filter character.....	2031
Change the filter type	2033
Case sensitive filtering	2034
Diacritics Filtering.....	2036
See Also	2037
Virtualization in ComboBox Component	2037
Binding local data.....	2038
Binding Remote data.....	2039
Grouping with Virtualization.....	2040
Filtering with Virtualization.....	2042
Localization in EJ2 JavaScript Combo box control	2044
Loading translations.....	2044
See Also	2045
Style in EJ2 JavaScript Combo box control.....	2045
Customizing the appearance of wrapper element	2045
Customizing the dropdown icon's color	2046
Customizing the focus color	2046
Customizing the outline theme's focus color	2046
Customizing the disabled component's text color	2046
Customizing the float label element's focusing color	2047
Customizing the color of the placeholder text	2047
Customizing the text selection color.....	2047
Customizing the background color of focus, hover, and active item's	2047

Customizing the appearance of pop-up element	2048
Adding mandatory asterisk to placeholder and float label.....	2048
Accessibility in EJ2 JavaScript Combo box control.....	2049
WAI-ARIA attributes.....	2050
Keyboard interaction	2051
Ensuring accessibility	2053
See also	2053
How To	2053
Autofill in EJ2 JavaScript Combo box control.....	2053
Cascading in EJ2 JavaScript Combo box control	2054
Icons support in EJ2 JavaScript Combo box control.....	2056
Achieve virtual scrolling in EJ2 JavaScript Combo box control	2057
Ej1 api migration in EJ2 JavaScript Combo box control	2059
DataBinding.....	2059
Filtering	2060
Template	2060
Applying CSS.....	2061
Grouping	2062
Accessibility.....	2062
Placeholder	2062
Miscellaneous	2062
Sorting.....	2063
Selection.....	2063
Popup.....	2064
Common.....	2064
ContextMenu	2065
Icons and navigation in EJ2 JavaScript Context menu control.....	2065
Icons	2065
Navigation	2067
See Also	2069
Template and multilevel nesting in EJ2 JavaScript Context menu control	2069
Template	2069
Multilevel nesting	2070
See Also	2071
Accessibility in EJ2 JavaScript Context menu control	2071

WAI-ARIA attributes.....	2072
Keyboard interaction	2073
Ensuring accessibility	2073
See also	2073
Style and appearance in EJ2 JavaScript Context menu control	2073
How To	2074
Populate menu items with data source in EJ2 JavaScript Context menu control	2074
Open and close contextmenu in EJ2 JavaScript Context menu control	2075
Change menu items dynamically in EJ2 JavaScript Context menu control.....	2077
Template in EJ2 JavaScript Context menu control.....	2079
Underline a character in the item text in EJ2 JavaScript Context menu control	2084
Open a dialog on contextmenu item click in EJ2 JavaScript Context menu control.....	2086
Change animation settings in EJ2 JavaScript Context menu control	2088
Add or remove context menu items in EJ2 JavaScript Context menu control	2089
Enable or disable context menu items in EJ2 JavaScript Context menu control	2090
DashboardLayout	2091
Setting size of cells in EJ2 JavaScript Dashboard layout control.....	2091
Modifying cell size.....	2092
Setting cell spacing.....	2093
Graphical representation of layout.....	2094
Rendering component in right-to-left direction	2095
Panels	2097
Position sizing of panels in EJ2 JavaScript Dashboard layout control.....	2097
Setting header of panels in EJ2 JavaScript Dashboard layout control.....	2100
Add remove panels in EJ2 JavaScript Dashboard layout control	2104
Interaction With Panels	2108
Dragging moving of panels in EJ2 JavaScript Dashboard layout control	2108
Moving panels in EJ2 JavaScript Dashboard layout control.....	2114
Resizing of panels in EJ2 JavaScript Dashboard layout control.....	2115
Floating of panels in EJ2 JavaScript Dashboard layout control.....	2119
Responsive adaptive in EJ2 JavaScript Dashboard layout control	2121
Save restore in EJ2 JavaScript Dashboard layout control	2122
Style in EJ2 JavaScript Dashboard layout control	2124
Customizing the dashboard layout panel header	2124
Customizing the dashboard layout panel content.....	2124

Customizing the dashboard layout panel resize icon	2125
Customizing the dashboard layout panel background	2125
How To	2125
Resize the panel dynamically in EJ2 JavaScript Dashboard layout control	2125
Accessibility in EJ2 JavaScript Dashboard Layout component.....	2127
WAI-ARIA attributes.....	2128
Keyboard interaction	2128
Ensuring accessibility	2128
See also	2128
DataManager	2129
Getting started in EJ2 JavaScript Data control.....	2129
Dependencies.....	2129
Setup for local environment	2129
Adding Syncfusion resources	2129
Connection to a data source.....	2131
Filter	2137
Sort.....	2141
Page.....	2145
Component binding	2149
Data binding in EJ2 JavaScript Data control.....	2154
Local data binding	2154
Remote data binding.....	2156
See Also	2157
Adaptors in EJ2 JavaScript Data control.....	2157
Json adaptor	2158
Url adaptor	2159
OData adaptor	2160
ODataV4 adaptor	2161
Web API adaptor	2163
WebMethod Adaptor.....	2163
Custom Data Adaptor	2164
GraphQL Adaptor	2167
Writing custom adaptor.....	2171
Querying in EJ2 JavaScript Data control	2173
Specifying resource name using `from`	2173

Projection using `select`	2175
Eager loading navigation properties	2176
Sorting	2178
Filtering	2179
Searching.....	2183
Grouping	2184
Paging.....	2186
Aggregation.....	2187
Hierarchical query	2189
Manipulation in EJ2 JavaScript Data control.....	2191
Insert	2191
Update.....	2193
Remove	2196
Batch Edit Operation.....	2198
State persistence in EJ2 JavaScript Data control	2201
Preventing a query from persistence.....	2201
How to get or set the existing persisted data.....	2202
Restoring the initial state of Datamanager.....	2203
Use case example demonstrating state persistence with the DataManager	2204
How to in EJ2 JavaScript Data control.....	2205
Work in offline mode	2205
Sending additional parameters to server	2206
Adding custom headers	2208
DatePicker	2208
Date range in EJ2 JavaScript Datepicker control.....	2208
Date format in EJ2 JavaScript Datepicker control.....	2210
Date masking in EJ2 JavaScript Datepicker control	2211
Globalization in EJ2 JavaScript Datepicker control	2215
Right-To-Left	2219
Strict mode in EJ2 JavaScript Datepicker control.....	2222
Customization in EJ2 JavaScript Datepicker control	2224
Adding mandatory asterisk to placeholder and float label.....	2226
See Also	2227
Date views in EJ2 JavaScript Datepicker control.....	2227
Start view	2228

Depth view	2229
Accessibility in EJ2 JavaScript DatePicker control.....	2230
WAI-ARIA attributes.....	2231
Keyboard Interaction	2231
Ensuring accessibility	2233
See also	2233
Style appearance in EJ2 JavaScript DatePicker control.....	2233
Customizing the appearance of DatePicker wrapper element.....	2233
Customizing the DatePicker icon element.....	2234
Customizing the Calendar popup of the DatePicker.....	2234
Full screen mode support in mobiles and tablets.....	2234
How To	2236
Disabled the datepicker component in EJ2 JavaScript DatePicker control	2236
Set the placeholder in EJ2 JavaScript DatePicker control.....	2237
Set the readonly in EJ2 JavaScript DatePicker control.....	2238
Prevent the popup close in EJ2 JavaScript DatePicker control.....	2239
Customize the datepicker day header in EJ2 JavaScript DatePicker control	2240
Open datepicker popup on input click in EJ2 JavaScript DatePicker control.....	2241
Client side validation in EJ2 JavaScript DatePicker control.....	2242
Ej1 api migration in EJ2 JavaScript DatePicker control	2244
Date Selection	2244
Date Format	2244
Calendar Views.....	2245
Date Range	2245
Disabled Dates	2245
Customization	2246
Accessibility.....	2249
Persistence	2249
Validation	2250
Common.....	2250
Globalization	2252
Strict Mode	2252
Open and Close	2252
View Navigation	2253
DateRangePicker	2254

Range restriction in EJ2 JavaScript Daterangepicker control	2254
Restrict the range within a range.....	2254
Range span.....	2255
Strict mode.....	2256
Globalization in EJ2 JavaScript Daterangepicker control	2257
Right-To-Left	2262
Customize the date format	2263
Customization in EJ2 JavaScript Daterangepicker control.....	2264
Day cell format.....	2264
Preset Ranges.....	2265
First day of week	2267
See Also	2268
Accessibility in EJ2 JavaScript Daterangepicker control	2268
WAI-ARIA attributes.....	2269
Keyboard Interaction	2269
Ensuring accessibility	2271
See also	2271
Style appearance in EJ2 JavaScript Daterangepicker control	2271
Customizing the appearance of DateRangePicker wrapper element.....	2272
Customizing the DateRangePicker icon element.....	2272
Customizing the DateRangePicker popup calendar header	2272
Customizing the DateRangePicker popup calendar header title	2272
Customizing the DateRangePicker popup calendar content	2273
Customizing the DateRangePicker popup calendar content title.....	2273
Customizing the DateRangePicker popup calendar previous and next icon	2273
Customizing the DateRangePicker popup calendar date cell grid on hovering.....	2273
Customizing the DateRangePicker popup calendar primary button in footer	2274
Customizing the DateRangePicker popup calendar cancel button in footer.....	2274
Customizing the footer element in the DateRangePicker popup calendar	2274
Customizing the selected date cell grid in the DateRangePicker popup calendar	2275
Full screen mode support in mobiles and tablets.....	2275
How To	2277
Disable the daterangepicker component in EJ2 JavaScript Daterangepicker control	2277
Set the placeholder in EJ2 JavaScript Daterangepicker control.....	2278
Customization using cssclass in EJ2 JavaScript Daterangepicker control	2279

Customize the daterangepicker day header in EJ2 JavaScript Daterangepicker control.....	2280
Ej1 api migration in EJ2 JavaScript Daterangepicker control.....	2282
Date Selection	2282
Date Format	2282
Date Range.....	2283
Disabled Dates	2284
Customization	2285
Accessibility.....	2286
Persistence	2287
Common.....	2287
Globalization	2289
Strict Mode	2289
Open and Close	2290
DateTimePicker	2290
Globalization in EJ2 JavaScript Datetimepicker control.....	2290
Right-To-Left	2295
Strict mode in EJ2 JavaScript Datetimepicker control	2296
Date time range in EJ2 JavaScript Datetimepicker control.....	2299
Date time format in EJ2 JavaScript Datetimepicker control	2300
Date time masking in EJ2 JavaScript Datetimepicker control.....	2301
Configure Mask Placeholder	2304
Customization in EJ2 JavaScript Datetimepicker control.....	2305
Day and Time Cell format.....	2305
Adding mandatory asterisk to placeholder and float label.....	2307
See Also	2308
Accessibility in EJ2 JavaScript Datetimepicker control	2308
WAI-ARIA attributes.....	2309
Keyboard Interaction	2309
Ensuring accessibility	2312
See also	2312
Style appearance in EJ2 JavaScript Datetimepicker control	2312
Customizing the appearance of DateTimePicker wrapper element.....	2312
Customizing the DateTimePicker icons element	2312
Customizing the time picker popup in the DateTimePicker	2312
Customizing the Calendar popup of the DateTimePicker	2313

Full screen mode support in mobiles and tablets.....	2313
How To	2315
Disable the datetimepicker component in EJ2 JavaScript Datetimepicker control	2315
Set the placeholder in EJ2 JavaScript Datetimepicker control	2316
Customize the datetimepicker day header in EJ2 JavaScript Datetimepicker control	2317
Ej1 api migration in EJ2 JavaScript Datetimepicker control.....	2318
DateTime Selection	2318
DateTime Format	2319
Calendar Views.....	2319
Date Range	2319
Disabled Dates	2320
Customization	2320
Accessibility	2322
Persistence	2323
Validation	2323
Common.....	2324
Globalization	2326
Strict Mode	2326
Open and Close	2327
View Navigation	2327
Diagram.....	2328
Getting started in EJ2 JavaScript Diagram control.....	2328
Dependencies.....	2328
Installation and Configuration	2329
Add diagram to the project.....	2330
Module Injection.....	2331
Flow Diagram	2332
Automatic organization chart	2337
Nodes in EJ2 JavaScript Diagram control	2340
Create node.....	2341
Add node through nodes collection.....	2341
Add/Remove node at runtime	2342
Add node from palette.....	2343
Create node through data source.....	2344
Draw nodes	2344

Position	2344
Flip.....	2345
Appearance	2347
Gradient	2348
Linear gradient.....	2348
Radial gradient	2350
Shadow.....	2352
Customizing shadow	2353
Icon.....	2354
Customizing expand icon	2356
Customizing collapse icon	2356
Interaction.....	2356
Constraints	2356
Custom properties	2356
Stack order	2357
Data flow	2357
See Also	2358
Shapes in EJ2 JavaScript Diagram control.....	2359
Text	2359
Image.....	2360
Image alignment	2364
HTML.....	2367
HTML Node With Template	2368
Native.....	2370
SVG content alignment	2374
Basic shapes	2375
Path	2377
Flow Shapes	2378
Bpmn shapes in EJ2 JavaScript Diagram control.....	2380
Event	2382
Gateway	2385
Activity	2388
Tasks.....	2389
Subprocess	2392
Event subprocess	2394

Transaction subprocess.....	2396
Process	2398
Loop.....	2398
Compensation	2400
Call.....	2402
Adhoc	2404
Boundary	2405
Data	2407
Datasource	2409
Artifact	2411
Text annotation.....	2411
Group	2413
BPMN flows.....	2414
Association	2414
Sequence.....	2416
Message	2418
UML diagram in EJ2 JavaScript Diagram control	2420
UML Class Diagram	2420
UML Class Diagram Shapes	2420
UML Class Relationships	2426
How to add UML child at runtime.....	2435
Adding UML Nodes in Symbol palette	2436
Editing in UML nodes	2439
UML Activity diagram.....	2439
UML Activity diagram Shapes	2439
Connectors in EJ2 JavaScript Diagram control.....	2444
Create connector	2444
Add connectors through connectors collection.....	2444
Add connector at runtime.....	2446
Connectors from palette.....	2447
Draw connectors	2447
Update connector at runtime	2447
Connect nodes	2449
Connections with ports	2451
Segments.....	2457

Straight.....	2457
Orthogonal	2460
Avoid overlapping	2465
How to customize Orthogonal Segment Thumb Shape.....	2467
Bezier	2472
Avoid overlapping with bezier	2477
Decorator	2481
Padding	2483
Hit padding.....	2485
Flip.....	2487
Bridging	2489
Corner radius.....	2492
Appearance	2493
Segment appearance	2494
Decorator appearance	2496
Interaction.....	2497
Automatic line routing	2498
Constraints	2501
Custom properties	2503
Stack order	2503
Enable Connector Splitting.....	2505
See Also	2506
Group in EJ2 JavaScript Diagram control	2506
Create group	2507
Add group when initializing diagram	2507
Add group at runtime	2510
Add children To group at runtime	2512
Remove children from group at runtime.....	2512
Container.....	2514
Difference between a basic group and containers	2517
Interaction.....	2518
See Also	2518
Swim lane in EJ2 JavaScript Diagram control.....	2518
Create a swimlane.....	2518
Lanes	2525

Phase	2542
Add swimlane to palette	2549
Limitations	2552
Labels in EJ2 JavaScript Diagram control	2553
Create annotation	2553
Add annotations at runtime	2554
Remove annotation	2555
Update annotation at runtime	2556
Alignment	2558
Offset	2558
Horizontal and vertical alignment	2559
Annotation alignment with respect to segments	2562
Margin	2563
Text align	2564
Hyperlink	2565
Template Support for Annotation	2567
Wrapping	2568
Text overflow	2570
Appearance	2571
Interaction	2573
Edit	2575
Read-only annotations	2575
Drag Limit	2576
Multiple annotations	2577
Constraints	2579
Ports in EJ2 JavaScript Diagram control	2579
Create port	2580
Add ports when initializing nodes	2580
Add ports at runtime	2581
Remove ports at runtime	2583
Update port at runtime	2585
Appearance	2587
Offset	2589
Constraints	2589
Constraints in EJ2 JavaScript Diagram control	2589

Diagram constraints	2589
Node constraints	2589
Connector constraints	2590
Port constraints	2590
Annotation constraints	2591
Selector constraints	2592
Snap constraints	2592
Boundary constraints	2593
Inherit behaviors	2593
Bitwise operations	2594
Add operation	2594
Remove Operation	2594
Check operation	2594
Interaction in EJ2 JavaScript Diagram control	2594
Selection	2594
Single selection	2595
Selecting a group	2595
Multiple selection	2595
Select/Unselect elements using program	2596
Select entire elements in diagram programmatically	2596
Drag	2596
Resize	2597
Customize the resize-thumb	2597
Rotate	2599
Connection editing	2600
End point handles	2600
Straight segment editing	2600
Orthogonal thumbs	2601
Bezier thumbs	2602
Drag and drop nodes over other elements	2602
User handles	2602
Alignment	2603
Offset	2603
Side	2603
Horizontal and vertical alignments	2603

Margin	2603
Notification for the mouse button clicked.....	2603
Appearance	2603
Zoom pan	2606
Zoom pan status.....	2606
Keyboard	2607
See Also	2608
Tools in EJ2 JavaScript Diagram control.....	2608
Drawing tools	2608
Shapes	2608
Connectors	2610
Text	2611
Polyline Connector	2614
Tool selection	2615
Events.....	2617
Freehand Drawing.....	2618
Grid lines in EJ2 JavaScript Diagram control	2619
Customize the gridlines visibility.....	2620
Appearance	2621
Line intervals	2622
Snapping.....	2624
Snap to lines.....	2624
Customization of snap intervals.....	2625
Snap to objects.....	2626
Page settings in EJ2 JavaScript Diagram control	2628
Page size and appearance	2628
Set background image.....	2630
Multiple page and page breaks.....	2632
Boundary constraints	2634
Scroll settings in EJ2 JavaScript Diagram control	2636
Get current scroll status.....	2636
Define scroll status.....	2636
Update scroll status	2637
AutoScroll.....	2638
Autoscroll border	2640

Scroll limit	2642
Scroll Padding.....	2643
Scrollable Area	2645
UpdateViewport.....	2646
Data binding in EJ2 JavaScript Diagram control.....	2646
Local data	2647
Remote data.....	2649
CRUD	2651
Read DataSource.....	2651
How to perform Editing at runtime	2652
InsertData.....	2652
DeleteData	2654
See Also	2654
Automatic layout in EJ2 JavaScript Diagram control	2654
Layout modes.....	2654
Hierarchical layout	2655
Radial tree layout	2657
Organizational Chart	2663
Symmetric layout	2673
Mind Map layout.....	2675
Tree Orientation in layout.....	2675
Complex hierarchical tree	2677
Customize layout.....	2682
Accessibility in EJ2 JavaScript Diagram control.....	2693
WAI-ARIA attributes.....	2694
Aria-label	2694
Keyboard interaction	2695
Ensuring accessibility	2696
See also	2696
Commands in EJ2 JavaScript Diagram control	2696
Align	2696
Distribute	2699
Sizing	2702
Clipboard.....	2704
Grouping	2706

Z-Order command.....	2708
Zoom	2714
Nudge command.....	2715
Nudge by using arrow keys	2716
BringIntoView	2716
BringToCenter.....	2716
FitToPage command	2717
Command manager.....	2718
Custom command	2718
Modify the existing command	2719
See Also	2720
Undo redo in EJ2 JavaScript Diagram control	2720
Undo and redo	2720
Undo/redo through shortcut keys	2721
Undo/redo through public APIs	2721
canLog	2723
History change event	2725
Stack Limit	2725
Retain Selection	2727
Virtualization in EJ2 JavaScript Diagram control.....	2727
Virtualization in Diagram	2727
Serialization in EJ2 JavaScript Diagram control.....	2728
Save	2728
Load.....	2728
Prevent Default Values	2728
Export in EJ2 JavaScript Diagram control.....	2729
Exporting options.....	2729
File Name	2729
Format.....	2729
Margin.....	2730
Mode.....	2730
Region	2731
PageSettings.....	2731
Content	2732
Custom bounds	2732

Export diagram with stretch option	2733
Print.....	2733
Limitations.....	2734
Tool tip in EJ2 JavaScript Diagram control	2734
Default tooltip.....	2734
Common tooltip for all nodes and connectors	2735
Tooltip for a specific node/connector.....	2737
Tooltip for Ports	2739
Tooltip template content.....	2742
Tooltip alignments	2744
Tooltip animation.....	2747
User handle in EJ2 JavaScript Diagram control	2749
Alignment.....	2749
Fixed user handles	2752
Initialization an fixed user handles	2752
Customization	2753
Customizing the node fixed user handle	2755
Customizing the connector fixed user handle	2759
Style in EJ2 JavaScript Diagram control	2763
Customizing the connector end point handle.....	2763
Customizing the connector end point handle when connected.....	2764
Customizing the connector end point handle when disabled	2764
Customizing the bezier connector handle	2764
Customizing the bezier connector line	2764
Customizing the resize handle	2765
Customizing the selector pivot line.....	2765
Customizing the selector border.....	2765
Customizing the rotate handle	2765
Customizing the symbolpalette while hovering	2766
Customizing the symbolpalette when selected	2766
Customizing the ruler.....	2766
Customizing the ruler overlap.....	2766
Customizing the text edit.....	2766
Customizing the text edit on selection	2767
Ruler in EJ2 JavaScript Diagram control.....	2767

Adding Rulers to the Diagram	2767
Customizing the Ruler	2768
Layers in EJ2 JavaScript Diagram control	2770
Visible	2770
Lock	2772
Objects	2774
AddInfo.....	2776
Context menu in EJ2 JavaScript Diagram control	2781
Customize context menu	2781
Template Support for Context menu.....	2786
Context menu events.....	2789
Symbol palette in EJ2 JavaScript Diagram control	2792
Create symbol palette.....	2792
Add palettes to SymbolPalette	2793
Customize the palette header	2807
Restrict expansion of the palette panel.....	2814
Stretch the symbols into the palette	2821
Add/Remove symbols to palette at runtime	2827
Customize the size of symbols	2827
Symbol preview.....	2834
Default settings	2841
Adding symbol description for symbols in the palette	2847
Appearance of symbol description	2853
Tooltip for symbols in symbol palette	2860
Palette interaction	2868
DragEnter	2869
DragLeave	2869
DragOver	2869
See Also	2869
Overview in EJ2 JavaScript Diagram control	2869
Create overview	2869
Ej1 api migration in EJ2 JavaScript Diagram control	2873
Background	2873
Bridging	2874
CommandManager	2874

Connectors	2878
ContextMenu	2896
DataSourceSettings	2898
DefaultSettings	2906
DrawType	2907
EnableAutoScroll	2907
EnableContextMenu	2908
GetCustomCursor	2908
GetCustomProperty	2908
GetCustomTool	2909
Height	2909
HistoryManager	2910
LabelRenderingMode	2915
Layout	2915
Nodes	2919
NodeTemplate	2952
Overview	2953
Layers	2953
Annotations	2954
PageSettings	2959
SymbolPalette	2960
ScrollSettings	2962
RulerSettings	2964
SnapSettings	2966
ZoomFactor	2967
Tool	2967
ShowTooltip	2968
SelectedItems	2968
SerializationSettings	2972
How to load EJ1 diagram in EJ2 diagram	2972
Tooltip	2973
How to load EJ1 diagram in EJ2 diagram	2975
Dialog	2976
Template in EJ2 JavaScript Dialog control	2976
Header	2976

Footer.....	2976
Content	2976
See Also	2978
Animation in EJ2 JavaScript Dialog control	2978
Resize in EJ2 JavaScript Dialog control.....	2981
Dialog utility in EJ2 JavaScript Dialog control	2982
Alert dialog.....	2983
Confirm dialog.....	2985
Close utility dialog.....	2987
Style in EJ2 JavaScript Dialog control	2989
Customizing the dialog header	2989
Customizing the dialog content	2989
Customizing modal dialog overlay	2989
Customizing the dialog resize icon.....	2990
Customizing the dialog close button.....	2990
Customizing the dialog footer button.....	2990
Localization in EJ2 JavaScript Dialog control.....	2991
Loading translations.....	2991
Accessibility in EJ2 JavaScript Dialog control	2992
WAI-ARIA attributes.....	2993
Keyboard interaction	2994
See Also	2996
Ensuring accessibility	2996
See also	2996
How To	2996
Create nested dialog in EJ2 JavaScript Dialog control	2996
Position the dialog on center of the page on scrolling in EJ2 JavaScript Dialog control	2998
Load dialog content using ajax in EJ2 JavaScript Dialog control.....	3000
Render a dialog without header in EJ2 JavaScript Dialog control.....	3000
Show dialog with full screen in EJ2 JavaScript Dialog control	3002
Display a dialog with custom position in EJ2 JavaScript Dialog control.....	3004
Prevent closing of modal dialog in EJ2 JavaScript Dialog control	3005
Prevent the focus on the first element in EJ2 JavaScript Dialog control	3008
Prevent opening of the dialog in EJ2 JavaScript Dialog control	3010
Read all the values from dialog on button click in EJ2 JavaScript Dialog control	3013

Customize the dialog appearance in EJ2 JavaScript Dialog control	3016
Close dialog while click on outside of dialog in EJ2 JavaScript Dialog control	3017
Add an icons to dialog buttons in EJ2 JavaScript Dialog control.....	3019
Add a minimize maximize buttons in EJ2 JavaScript Dialog control	3022
Setting max height to the dialog in EJ2 JavaScript Dialog control	3025
Ej1 api migration in EJ2 JavaScript Dialog control	3027
Accessibility and Localization.....	3027
Header.....	3027
Footer.....	3029
Content	3029
Animation.....	3030
Draggable and resizing.....	3031
Target	3032
Position	3033
Visibility.....	3033
Dialog Mode.....	3033
Tooltip	3034
Control State	3034
State Maintenance.....	3034
Common.....	3034
DocumentEditor.....	3038
Overview	3038
Key Features.....	3038
Supported Web platforms	3039
Getting started in EJ2 JavaScript Document editor control.....	3039
Component Initialization.....	3039
Server side dependencies	3050
Frequently Asked Questions	3050
Feature module in EJ2 JavaScript Document editor control	3050
Import in EJ2 JavaScript Document editor control	3053
Import document from local machine	3054
Convert word documents into SFDT	3055
Compatibility with Microsoft Word	3058
See Also	3059
Export in EJ2 JavaScript Document editor control.....	3059

SFDT export	3060
Word export	3061
Text export	3062
Export as blob	3063
See Also	3066
Web services in EJ2 JavaScript Document editor control	3066
Which operations require server-side interaction.....	3066
Required Web API structure	3067
Customize the expected method name.....	3068
Modify the XMLHttpRequest before request send	3068
Server Deployment	3069
Word processor server docker image overview in EJ2 JavaScript Document editor control	3069
Provide your license key for activation.....	3070
Provide your license key for activation.....	3072
Provide your license key for activation.....	3073
Provide your license key for activation.....	3074
How to deploy word processor server docker container in azure app service in EJ2 JavaScript Document editor control	3075
How to deploy word processor server docker container in azure kubernetes service in EJ2 JavaScript Document editor control	3076
How to publish documenteditor web api application in azure app service from visual studio in EJ2 JavaScript Document editor control	3079
How to deploy documenteditor java web api in azure in EJ2 JavaScript Document editor control	3081
Accessibility in Angular Document editor component	3084
Keyboard interaction	3084
Ensuring accessibility	3085
See also	3085
Image in EJ2 JavaScript Document editor control	3085
Alternate text	3087
Image resizing	3087
Changing size.....	3088
Text wrapping style	3088
Positioning the image	3088
See Also	3088
Shapes in EJ2 JavaScript Document editor control.....	3088

Supported shapes	3088
Text box Shape	3089
Shape Resizer	3089
Text wrapping style	3089
Positioning the shape.....	3089
Text wrapping style in EJ2 JavaScript Document editor control.....	3090
In-Line with Text.....	3090
In Front of Text.....	3090
Top and Bottom	3090
Behind	3091
Square	3091
Bookmark in EJ2 JavaScript Document editor control.....	3091
Add bookmark.....	3092
Select Bookmark	3092
Delete Bookmark	3092
Get Bookmark from document.....	3092
Get Bookmark from selection	3092
Replace bookmark content.....	3092
Show or Hide bookmark.....	3093
Bookmark Dialog.....	3093
See Also	3094
Link in EJ2 JavaScript Document editor control.....	3095
Navigate a hyperlink	3095
Copy link.....	3097
Add hyperlink	3097
Customize screen tip.....	3098
Remove hyperlink	3099
Hyperlink dialog	3099
See Also	3101
Table in EJ2 JavaScript Document editor control	3101
Create a table.....	3101
Insert rows	3101
Delete table.....	3102
Delete row.....	3103
Delete column.....	3103

Merge cells.....	3103
Positioning the table	3103
How to work with tables	3103
See Also	3107
Table of contents in EJ2 JavaScript Document editor control	3107
Inserting table of contents.....	3107
Update or edit table of contents	3110
See Also	3111
Header footer in EJ2 JavaScript Document editor control	3111
Go to header footer region.....	3111
Header and footer distance	3111
Close header footer region	3112
Link to previous.....	3112
See Also	3113
Text format in EJ2 JavaScript Document editor control	3113
Bold	3113
Italic.....	3113
Underline property	3113
Strikethrough property	3114
Superscript property	3114
Subscript property	3115
Size	3115
Color.....	3115
Font	3116
Highlight color.....	3116
Toolbar with options for text formatting.....	3116
See Also	3120
Paragraph format in EJ2 JavaScript Document editor control.....	3120
Indentation.....	3120
Special indentation	3120
Increase indent	3121
Decrease indent	3121
Text alignment	3121
Line spacing and its type	3122
Paragraph spacing.....	3122

Pagination properties.....	3123
Paragraph Border	3123
Show or Hide Paragraph marks.....	3124
Toolbar with paragraph formatting options	3124
See Also	3128
Styles in EJ2 JavaScript Document editor control	3128
Styles definition overview	3128
Default style	3129
Style hierarchy	3129
Defining new styles	3130
Applying a style	3132
Get Styles	3133
Modify an existing style	3133
List format in EJ2 JavaScript Document editor control	3134
Create bullet list	3134
Create numbered list	3134
Clear list.....	3134
Working with lists	3135
Editing numbered list.....	3137
See Also	3137
Table format in EJ2 JavaScript Document editor control	3137
Cell margins.....	3137
Background color	3138
Cell spacing.....	3138
Cell vertical alignment.....	3138
Table alignment	3139
Cell width	3139
Table width	3139
Apply borders.....	3140
Working with row formatting	3141
See Also	3142
Section format in EJ2 JavaScript Document editor control	3142
Page size.....	3142
Page margins.....	3143
Header distance	3143

Footer distance	3143
Columns	3143
Breaks.....	3144
See Also	3144
Comments in EJ2 JavaScript Document editor control.....	3144
Add a new comment.....	3144
Comment navigation.....	3144
Delete comment	3145
Delete all comment.....	3145
Protect the document in comments only mode.....	3145
Track changes in EJ2 JavaScript Document editor control	3146
Enable track changes in Document Editor	3146
Get all tracked revisions.....	3146
Accept or Reject all changes programmatically	3147
Accept or reject a specific revision	3147
Navigate between the tracked changes	3148
Filtering changes based on user.....	3148
Protect the document in track changes only mode.....	3149
Events.....	3150
Fields in EJ2 JavaScript Document editor control	3151
Adding Fields.....	3151
Update fields.....	3151
Get field info	3152
Set field info	3152
See Also	3152
Form fields in EJ2 JavaScript Document editor control	3153
Insert form field	3153
Get form field names	3153
Get form field properties	3153
Set form field properties.....	3154
Export form field data	3154
Import form field data	3155
Reset form fields	3155
Protect the document in form filling mode	3155
Clipboard in EJ2 JavaScript Document editor control.....	3156

Copy	3156
Cut	3156
Paste.....	3156
Local paste (copy/paste within control)	3156
Paste with formatting	3157
See Also	3158
Collaborative Editing (preview).....	3158
Prerequisites	3159
How to enable collaborative editing in client side.....	3159
How to enable collaborative editing in ASP.NET Core.....	3161
History in EJ2 JavaScript Document editor control.....	3167
Enable or disable history.....	3167
Undo and redo	3168
Stack size.....	3168
See Also	3168
Find and replace in EJ2 JavaScript Document editor control	3168
Options pane.....	3168
Search.....	3170
Search results.....	3171
SearchResultsChange event.....	3172
Customize find and replace.....	3172
Keyboard shortcut in EJ2 JavaScript Document editor control	3175
Text formatting	3175
Paragraph formatting.....	3175
Clipboard.....	3175
Keyboard shortcut to navigate around the document	3176
Keyboard shortcut to extend selection.....	3176
Find and Replace.....	3177
Create, Save and Print document	3177
Edit Operation.....	3177
Insert special characters	3177
Dialog	3177
See Also	3177
Scrolling zooming in EJ2 JavaScript Document editor control.....	3178
Zooming	3182

Page Fit Type	3183
Zoom option using UI.....	3183
Print in EJ2 JavaScript Document editor control	3187
Improve print quality	3190
Print using window object	3190
Page setup.....	3190
See Also	3192
Dialog in EJ2 JavaScript Document editor control	3193
Font Dialog	3193
Paragraph dialog	3194
Table dialog	3196
Bookmark dialog	3197
Hyperlink dialog	3198
Table of contents dialog.....	3200
Styles Dialog	3201
Style dialog	3203
List dialog	3204
Borders and shading dialog.....	3206
Table options dialog.....	3207
Table properties dialog	3209
Page setup dialog	3210
Column dialog	3211
See Also	3213
R t l in EJ2 JavaScript Document editor control	3213
Chart in EJ2 JavaScript Document editor control	3220
Supported Chart Types	3240
Content control in EJ2 JavaScript Document editor control.....	3240
Types of Content Controls	3240
Restrict editing in EJ2 JavaScript Document editor control.....	3241
Set current user	3241
Highlighting the text area	3241
Restrict Editing Pane	3241
See Also	3248
Spell check in EJ2 JavaScript Document editor control	3248
Features	3249

Enable SpellCheck	3249
Disable SpellCheck	3249
Spell check settings	3249
Context menu.....	3251
Global local in EJ2 JavaScript Document editor control	3253
Localization	3253
Document Editor.....	3253
Color Picker	3257
Notes in EJ2 JavaScript Document editor control.....	3258
Insert footnotes	3258
Insert endnotes.....	3258
Update or edit footnotes and endnotes	3259
How To	3259
Override the keyboard shortcuts in EJ2 JavaScript Document editor control.....	3259
Customize context menu in EJ2 JavaScript Document editor control	3262
Customize tool bar in EJ2 JavaScript Document editor control	3267
Change document view in EJ2 JavaScript Document editor control	3268
Open default document in EJ2 JavaScript Document editor control.....	3269
Read only default in EJ2 JavaScript Document editor control	3274
Open document by address in EJ2 JavaScript Document editor control.....	3280
Deploy document editor component for mobile in EJ2 JavaScript Document editor control.....	3281
Disable optimized text measuring in EJ2 JavaScript Document editor control	3282
Get the selected content in EJ2 JavaScript Document editor control	3283
Set default format in document editor in EJ2 JavaScript Document editor control.....	3285
Show hide spinner in EJ2 JavaScript Document editor control	3288
Resize document editor in EJ2 JavaScript Document editor control	3291
Export document as pdf in EJ2 JavaScript Document editor control.....	3293
Customize font family drop down in EJ2 JavaScript Document editor control	3297
Auto save document in document editor in EJ2 JavaScript Document editor control.....	3298
Retrieve the bookmark content as text in EJ2 JavaScript Document editor control	3301
Get current word in EJ2 JavaScript Document editor control	3304
Insert page number and navigate to page in EJ2 JavaScript Document editor control.....	3305
Move selection to specific position in EJ2 JavaScript Document editor control	3307
Disable header and footer edit in document editor in EJ2 JavaScript Document editor control...	3308
Insert text in current position in EJ2 JavaScript Document editor control	3310

Change the cursor color in document editor in EJ2 JavaScript Document editor control.....	3313
Hide tool bar and properties pane in EJ2 JavaScript Document editor control	3314
Insert text or image in table programmatically in EJ2 JavaScript Document editor control	3315
Change the default search highlight color in EJ2 JavaScript Document editor control	3317
Optimize sfdt in EJ2 JavaScript Document editor control.....	3318
Disable auto focus in EJ2 JavaScript Document editor control.....	3319
Disable drag and drop in EJ2 JavaScript Document editor control.....	3319
Enable ruler	3320
DropDown Menu	3322
Icons in EJ2 JavaScript Drop down button control	3322
DropDownButton icons.....	3322
Vertical button	3328
See Also	3331
Popup items in EJ2 JavaScript Drop down button control.....	3331
Icons	3331
Navigations	3334
Template	3336
Separator.....	3343
See Also	3347
Accessibility in EJ2 JavaScript Drop down button control	3347
WAI-ARIA attributes.....	3348
Keyboard interaction	3348
Ensuring accessibility	3348
See also	3349
How To	3349
Change caret icon in EJ2 JavaScript Drop down button control.....	3349
Create dropdownbutton with rounded corner in EJ2 JavaScript Drop down button control	3351
Create right to left dropdownbutton in EJ2 JavaScript Drop down button control	3352
Customize icon and width in EJ2 JavaScript Drop down button control	3354
Disable a dropdownbutton in EJ2 JavaScript Drop down button control.....	3356
Group popup items with listview component in EJ2 JavaScript Drop down button control.....	3358
Hide dropdown arrow in EJ2 JavaScript Drop down button control	3360
Open a dialog on popup item click in EJ2 JavaScript Drop down button control.....	3362
Position popup open in EJ2 JavaScript Drop down button control	3365
Underline a character in the item text in EJ2 JavaScript Drop down button control	3367

DropDownList	3368
Tags in EJ2 JavaScript Drop down list control	3368
Select element	3368
UL element	3370
Input element	3371
Data binding in EJ2 JavaScript Drop down list control	3371
Binding local data	3371
Binding remote data	3375
See Also	3376
Templates in EJ2 JavaScript Drop down list control	3376
Item template	3376
Value template	3377
Group template	3379
Header template	3380
Footer template	3381
No records template	3383
Action failure template	3384
See Also	3385
Grouping in EJ2 JavaScript Drop down list control	3385
HTML select	3386
Customization	3388
See Also	3388
Filtering in EJ2 JavaScript Drop down list control	3388
Limit the minimum filter character	3390
Change the filter type	3391
Case sensitive filtering	3393
Diacritics Filtering	3394
See Also	3395
Virtualization in DropDown List	3395
Binding local data	3396
Binding Remote data	3397
Grouping with Virtualization	3399
Filtering with Virtualization	3400
Localization in EJ2 JavaScript Drop down list control	3402
Loading translations	3402

See Also	3403
Style in EJ2 JavaScript Drop down list control	3404
Customizing the appearance of wrapper element	3404
Customizing the dropdown icon's color	3404
Customizing the focus color	3404
Customizing the outline theme's focus color	3404
Customizing the disabled component's text color	3405
Customizing the float label element's focusing color	3405
Customizing the color of the placeholder text	3405
Customizing the background color of focus, hover, and active item's	3406
Customizing the appearance of pop-up element	3406
Adding mandatory asterisk to placeholder and float label.....	3406
Accessibility in EJ2 JavaScript Drop down list control.....	3407
WAI-ARIA attributes.....	3408
Keyboard interaction	3409
Ensuring accessibility	3410
See also	3411
How To	3411
Add item in EJ2 JavaScript Drop down list control	3411
Cascading in EJ2 JavaScript Drop down list control	3412
Clear item in EJ2 JavaScript Drop down list control.....	3415
Close popup in EJ2 JavaScript Drop down list control	3416
Group header in EJ2 JavaScript Drop down list control.....	3417
Highlight filtering in EJ2 JavaScript Drop down list control	3419
Icons support in EJ2 JavaScript Drop down list control	3420
Incremental search in EJ2 JavaScript Drop down list control	3421
Modify data in EJ2 JavaScript Drop down list control.....	3421
Multiple cascading in EJ2 JavaScript Drop down list control	3423
Remote data bind in EJ2 JavaScript Drop down list control	3425
Remove item in EJ2 JavaScript Drop down list control.....	3427
Search on filtering in EJ2 JavaScript Drop down list control.....	3428
Tooltip in EJ2 JavaScript Drop down list control.....	3430
Value change in EJ2 JavaScript Drop down list control.....	3431
Value support in EJ2 JavaScript Drop down list control.....	3433
Achieve virtual scrolling in EJ2 JavaScript Drop down list control	3433

Ej1 api migration in EJ2 JavaScript Drop down list control	3435
DataBinding	3435
Filtering	3435
Template	3436
Virtual Scrolling	3437
Applying CSS	3437
Sorting	3438
Popup	3438
Placeholder	3439
Grouping	3440
Accessibility	3440
Miscellaneous	3440
Selection	3441
Common	3442
Dropdown Tree	3443
Data binding in EJ2 JavaScript Drop down tree control	3443
Local data	3444
Remote data	3447
Prevent Node selection	3449
Templates in EJ2 JavaScript Drop down tree control	3451
Item template	3451
Header template	3452
Footer template	3454
No records template	3456
Action failure template	3457
Custom template to show selected items in input	3458
Checkbox in EJ2 JavaScript Drop down tree control	3461
Auto Check	3463
Select All	3464
Accessibility in EJ2 JavaScript Dropdown Tree component	3466
WAI-ARIA attributes	3467
Keyboard interaction	3468
Ensuring accessibility	3469
See also	3469
FileManager	3469

User interface in EJ2 JavaScript File manager control	3469
Toolbar	3470
Files and folders navigation	3470
View	3471
Context menu.....	3472
File operations in EJ2 JavaScript File manager control	3473
Folder Upload support	3474
File operation request and response Parameters	3479
File request and response contents.....	3495
Action Buttons	3496
Views in EJ2 JavaScript File manager control	3497
Largelcons View	3497
Details View.....	3498
Customization in EJ2 JavaScript File manager control.....	3500
Context menu customization.....	3500
Details view customization	3501
Navigation pane customization	3503
Show/Hide file extension	3504
Show/Hide hidden items.....	3506
Show/Hide thumbnail images in large icons view	3507
Toolbar customization	3508
Upload customization	3510
Tooltip customization	3511
Multiple selection in EJ2 JavaScript File manager control.....	3514
Drag and drop in EJ2 JavaScript File manager control.....	3515
File system provider in EJ2 JavaScript File manager control	3517
ASP.NET Core file system provider	3517
ASP.NET MVC 5 file system provider	3518
ASP.NET Core Azure cloud file system provider	3519
ASP.NET MVC Azure cloud file system provider	3520
ASP.NET Core Amazon S3 cloud file provider	3521
ASP.NET MVC Amazon S3 cloud file provider.....	3522
File Transfer Protocol file system provider	3523
SQL database file system provider.....	3524
NodeJS file system provider.....	3526

Google Drive file system provider.....	3527
Firebase Realtime Database file system provider.....	3528
IBM Cloud Object Storage file provider	3534
Localization in EJ2 JavaScript File manager control.....	3536
Virtualization in EJ2 JavaScript File manager control	3542
Module Injection.....	3542
Enable Virtualization	3542
Limitations for Virtualization	3544
Accessibility in EJ2 JavaScript File Manager component	3544
WAI-ARIA attributes.....	3545
Keyboard interaction	3546
Ensuring accessibility	3547
See also	3547
Access control in EJ2 JavaScript File manager control.....	3547
Access Rules	3547
Permissions	3548
How To	3551
Adding custom item to context menu in EJ2 JavaScript File manager control.....	3551
Adding custom item to toolbar in EJ2 JavaScript File manager control	3553
Enable disable toolbar item in EJ2 JavaScript File manager control.....	3555
Customize custom thumbnail in EJ2 JavaScript File manager control.....	3556
Nested items in EJ2 JavaScript File manager control.....	3557
Change the localization content in EJ2 JavaScript File manager control.....	3562
Create the custom file provider using NodeJS.....	3563
Floating Action Button	3582
Icons in EJ2 JavaScript Floating action button control.....	3582
FAB with icon	3582
FAB with icon and text.....	3583
Styles in EJ2 JavaScript Floating action button control.....	3586
FAB styles	3586
Styles customization	3588
Show text on hover	3588
Positions in EJ2 JavaScript Floating action button control	3590
Custom position	3594
Events in EJ2 JavaScript Floating action button control	3595

created	3595
onclick	3596
Form Validator	3598
Validation rules in EJ2 JavaScript Form validator control	3598
Default Rules	3598
Error messages in EJ2 JavaScript Form validator control	3599
Customizing Error Messages	3600
Customizing Error Placement	3601
Localization in EJ2 JavaScript Form validator control	3603
Loading translations	3604

Syncfusion JavaScript (ES5) UI Controls (Essential JS 2)

Syncfusion JavaScript (ES5) is a modern UI Controls library that has been built from the ground up to be lightweight, responsive, modular and touch friendly. It is written in TypeScript and has no external dependencies. It also includes complete support for Angular, React, Vue, ASP.NET MVC and ASP.NET Core frameworks.

The Syncfusion JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Components list

The Syncfusion JavaScript (ES5) UI controls are listed below.

<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>

```

| | | | |
|------------------------------------|--|--------------------------------------|--------------------------------------|
| | DATA
VISUALIZATION | CALENDARS | DROPDOWNS |
| GRIDS | Charts | Scheduler | AutoComplete |
| DataGrid | Accumulation Chart | Gantt Chart | ListBox |
| Pivot Table | Stock Chart | Calendar | ComboBox |
| TreeGrid | Circular Gauge | DatePicker | Dropdown List |
| Spreadsheet | Linear Gauge | DateRangePicker | Multiselect DropDown |
| FILE VIEWERS &
EDITORS | Diagram | DateTime Picker | DropDown Tree |
| In-place Editor | HeatMap Chart | TimePicker | Mention |
| RichTextEditor | Map | INPUTS | NAVIGATION |
| PDF Viewer | Range Selector | TextBox | Accordion |
| Word Processor | Smith Chart | Input Mask | Carousel |
| LAYOUT | Sparkline Charts | Numeric TextBox | Context Menu |
| Dialog | Barcode | RadioButton | Menu Bar |
| ListView | TreeMap | CheckBox | Ribbon |
| Predefined Dialogs | Bullet Chart | Color Picker | Sidebar |
| Tooltip | Kanban | File Upload | Tabs |
| Splitter | BUTTONS | Range Slider | Toolbar |
| Dashboard | Button | Toggle Switch Button | TreeView |
| Card | ButtonGroup | Signature | File Manager |
| Avatar | Dropdown Menu | Rating | Breadcrumb |
| | Progress Button | FORMS | Pager |
| | SplitButton | Form Validator | AppBar |
| | Chips | Query Builder | NOTIFICATION |
| | Floating Action Button | | Toast |

| | | | |
|--|----------------------------|--|------------------------------|
| | Speed Dial | | Progress Bar |
| | | | Spinner |
| | | | Badge |
| | | | Skeleton |
| | | | Message |

How to best read this user guide

- The best way to get started would be to read the "Getting Started" section of the documentation for the control that you would like to start using first. The "Getting Started" guide gives just enough information that you need to know before starting to write code. This is the only section that we recommend reading end-to-end before starting to write code, all other information can be referred as needed.

- Now that you are familiar with the basics of using the control, the next step would be to start integrating the control into your application. A good starting point would be to refer to the code snippets in the [online sample browser](#) which contains hundreds of code samples, it is very likely that you will find a code sample that resembles your intended usage scenario.

- Another valuable resource is the API reference which provides detailed information on the object hierarchy as well as the settings available on every object.

Getting help

- If you are still not able to find the information that you are looking for in the self-help resources mentioned above then please contact us by creating a support ticket in [our support site](#) or ask your query in Stack Overflow with tag `syncfusion-ej2`.

Syncfusion does not collect any kind of information when our components are used in customer applications.

See also

- [Product Development Life Cycle](#)
- [Getting Started with Syncfusion Javascript ES5 controls](#)

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

Getting Started with Syncfusion JavaScript (ES5) UI control in a quickstart application

This section provides a step-by-step guide to configuring the Syncfusion JavaScript (ES5) UI control (Essential JS 2) and building a simple HTML web application.

Prerequisites

To get started, ensure the following software to be installed in the machine.

- [Essential Studio JavaScript \(Essential JS 2\)](#)
- [Visual Studio Code](#)

Check out the [download and installation](#) section of **Essential Studio JavaScript**. If you are using Syncfusion CDN resources to build your web application, you can skip the Essential Studio JavaScript prerequisite.

Set up development environment

- Create a folder named `~/quickstart` from the required directory and open it in Visual Studio Code.
- Create an HTML file `~/quickstart/index.html` to add the Syncfusion JavaScript (ES5) control's resources.

Add Syncfusion resources

You can access Syncfusion control resources using either of the following methods:

- [Using local scripts and styles](#)
- [Using CDN link for scripts and styles](#)

Syncfusion JavaScript controls have dependencies on other Syncfusion controls, so you must add these dependent control resources to use all their functionality. These dependencies are listed in the corresponding control's getting started documentation. For example, you can find the dependencies for the Grid control [here](#).

Using local scripts and styles

- After installing the [Essential Studio JavaScript \(Essential JS 2\)](#) build, it contains the scripts and styles reference for each individual package and all control resources in a single script and style file. In this getting started, simple Grid control is used as an example.
- Create a folder named `~/quickstart/resources`. Inside that folder, create the `base/styles/`, `popups/styles/`, `grids/styles/` and `scripts` folders. Then, copy/paste the individual scripts and styles from the below installed location to the corresponding `~/quickstart/resources/` location.

Individual control resources:

Dependency script:

(installed location) \Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web
(Essential JS
2)\JavaScript\{DEPENDENCYPACKAGE_NAME}\dist\global\{DEPENDENCYPACKAGE_NAME}.min.js

Control script:

(installed location) \Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web
(Essential JS 2)\JavaScript\{PACKAGE_NAME}\dist\global\{PACKAGE_NAME}.min.js

Example:

Grid's control script:

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS
2)\JavaScript\ej2-grids\dist\global\ej2-grids.min.js

Grid's dependency scripts:

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS
2)\JavaScript\ej2-base\dist\global\ej2-base.min.js

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS
2)\JavaScript\ej2-data\dist\global\ej2-data.min.js

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS
2)\JavaScript\ej2-popups\dist\global\ej2-popups.min.js

Dependency style:

(installed location) \Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web
(Essential JS 2)\JavaScript\{DEPENDENCYPACKAGE_NAME}\styles\material.css

Control style:

(installed location)\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\styles\material.css

Example:

Grid's control Style:

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS 2)\JavaScript\ej2-grids\styles\material.css

Grid's dependency styles:

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS 2)\JavaScript\ej2-base\styles\material.css

C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\20.4.0.38\Web (Essential JS 2)\JavaScript\ej2-popups\styles\material.css

You can also refer to a single script and style file that contains all Syncfusion JavaScript control resources from the following location.

Single script and style reference for all controls:

Script reference for all controls: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\dist\ej2.min.js

Style reference for all controls: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\material.css

Since this file includes all Syncfusion controls, it may impact the website's loading time. To reduce the size of the single file, you can generate custom scripts and styles for a set of specific Syncfusion JavaScript controls using the [Custom Resource Generator \(CRG\)](#) tool. This tool is useful for combining the required control scripts and styles into a single file. To know more about the CRG, refer to this [documentation](#).

- Once copy/paste the individual scripts and styles to the `~/quickstart/resources/` location. Add the required Grid control resources to the `~/quickstart/index.html` file in the following order.

`html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Syncfusion JavaScript (ES5) UI Control</title>
<!-- Essential JS 2 Grid's dependent material themes -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/popups/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Grid's material theme -->
<link href="resources/grids/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Grid's dependent scripts -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Grid's global script -->
<script src="resources/scripts/ej2-grids.min.js" type="text/javascript"></script>
</head>
<body>
...
</body>
</html>
```

Using CDN link for scripts and styles

- Syncfusion hosts every Syncfusion JavaScript (ES5) control as a separate package on 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 (ES5) controls, allowing you to load the scripts and styles for all controls as a single script and style file. In this getting started, simple Grid control is used as an example. Refer to the following CDN scripts and styles link.

Individual control CDN reference

Dependency script:

<https://cdn.syncfusion.com/ej2/22.1.34/{DEPENDENCYPACKAGENAME}/dist/global/{DEPENDENCYPACKAGENAME}.min.js>

Control script:

`https://cdn.syncfusion.com/ej2/22.1.34/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js`

Example:

Grid's control script:

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-grids/dist/global/ej2-grids.min.js>

Grid's dependency scripts:

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-base/dist/global/ej2-base.min.js>

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-data/dist/global/ej2-data.min.js>

<https://cdn.syncfusion.com/ej2/22.1.34/ej2-popups/dist/global/ej2-popups.min.js>

Dependency styles:

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

Control styles:

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

Example:

Grid's control style:

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

Grid's dependency styles:

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

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

You can also refer to a single script and style CDN link that contains all Syncfusion JavaScript control resources as follows:

Single script and style CDN reference for all controls:

Script reference for all controls: <https://cdn.syncfusion.com/ej2/22.1.34/dist/ej2.min.js>

Style reference for all controls: <https://cdn.syncfusion.com/ej2/22.1.34/material.css>

Warning: The un-versioned CDN links which always maintains latest version scripts are deprecated from 2022 Vol1 - 20.1 version. These links are available with 19.4 version scripts to avoid breaking in sites and apps that uses un-versioned links.

- Add the following Grid control CDN link references in the given order to the `~/quickstart/index.html` file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Syncfusion JavaScript (ES5) UI Control</title>
<!-- Essential JS 2 Grid's dependent material theme -->
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Grid's material theme -->
<link href="https://cdn.syncfusion.com/ej2/22.1.34/ej2-grids/styles/material.css" rel="stylesheet"
type="text/css"/>
<!-- Essential JS 2 Grid's dependent script -->
<script src="https://cdn.syncfusion.com/ej2/22.1.34/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/22.1.34/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/22.1.34/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Grid's global script -->
```

```
<script src="https://cdn.syncfusion.com/ej2/22.1.34/ej2-grids/dist/global/ej2-grids.min.js"
type="text/javascript"></script>
</head>
<body>
...
</body>
</html>
`
```

Add Syncfusion control to the application

Now, add the Grid element and Syncfusion JavaScript (ES5) Grid control in the `~/quickstart/index.html` file as follows.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
...
</head>
<body>
<h2>Syncfusion JavaScript (ES5) Grid Control</h2>
<!-- Add the HTML <div> element for grid -->
<div id="Grid"></div>
<script>
//Grid data
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
}
];

// Initialize Essential JS 2 JavaScript Grid control
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' }
  ]
});

// Render initialized Grid control
grid.appendTo('#Grid');
</script>
</body>
</html>
`
```

Run the application

Open the `~/quickstart/index.html` file in the web browser and it will render the Syncfusion JavaScript (ES5) Grid control.

INDEX.HTML

```

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Essential JS 2 Grid</title>
  <!-- Essential JS 2 Grid's dependent material theme -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet" type="text/css"/>
  <!-- Essential JS 2 Grid's material theme -->
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/styles/material.css" rel="stylesheet" type="text/css"/>
  <!-- Essential JS 2 Grid's dependent script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
data/dist/global/ej2-data.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/dist/global/ej2-popups.min.js" type="text/javascript"></script>
  <!-- Essential JS 2 Grid's global script -->
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
grids/dist/global/ej2-grids.min.js" type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
  <h2>Syncfusion JavaScript (ES5) Grid Control</h2>
  <!-- Add the HTML <div> element for grid -->
  <div id="Grid"></div>
  <script>
    //Grid data
    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
    }
};
// Initialize Essential JS 2 JavaScript Grid control
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' }
    ]
});
// render initialized grid
grid.appendTo('#Grid');
</script>
</body>
</html>

```

To learn more about the functionality of the Grid control, refer to the [Grid control](#) section.

[View CDN link reference sample in github](#)

[View local script and style sample in github](#)

See also

- [How to register Syncfusion license key in JavaScript\(ES5\) application](#)

Compatibility with Syncfusion JavaScript (Essential JS 1)

This article provides a step-by-step introduction to configure the Essential JS 1 and Essential JS 2 JavaScript controls in a same web page.

Prerequisites

To work with the Essential JS 1 and the Essential JS 2 controls compatibility in JavaScript (ES5), the below mentioned System requirements are necessary,

- [Essential JS 1 Dependencies](#)
- [Visual Studio Code](#)

Creating JavaScript application with Syncfusion JavaScript (Essential JS 2) control

1. Create a JavaScript (ES5) application with the help of the given Essential JS 2 [Getting Started Documentation](#).

2. Now, the Syncfusion (Essential JS 2) JavaScript Button control rendered successfully in the web page.

Adding Syncfusion JavaScript (Essential JS 1) control in the application

1. Before adding the Essential JS 1 control in the application, you should change the Essential JS 2 compatibility styles in the application.

The compatibility styles of Essential JS 1 and Essential JS 2 must be added in the application to prevent the UI conflicts between the Essential JS 1 and Essential JS 2 controls.

Replace the style file in the **resources** folder from the below location.

Syntax:

Styles: **(installed location)**\Syncfusion\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\styles\compatibility\material.css

Example:

Styles: C:\Program Files (x86)\Syncfusion\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2\styles\compatibility\material.css

If the CDN link is used in the application, then replace it with below link.

Styles: <https://cdn.syncfusion.com/ej2/styles/compatibility/material.css>

2. Now, add the Essential JS 1 script and compatibility styles with its dependencies in the **resources/ej1** location. You can get the Essential JS 1 scripts and styles from the below installed location.

Syntax:

Script:

(installed location)\Syncfusion\{RELEASE_VERSION}\Web (Essential JS 1)\JavaScript\assets\scripts\web\ej.web.all.min.js

Styles:

(installed location)\Syncfusion\{RELEASE_VERSION}\Web (Essential JS 1)\JavaScript\assets\css\web\default-theme\ej.web.all.compatibility.min

Example:

Script:

C:\Program Files (x86)\Syncfusion\16.3.0.17\Web (Essential JS 1)\JavaScript\assets\scripts\web\ej.web.all.min.js

Styles:

C:\Program Files (x86)\Syncfusion\16.3.0.17\Web (Essential JS 1)\JavaScript\assets\css\web\default-theme\ej.web.all.compatibility.min

3. Add the Essential JS 1 file references in the `index.html` with the HTML Button element for rendering Essential JS 1 Button control.

The Essential JS 1 scripts must be added before the Essential JS 2 script reference.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2 - Essential JS 1</title>
<!-- Essential JS 1 default theme -->
<link href="resources/ej1/default-theme/ej.web.all.compatibility.min.css" rel="stylesheet"
type="text/css" />
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css" />
<!-- Essential JS 1 scripts -->
<script src="https://cdn.syncfusion.com/js/assets/external/jquery-1.10.2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/js/assets/external/jquery.easing.1.3.min.js"
type="text/javascript"></script>
<script src="resources/ej1/ej.web.all.min.js" type="text/javascript"></script>
<!-- Essential JS 2 script -->
<script src="resources/ej2.min.js" type="text/javascript"></script>
</head>
<body>
<div style="margin: 50px;">
<!-- Add HTML Button element for Essential JS 2 -->
<button id="btn2">Essential JS 2</button>
</div>
<div style="margin: 50px;">
<!-- Add HTML Button element for Essential JS 1 -->
<button id="btn1">Essential JS 1</button>
</div>
</body>
```

</html>

,

4. Initialize the Essential JS 1 and Essential JS 2 Button control inside the `

`html

<script>

// Extend ej namespace with Syncfusion

\$.extend(ej, Syncfusion);

// Initialize Essential JS 2 JavaScript Button control

var button = new ej.buttons.Button({

isPrimary: true

});

button.appendTo('#btn2');

// Initialize Essential JS 1 JavaScript Button control

\$("#btn1").ejButton();

</script>

,

The `ej` namespace should be extended with `Syncfusion` before initializing the Essential JS 1 and Essential JS 2 controls.

5. Run `~/quickstart/index.html` in the web browser and the Essential JS 1 and Essential JS 2 Button controls will be rendered in the same web page.



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.)

Thank you for choosing to evaluate Essential Studio for JavaScript!

Please choose your preferred evaluation option to proceed.

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

Full installation evaluation experience

Download and install the evaluation version of the full product along with all the required assemblies and demo code.

Choose Platform

☒ Windows ☐ macOS

Choose Download Format

☐ Web Installer ☒ Offline Installer

DOWNLOAD

.EXE

NuGet evaluation experience

No installation required. Get started in 2 minutes

Install assemblies from NuGet

&

Download demo code from
GitHub

Open demo code in
Visual Studio

Version: 20.3.0.47, Released: September 29, 2022

[Release Notes](#) | [License Agreement](#) | [Installation Instructions](#)

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](#)

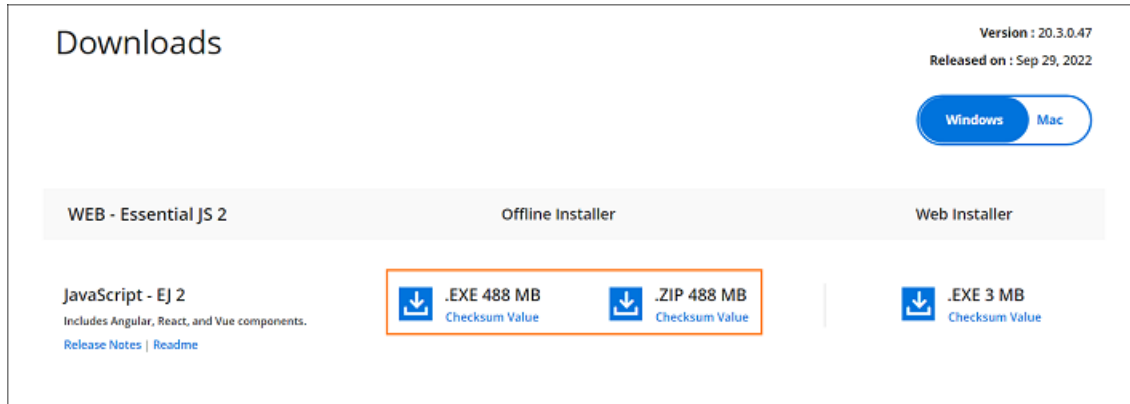
Download 1

More Download Options 2

Copyright © 2001 -2024 Syncfusion Inc.

90

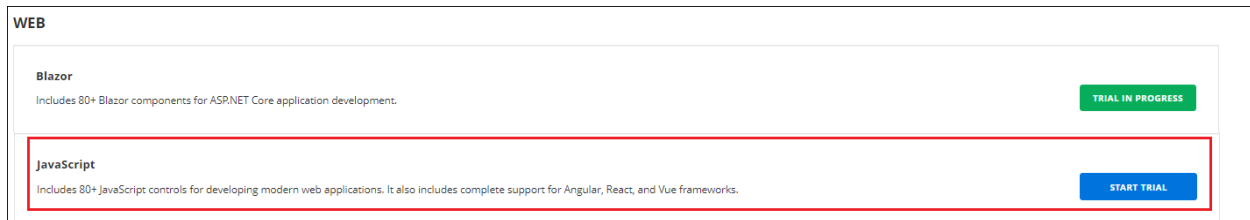
- Click the More Download Options (element 2 in the above screenshot) button to get the Essential Studio JavaScript – EJ2 Offline trial installer which is available in EXE and ZIP format.



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.



- 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

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)

Trial Version :

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

Download 1

More Download Options 2

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. Click the Download (element 1 in the screenshot below) button to download the respective product's installer.
4. The most recent version of the installer will be downloaded from this page.
5. To download older version installers, go to [Downloads Older Versions](#) (element 2 in the screenshot below).
6. You can download other platform\add-on installers by going to More Downloads Options (element 3 in the screenshot below).
7. For Windows OS, EXE and Zip formats are available for download. They are both Offline Installers.

License Downloads and Unlock Keys

License SKU Name
Essential Studio

Latest Official Release :

[What's New](#) | [Generate License File](#) ⓘ | [Get Unlock Key](#) ⓘ

Download Older Versions 2

Download 1

More Download Options 3

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](#).

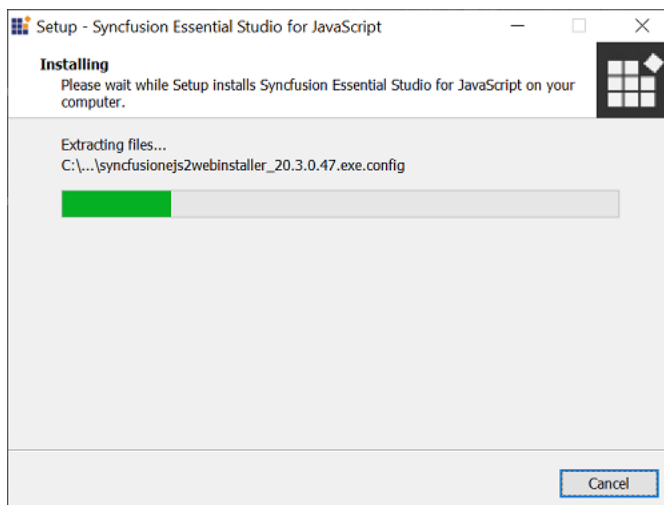
The frameworks listed below are supported in this installer.

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

Installation

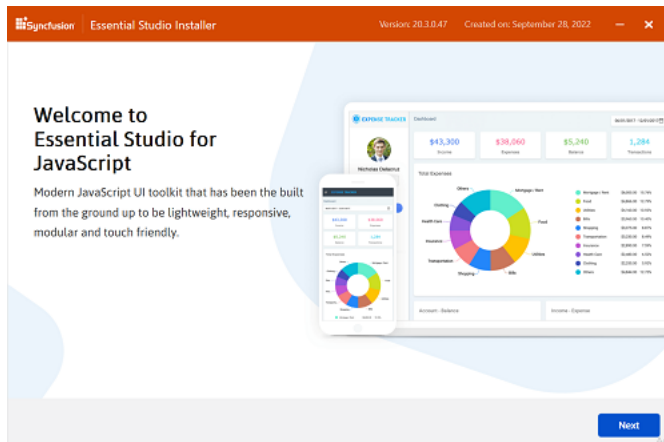
The steps below show how to install Essential Studio JavaScript – EJ2 Web Installer.

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.



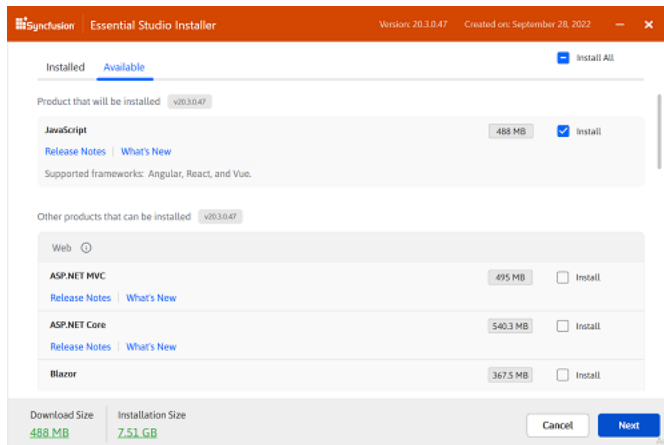
Note: The installer wizard extracts the syncfusionejs2webinstaller_{version}.exe dialog, which displays the package's unzip operation.

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



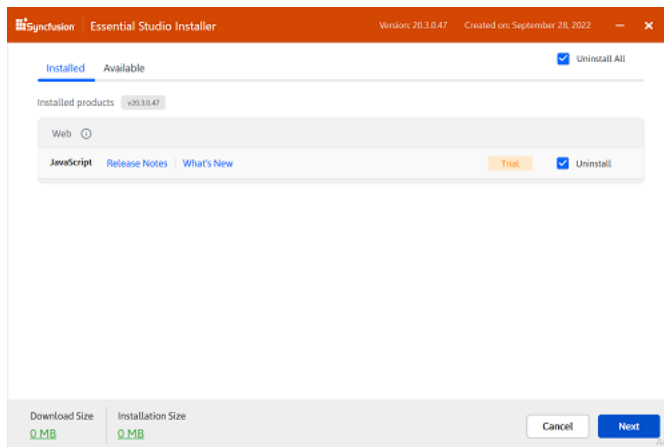
3. The Platform Selection Wizard will appear. From the **Available** tab, select the products to be installed. Select the Install All checkbox to **install all** products.

Available



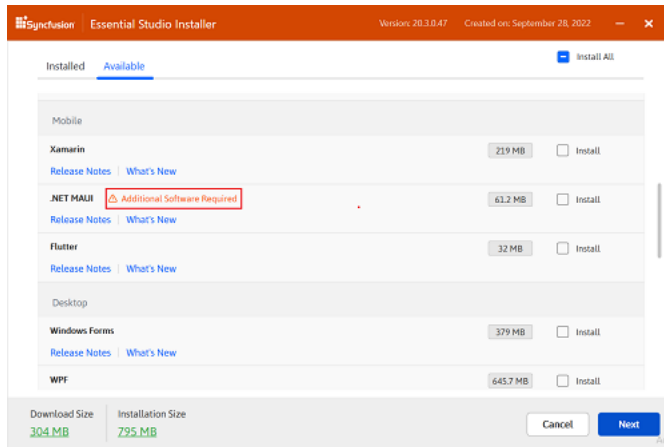
If you have multiple products installed in the same version, they will be listed under the **Installed** tab. You can also select which products to uninstall from the same version. Click the Next button.

Installed

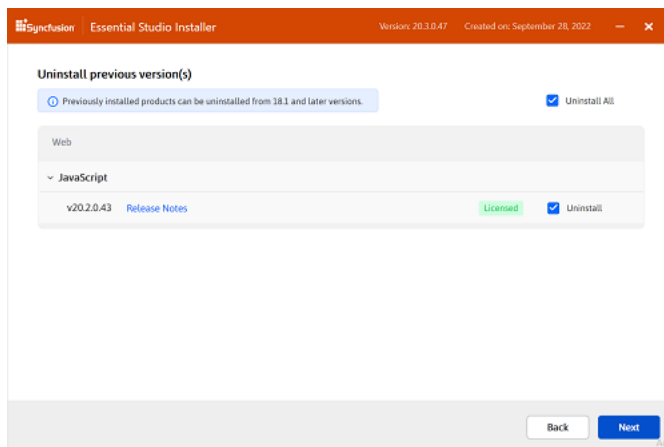


Important: If the required software for the selected product isn't already installed, the **Additional Software Required** alert will appear. You can, however, continue the installation and install the necessary software later.

Required Software

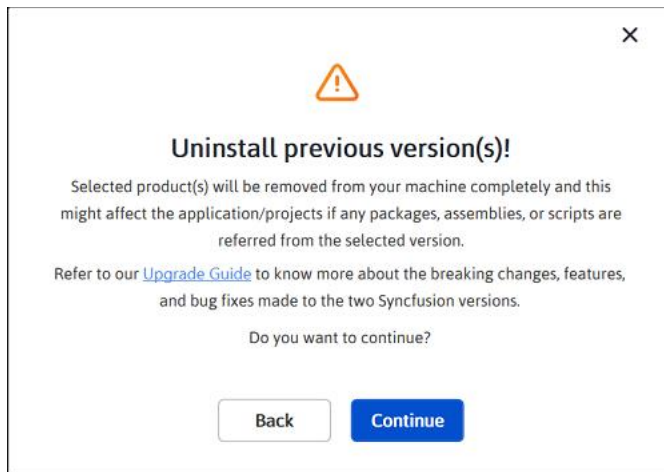


4. If previous version(s) for the selected products are installed, the Uninstall previous version wizard will be displayed. You can see the list of previously installed versions for the products you've chosen here. To remove all versions, check the **Uninstall All** checkbox. Click the Next button.

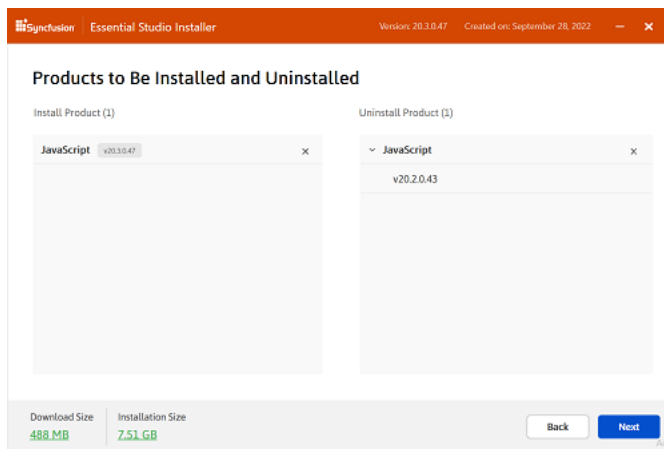


Note: From the 2021 Volume 1 release, Syncfusion has provided option to uninstall the previous versions from 18.1 while installing the new version.

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

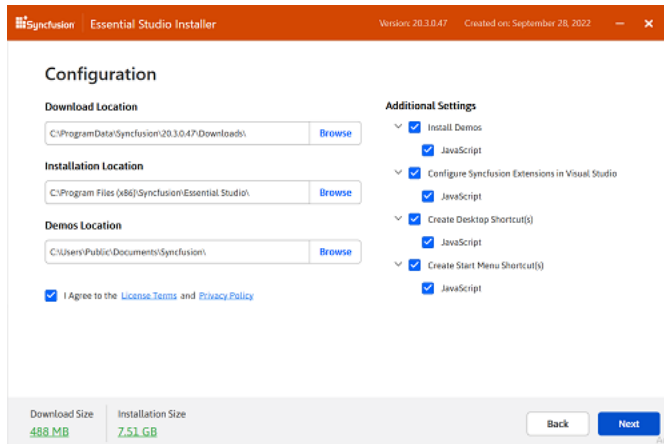


6. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. You can view and modify the list of products that will be installed and uninstalled from this page.



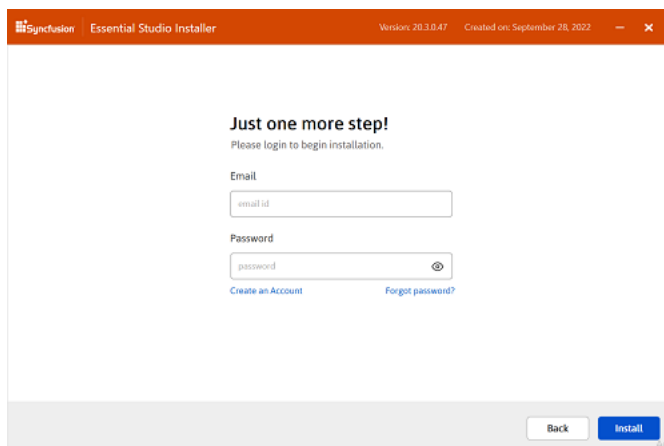
Note: By clicking the **Download Size** and **Installation Size** links, you can determine the approximate size of the download and installation

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.



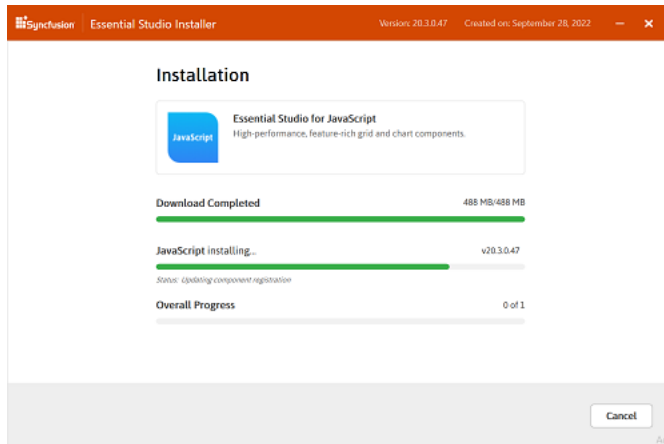
Additional settings

- 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.

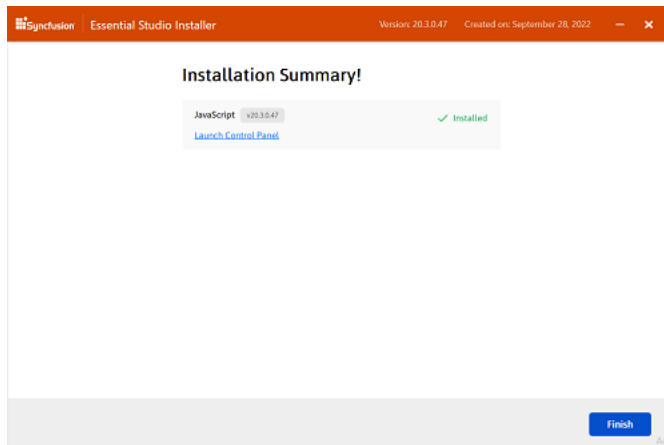


Important: The products you have chosen will be installed based on your Syncfusion License (Trial or Licensed).

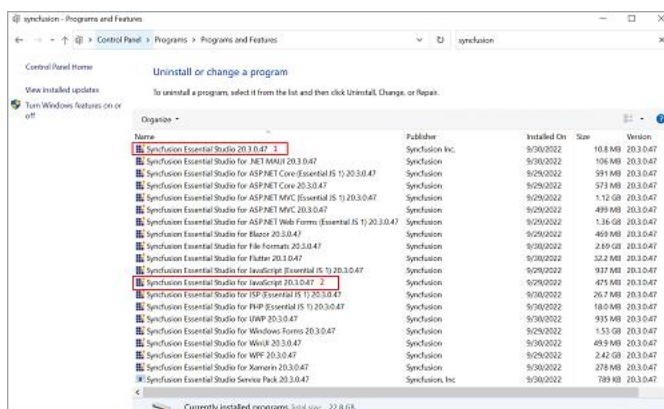
10. The download and installation\uninstallation progress will be displayed as shown below.



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.

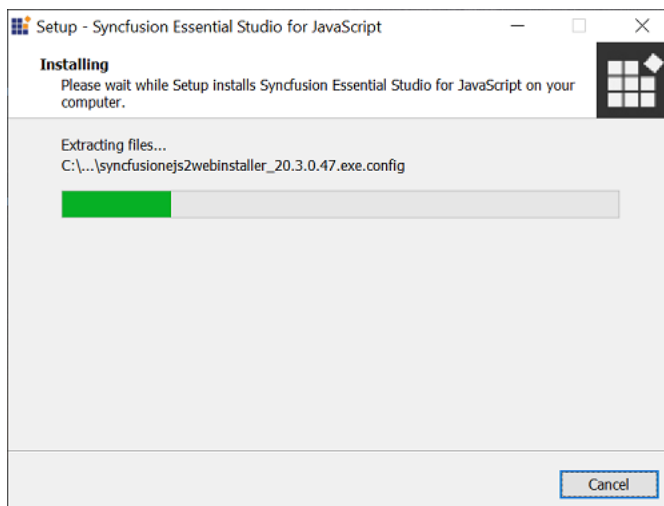
- Uninstall the JavaScript – EJ2 using the Syncfusion JavaScript – EJ2 web installer
- Uninstall the JavaScript – EJ2 from Windows Control Panel

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

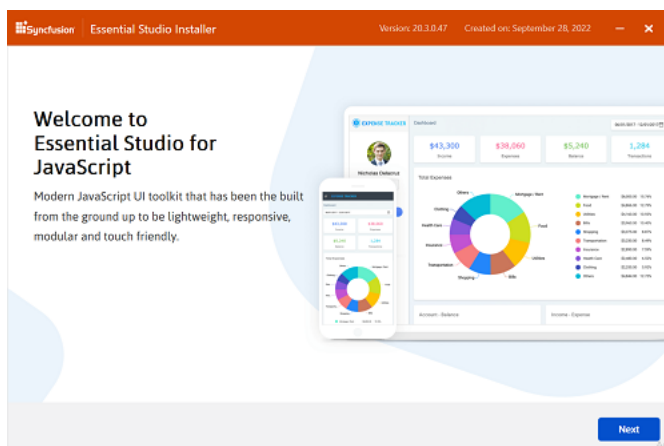
Option 1: Uninstall the JavaScript–EJ2 using the Syncfusion JavaScript–EJ2 web installer

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



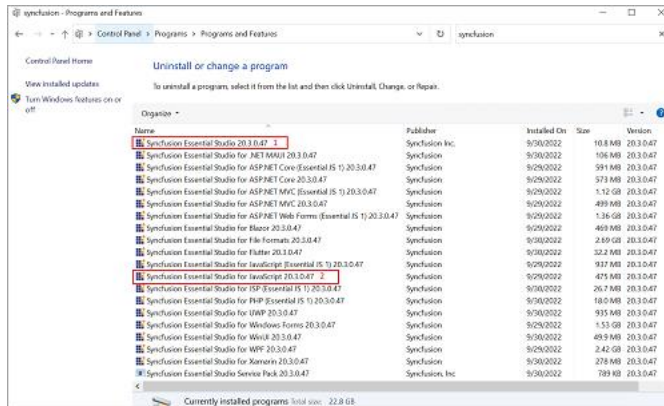
The Syncfusion JavaScript – EJ2 Web Installer’s welcome wizard will be displayed. Click the Next button



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 –

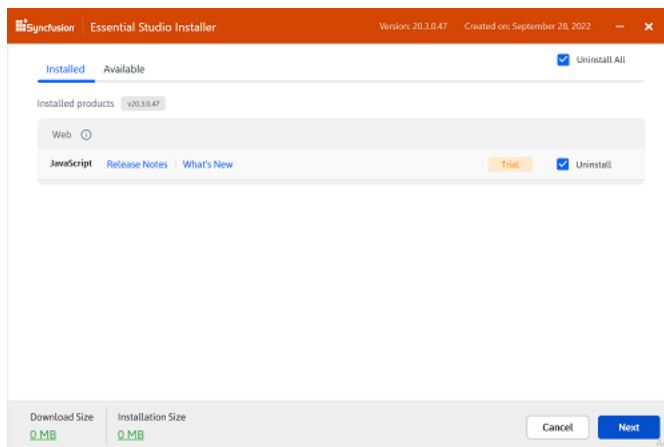
EJ2 alone by selecting the **Syncfusion Essential Studio for JavaScript – EJ2 {version}** entry (element 2 in the below screenshot) from the Windows control panel.



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.

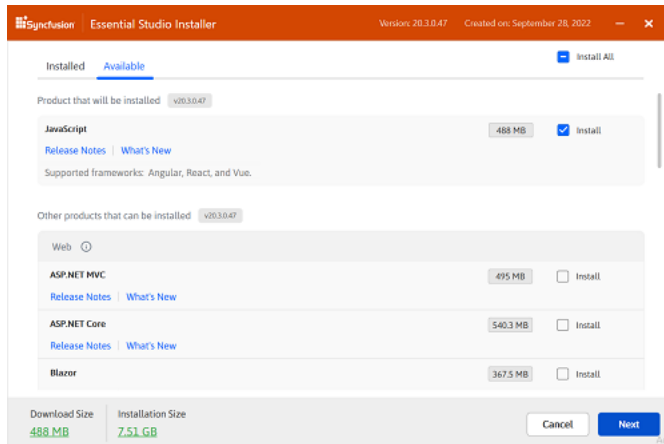
1. The Platform Selection Wizard will appear. From the **Installed** tab, select the products to be uninstalled. To select all products, check the **Uninstall All** checkbox. Click the Next button.

Installed

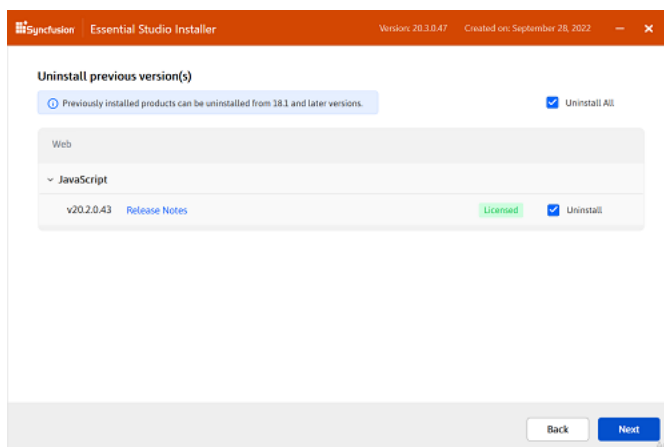


You can also select the products to be installed from the **Available** tab. Click the Next button.

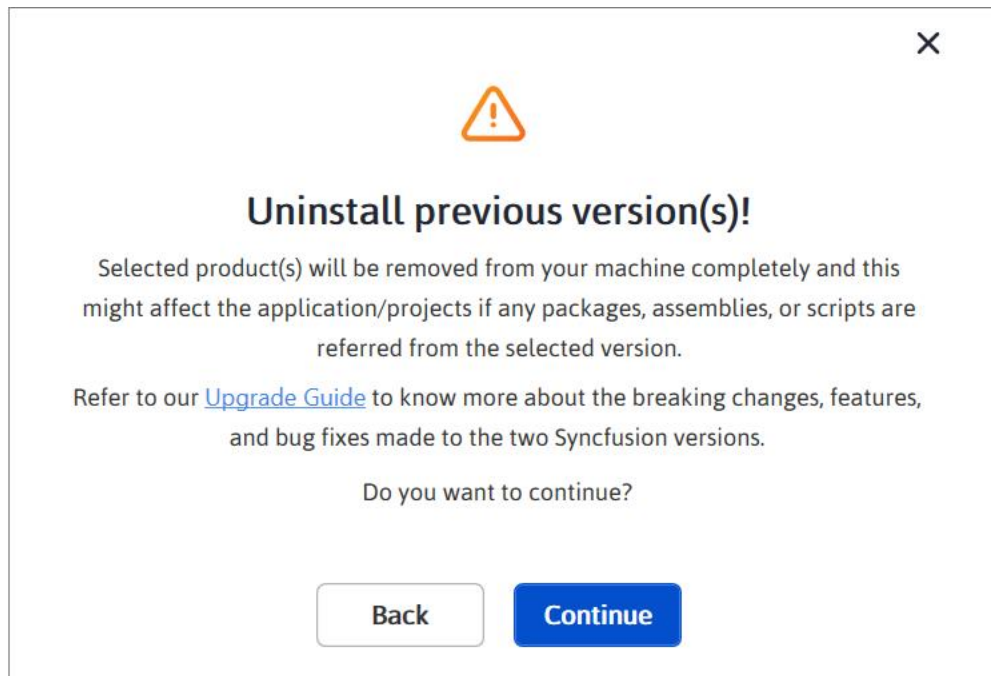
Available



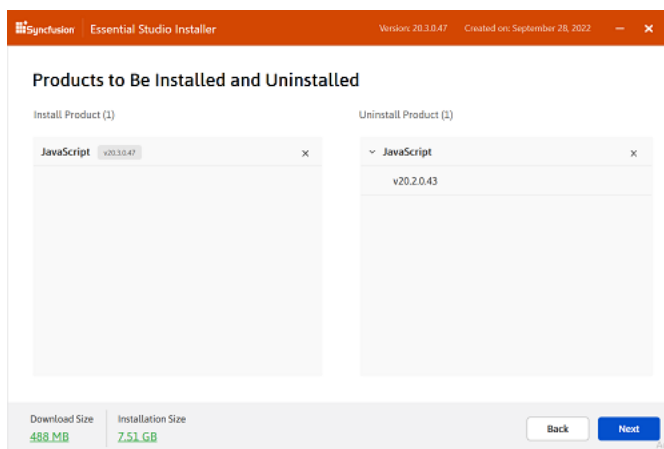
2. If any other products selected for installation, Uninstall previous version wizard will be displayed with previous version(s) installed for the selected products. Here you can view the list of installed previous versions for the selected products. Select **Uninstall All** checkbox to select all the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



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

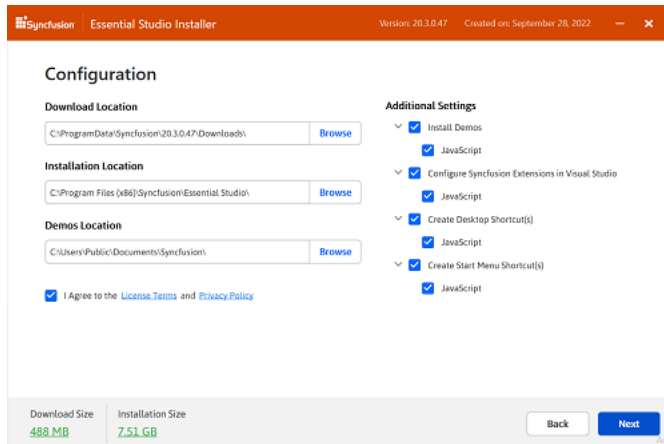


4. The Confirmation Wizard will appear with the list of products to be installed/uninstalled. Here you can view and modify the list of products that will be installed/uninstalled.

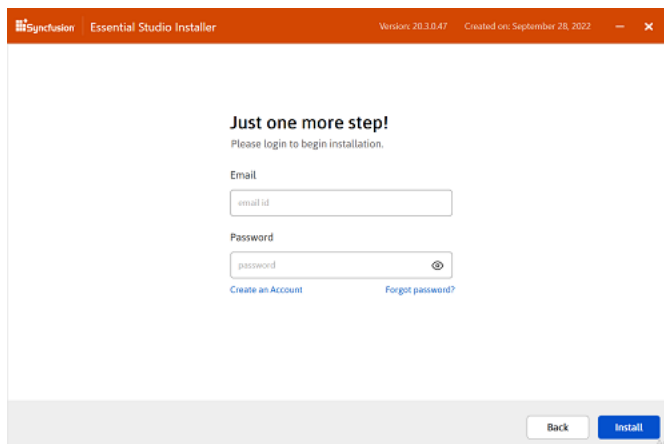


Note: By clicking the **Download Size** and **Installation Size** links, you can determine the approximate size of the download and installation

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.

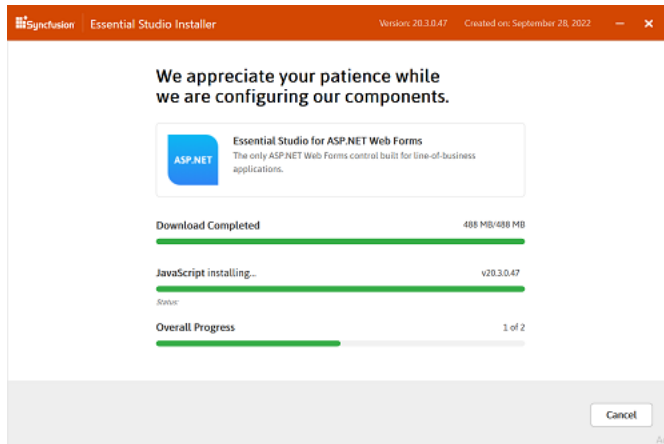


6. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click the Next button.
7. The login wizard will appear. You must enter your Syncfusion email address and password. If you do not already have a Syncfusion account, you can create 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.

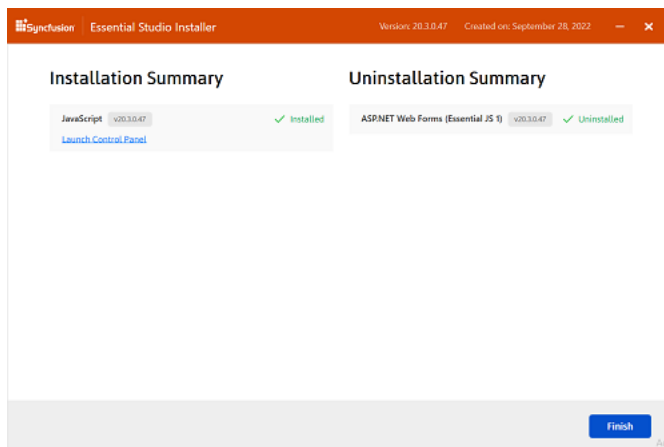


Important: The products you have chosen will be installed based on your Syncfusion License (Trial or Licensed).

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.

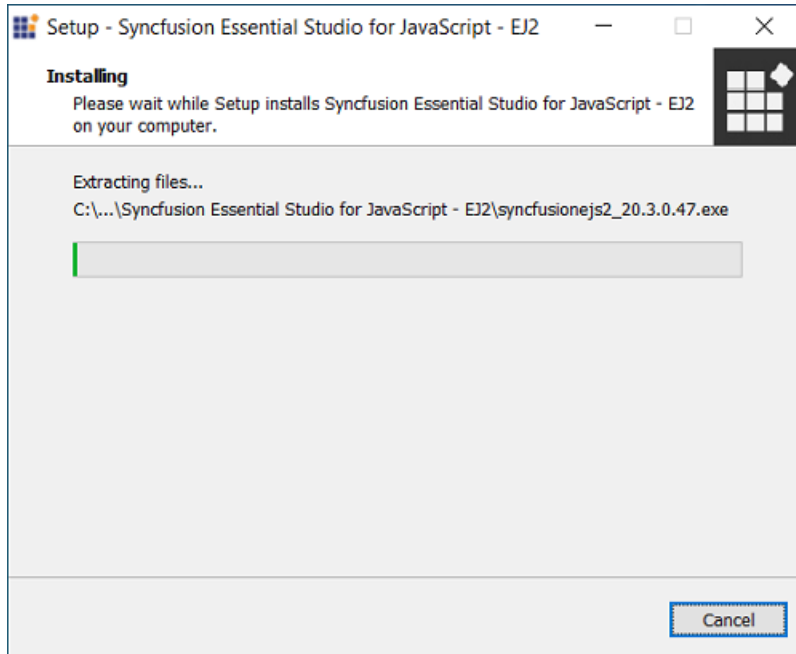
The frameworks listed below are supported in this installer.

- 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

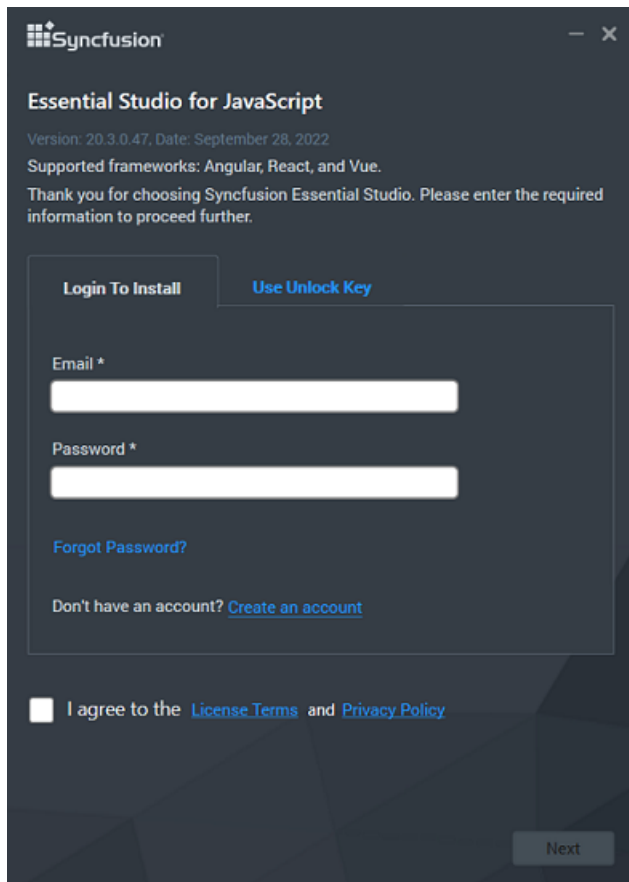


Note: The Installer wizard extracts the syncfusionejs2 (version).exe dialog, which displays the package's unzip operation.

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.



Syncfusion

Essential Studio for JavaScript

Version: 20.3.0.47, Date: September 28, 2022

Supported frameworks: Angular, React, and Vue.

Thank you for choosing Syncfusion Essential Studio. Please enter the required information to proceed further.

Login To Install **Use Unlock Key**

Email *

Password *

[Forgot Password?](#)

Don't have an account? [Create an account](#)

☐ I agree to the [License Terms](#) and [Privacy Policy](#)

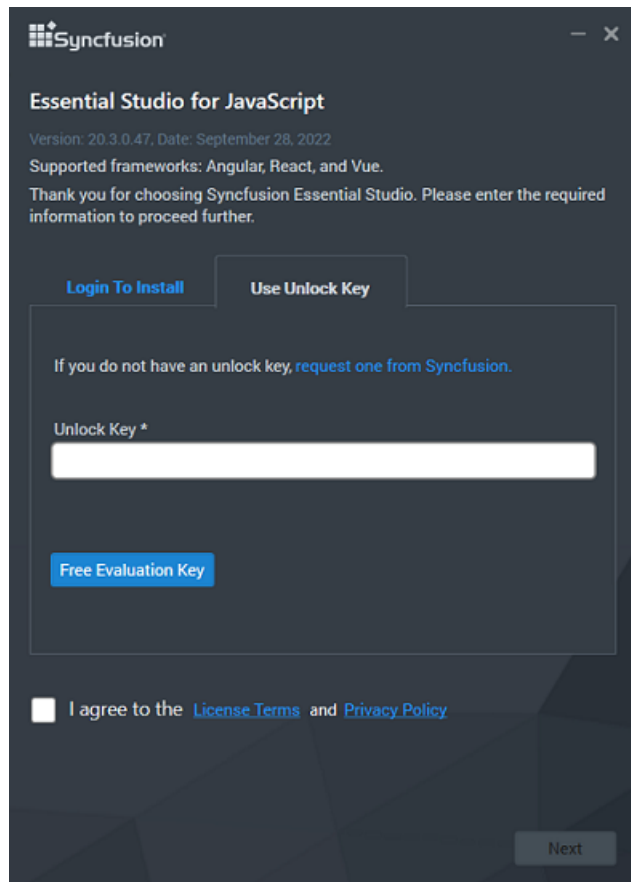
Next

Use Unlock Key

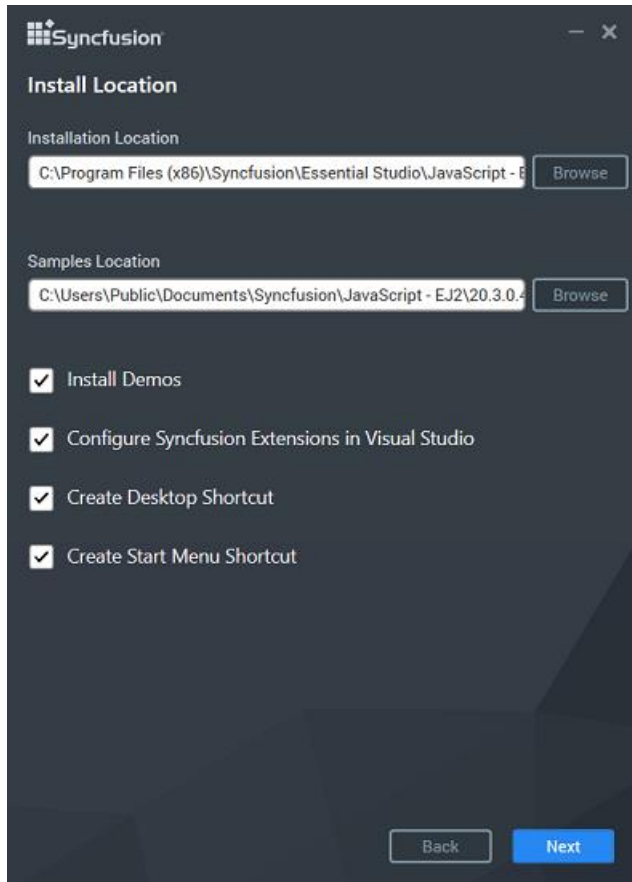
Unlock keys are used to unlock the Syncfusion offline installer, and they are platform and version specific. You should use either Syncfusion licensed or trial Unlock key to unlock Syncfusion JavaScript – EJ2 installer.

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.

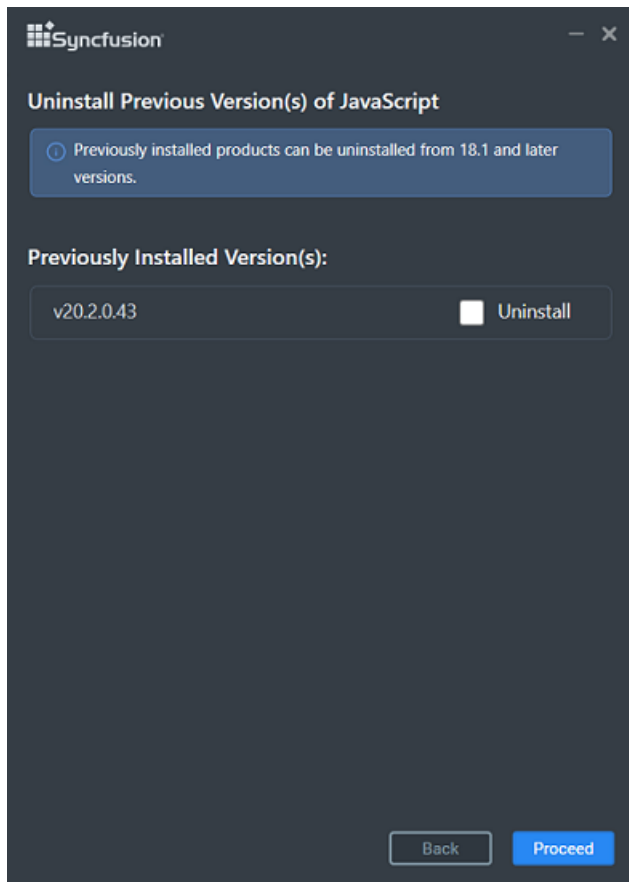


3. After reading the License Terms and Privacy Policy, check the **“I agree to the License Terms and Privacy Policy”** check box. Click the Next button.
4. Change the install and sample locations here. You can also change the Additional settings. Click Next\Install to install with the default settings.



Additional Settings

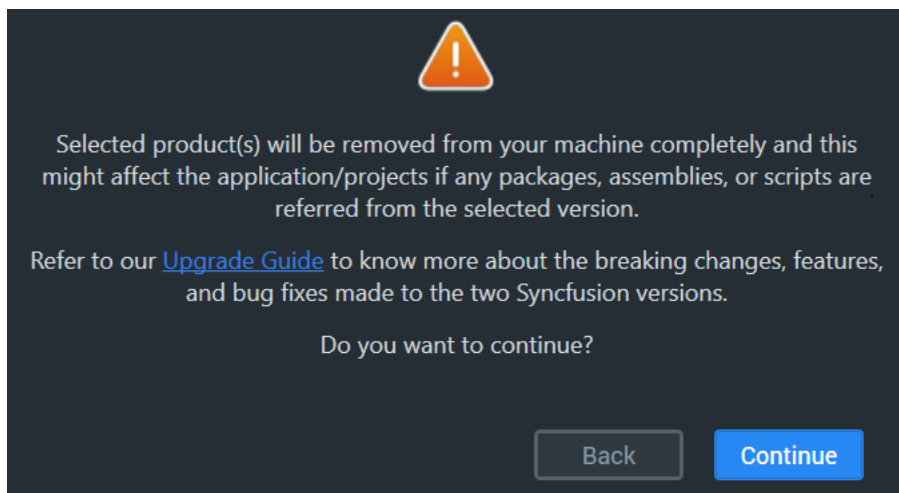
- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box unchecked, if you do not want to install Syncfusion samples
- Select the **Configure Syncfusion Extensions controls in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear 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.

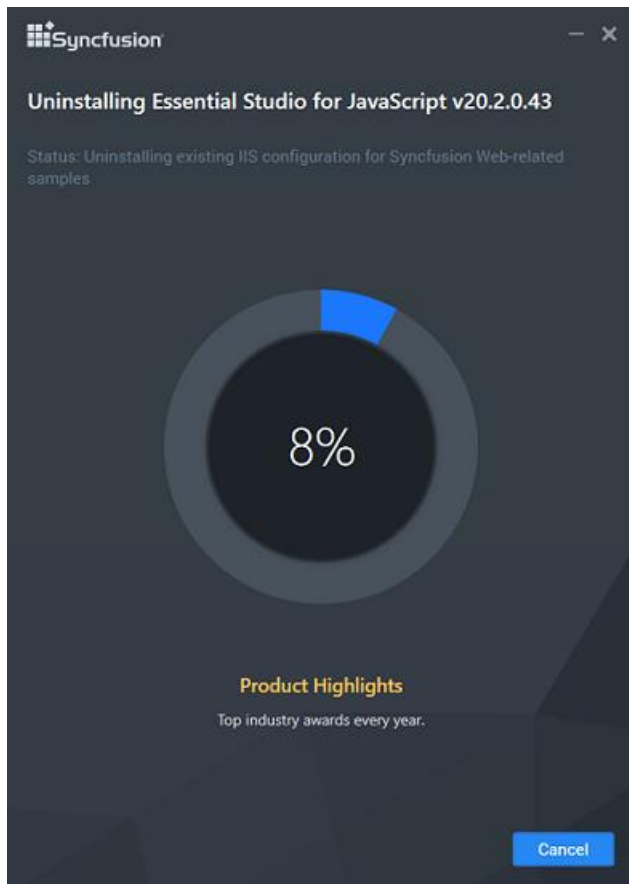


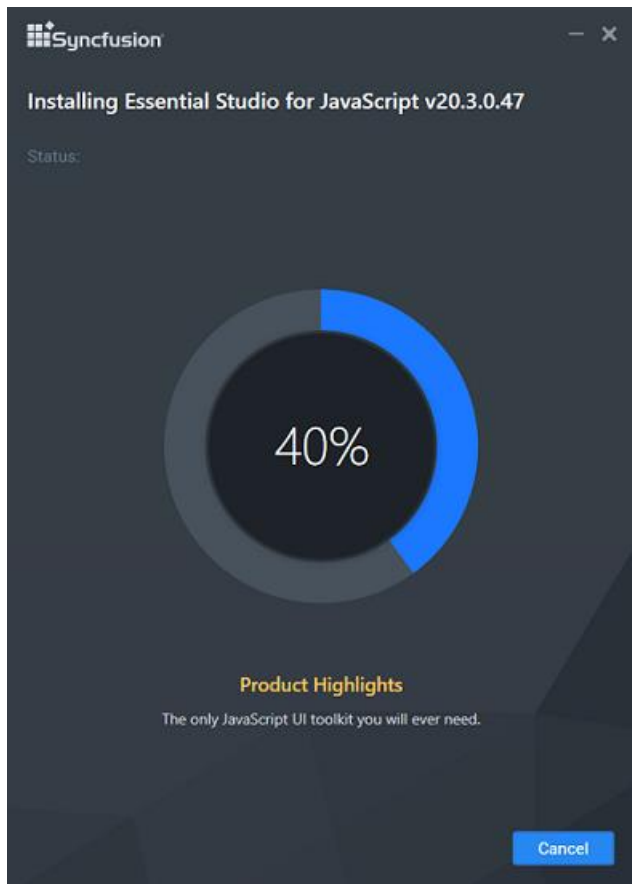
Note: From the 2021 Volume 1 release, Syncfusion has added the option to uninstall previous versions from 18.1 while installing the new version.

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.

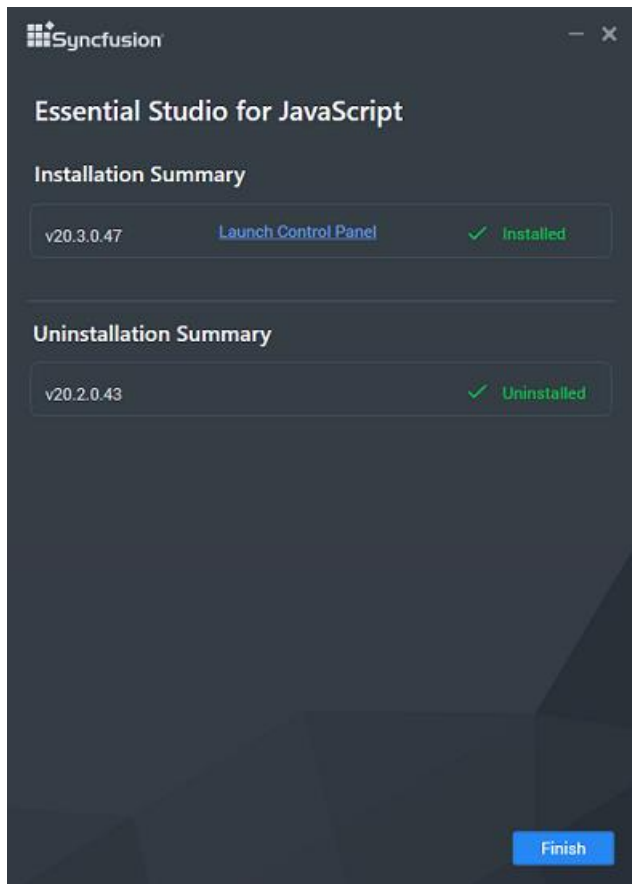
Confirmation Alert



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

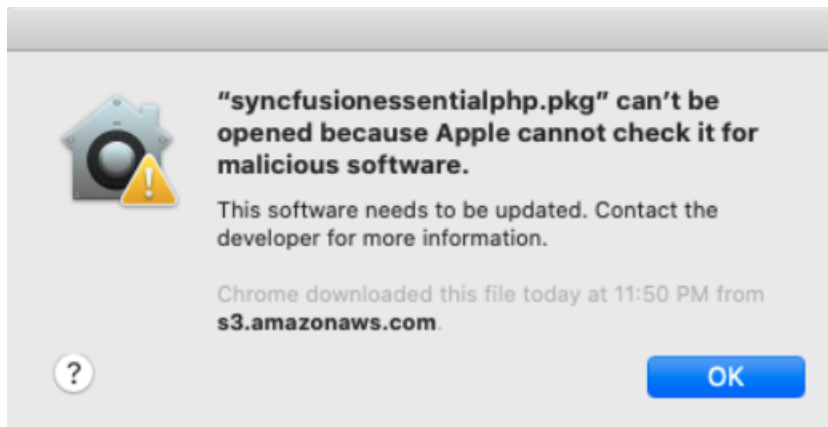
Example: “D:\Temp\syncfusionejs2_x.x.x.x.exe” /uninstall silent

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

Installing Syncfusion JavaScript – EJ2 Mac Installer

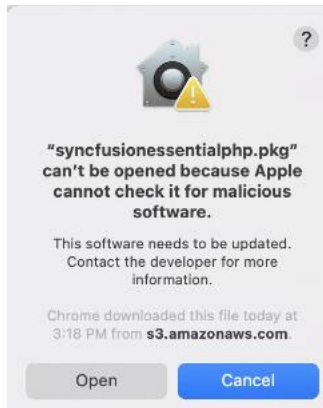
Steps to resolve the warning message in Catalina OS or later

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.

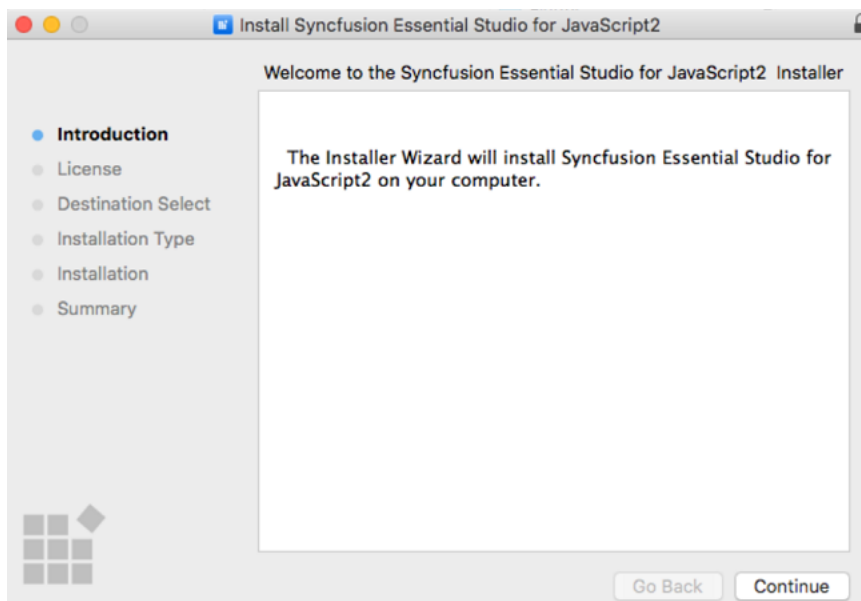


3. When you click "Open" the installer window will be opened.

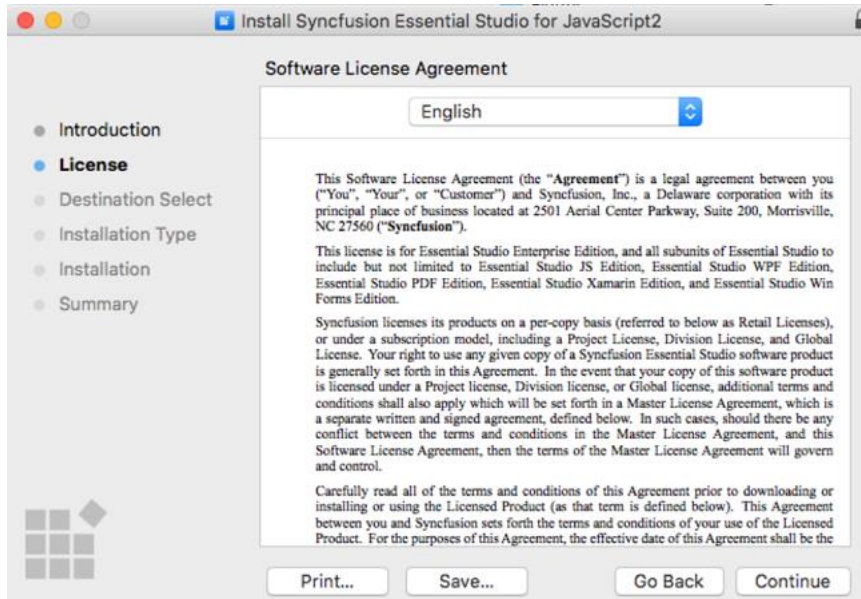
Step-by-Step Installation

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

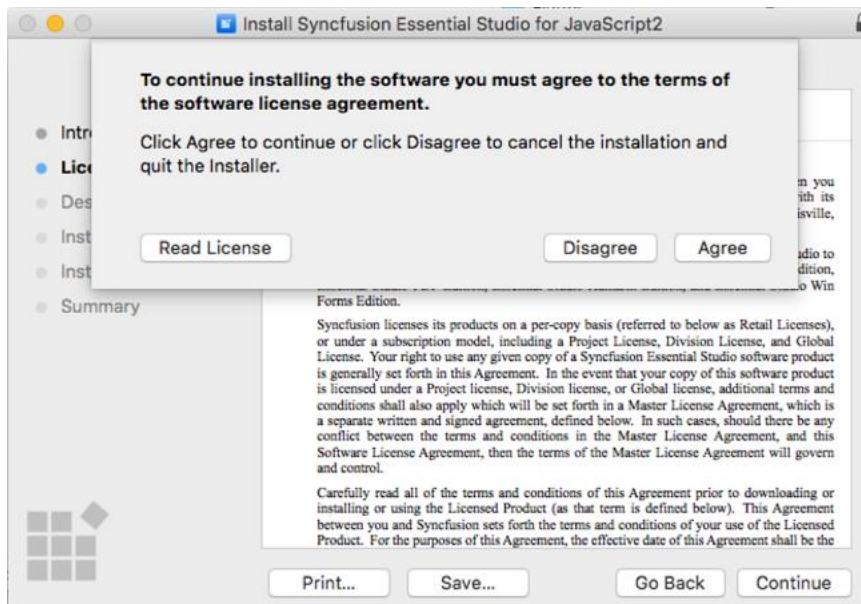
1. Double-click the Syncfusion Essential Studio JavaScript - EJ2 Mac installer(.pkg) file. The installer Wizard opens. Click Continue.



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

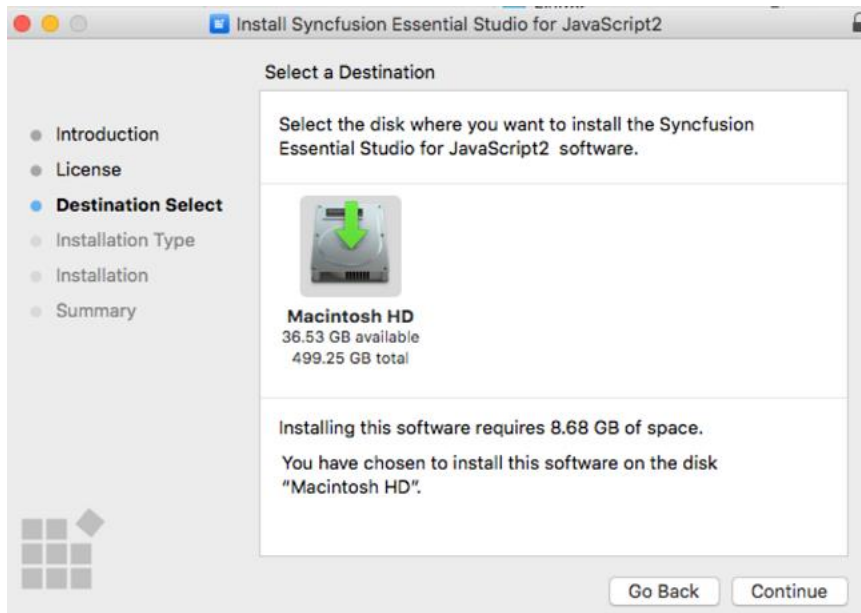


3. The License Agreement's Confirmation window will appear. If you have read the Software License Agreement, click **Agree**.

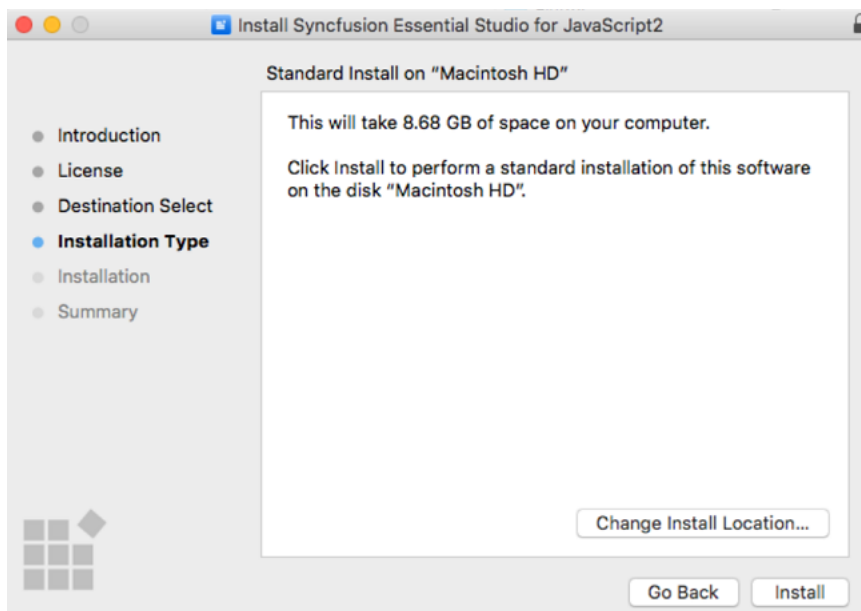


Note: The Unlock key is not required to install the Mac installer. The Syncfusion Essential Studio JavaScript - EJ2 Mac installer can be used for development purposes without registering the Unlock key.

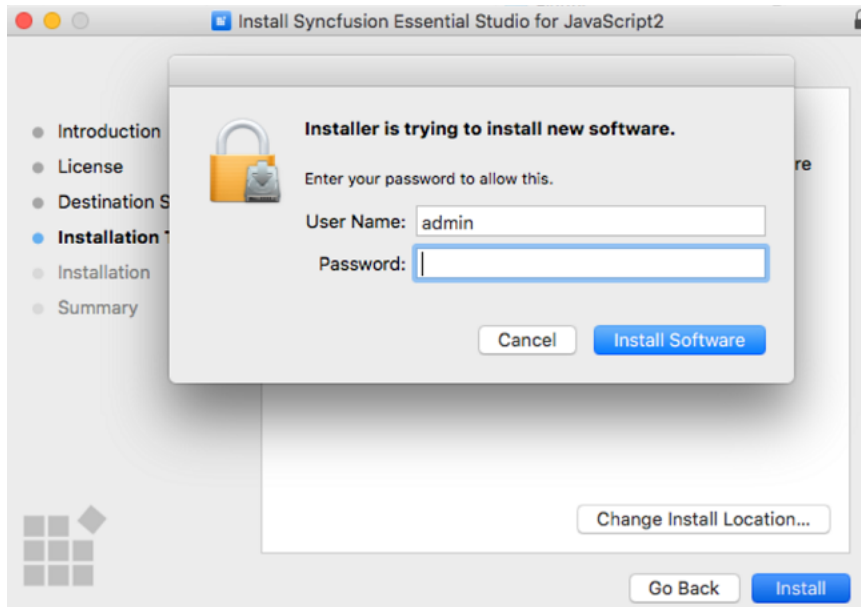
4. The Destination select wizard will appear. You can choose which disc to install the Syncfusion Essential Studio for JavaScript - EJ2 Mac installer on here.



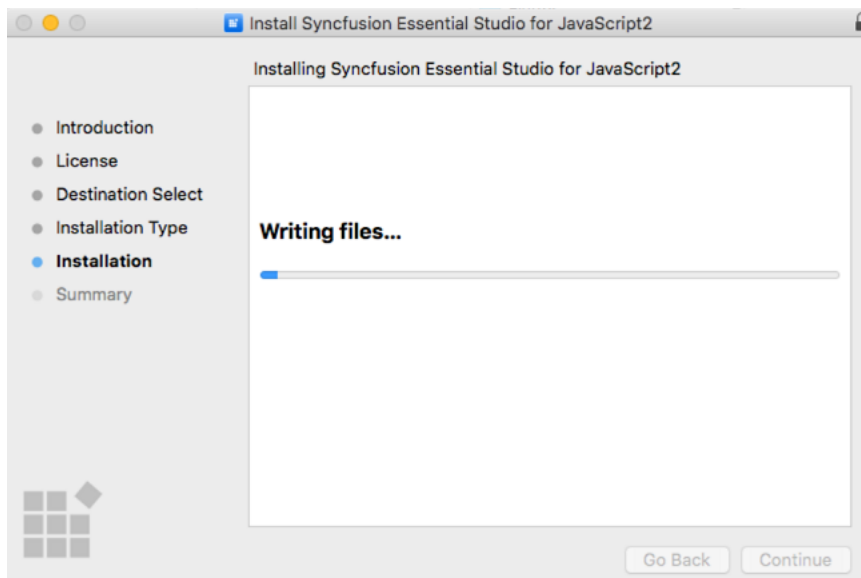
5. The Installation Type wizard will appear. Click Install to begin the standard installation of the Syncfusion Essential Studio JavaScript - EJ2 Mac installer.



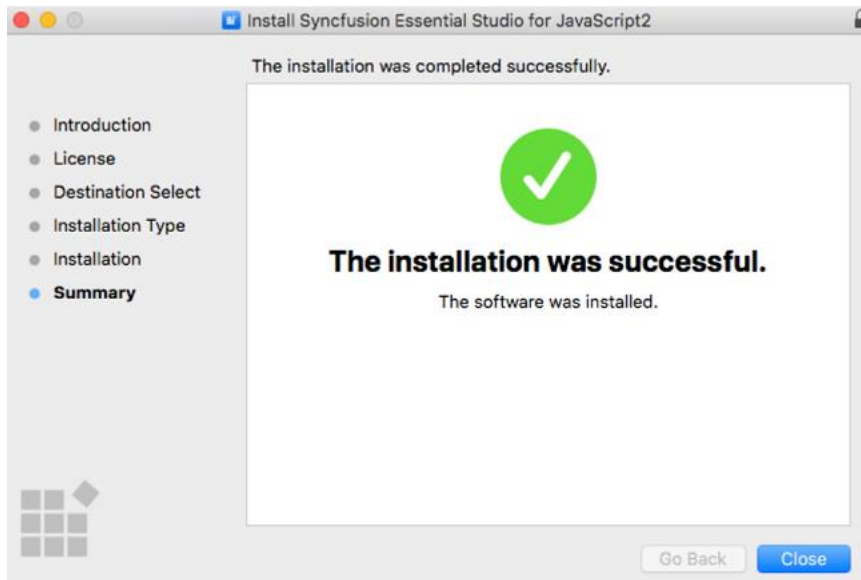
6. The Authentication window will appear. To begin the installation, enter the Mac machine's password and click **Install Software**.



7. The installation process will begin on your machine.

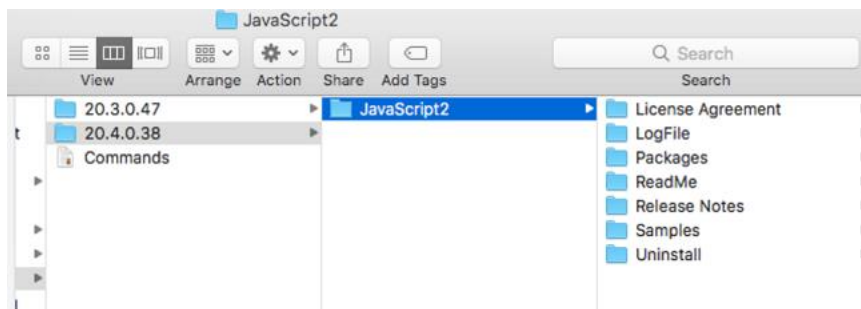


8. Once the installation is complete, the completed screen will be displayed. To exit the installation wizard, click Close.



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.

Register the license key by using [registerLicense](#) method after the Syncfusion JavaScript script file reference.

Linux Installer

Download Syncfusion JavaScript Linux Installer

The Syncfusion 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

You can download the Syncfusion installer from [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.)




Thank you for choosing to evaluate Essential Studio for JavaScript!

Please choose your preferred evaluation experience to proceed.

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

Full installation evaluation

Download and install the evaluation version of the full product along with all the required assemblies and demo code.

Choose Platform ☐  ☐  ☒ 


Choose Download Format ☐ Web Installer ☒ Offline Installer

DOWNLOAD

[More Download Options](#)

NuGet evaluation

No installation required. Get started in 2 minutes

 **Install assemblies from NuGet**

&

Download demo code from [GitHub](#) | Open demo code in [Visual Studio](#)

Version: 24.1.41, Released: November 28, 2023

[Release Notes](#) | [License Agreement](#) | [Installation Instructions](#) | [Security Management Report](#)

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.)

Trial Downloads and Unlock Keys

Essential Studio

Trial Version :

[What's New](#) | [Generate License File](#) | [Get Unlock Key](#)

Download ¹

More Download Options ²

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.

Downloads

Version : 24.1.41

Released on : Nov 28, 2023

Windows

Mac

Linux

[Read our Security Management Report](#) to learn about our security practices.

Essential Studio for JavaScript (Essential JS 2) Linux Offline Installer

JavaScript - EJ 2

Includes Angular, React, and Vue components.

[Release Notes](#) | [Readme](#)

.ZIP 0 MB

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

DESKTOP

WinForms
Includes 100+ enterprise-class WinForms controls for developing modern desktop applications.
[START TRIAL](#)

WPF
Includes 100+ enterprise-class WPF controls for developing modern desktop applications.
[START TRIAL](#)

WinUI
Syncfusion WinUI platform is a set of ever-growing UI controls for developing rich UWP apps.
[START TRIAL](#)

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.

4. 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](#) and [license key](#) here at any time before the trial period expires. (as shown in below screenshot.)



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



WEB - Essential JS 2		Offline Installer	
Blazor	Release Notes Readme		.ZIP 652 MB
			Checksum Value
ASP.NET Core - EJ 2	Release Notes Readme		.ZIP 1092 MB
			Checksum Value
JavaScript - EJ 2	Includes Angular, React, and Vue components. Release Notes Readme		.ZIP 2066 MB
			Checksum Value

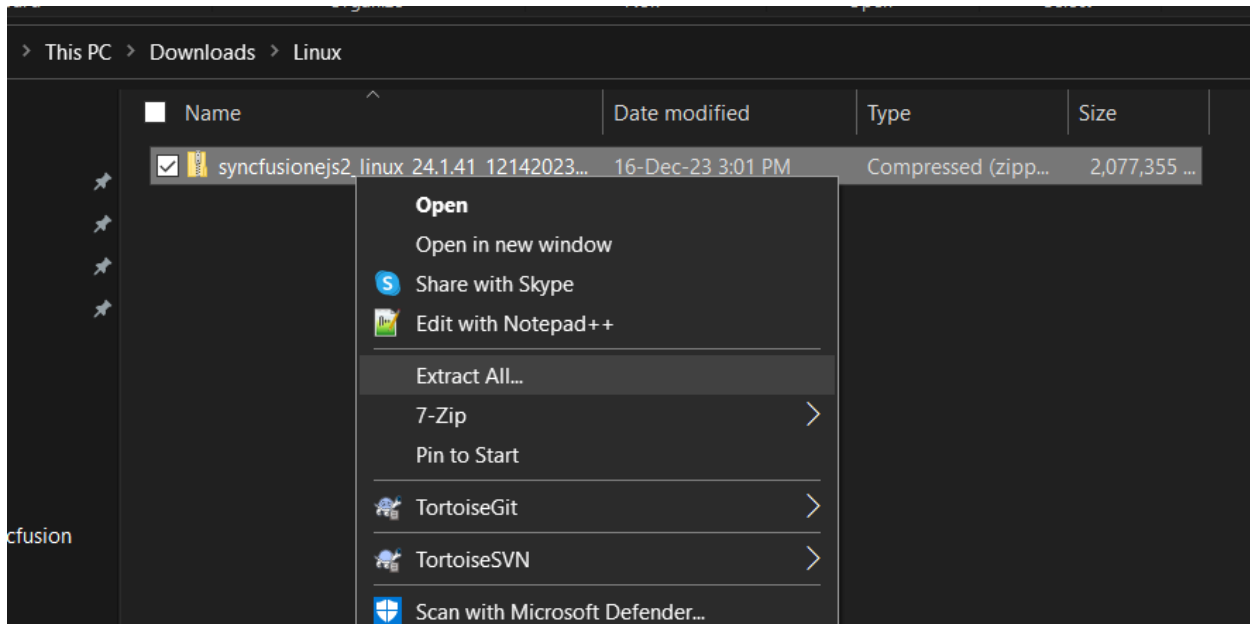
You can also refer to the [JavaScript Linux installer](#) links for step-by-step installation guidelines.

Installing Syncfusion JavaScript Linux installer

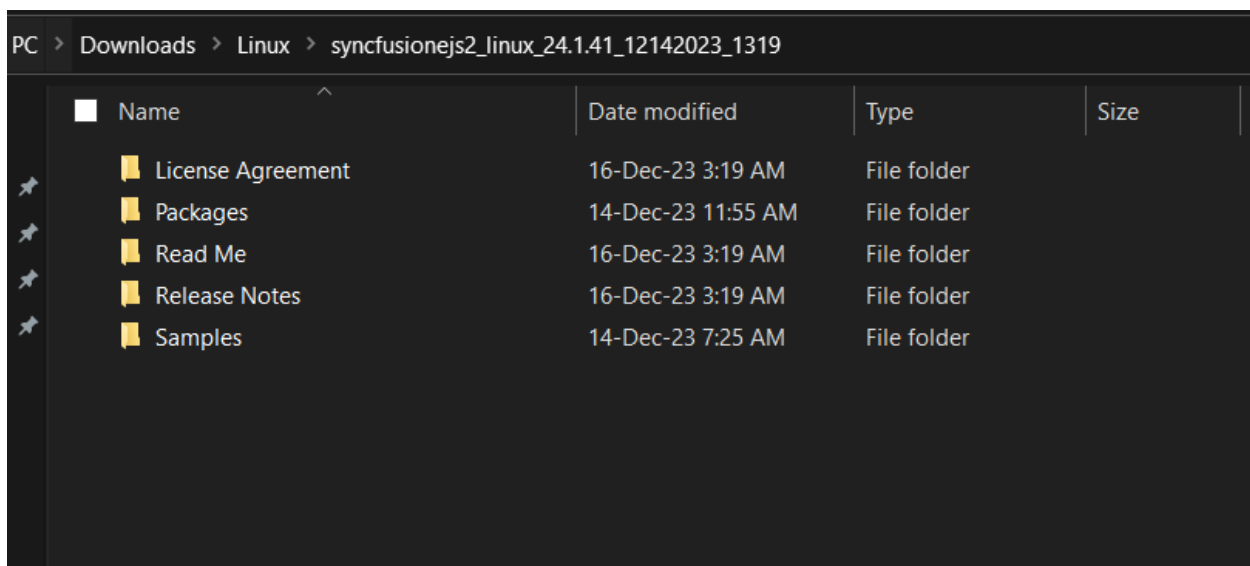
Step-by-Step Installation

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.

Register the license key by using [registerLicense](#) method after the Syncfusion JavaScript script file reference.

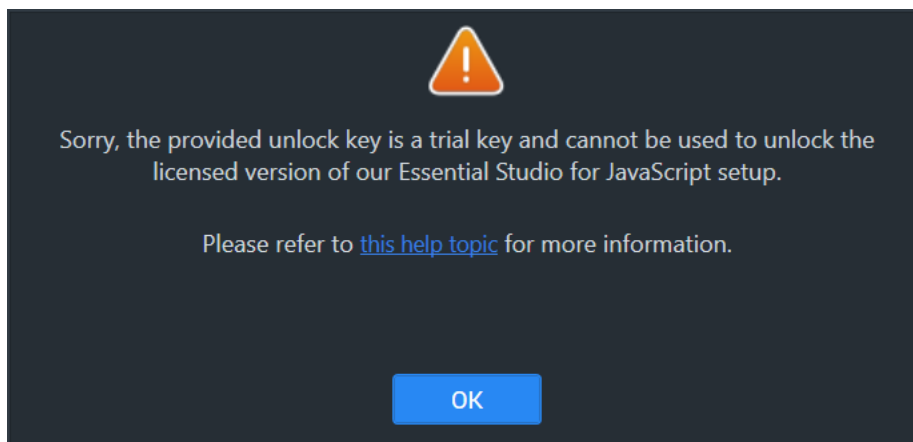
Common Installation Errors

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.



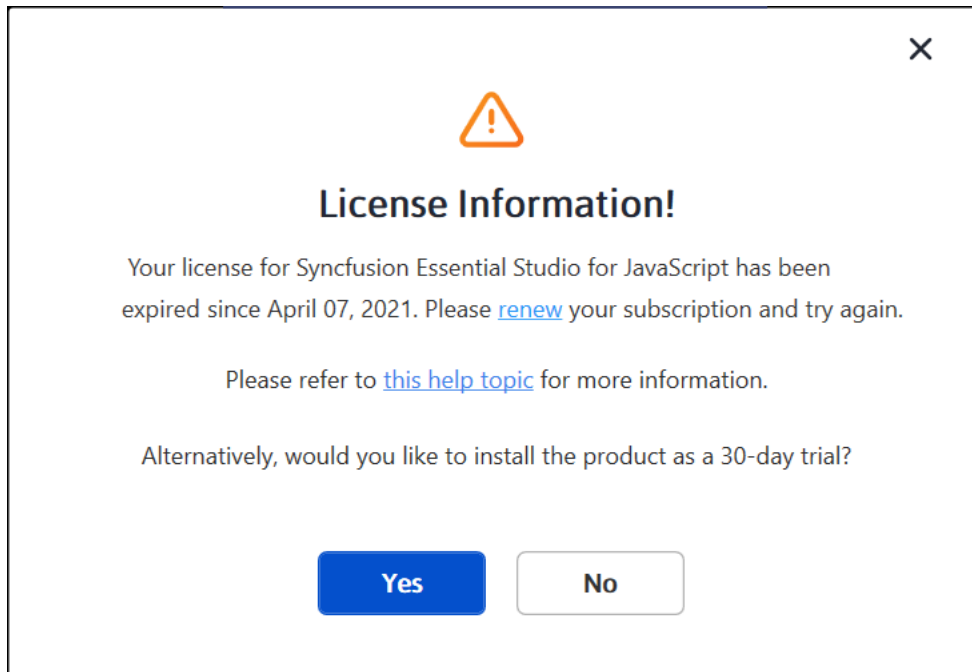
Reason
 You are attempting to use a Trial unlock key to unlock the licensed installer.

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.

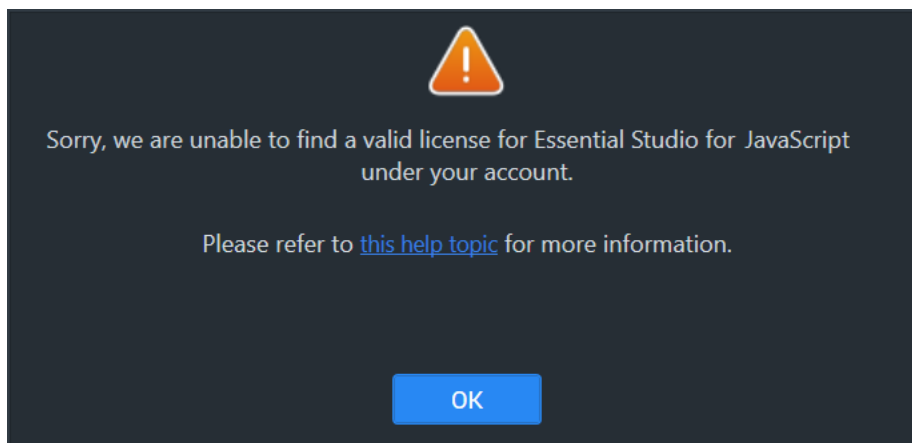
Suggested solution
 You can choose from the options below.

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.

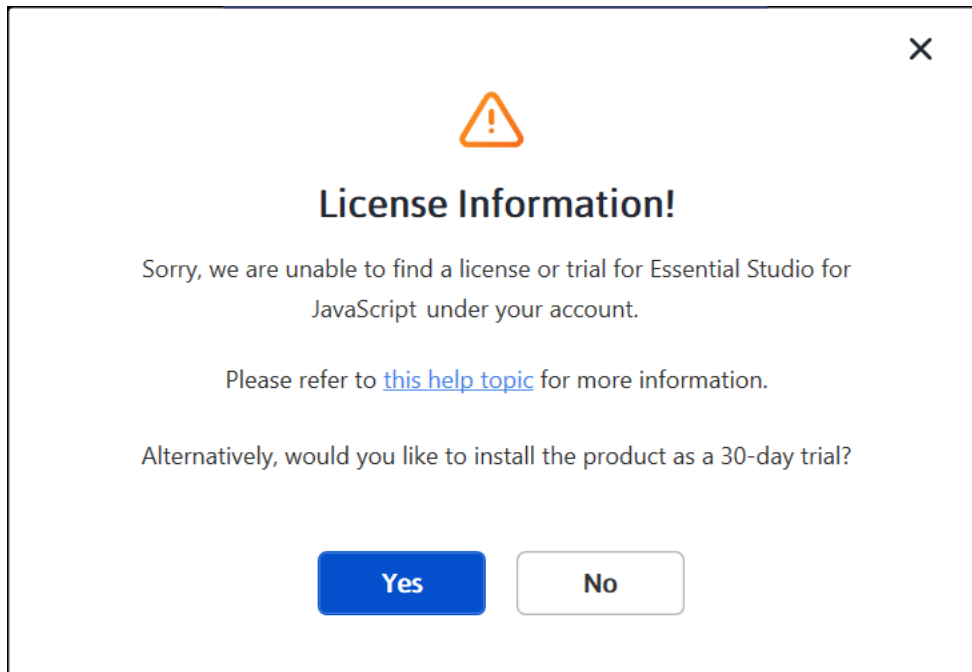
Unable to find a valid license or trial

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:

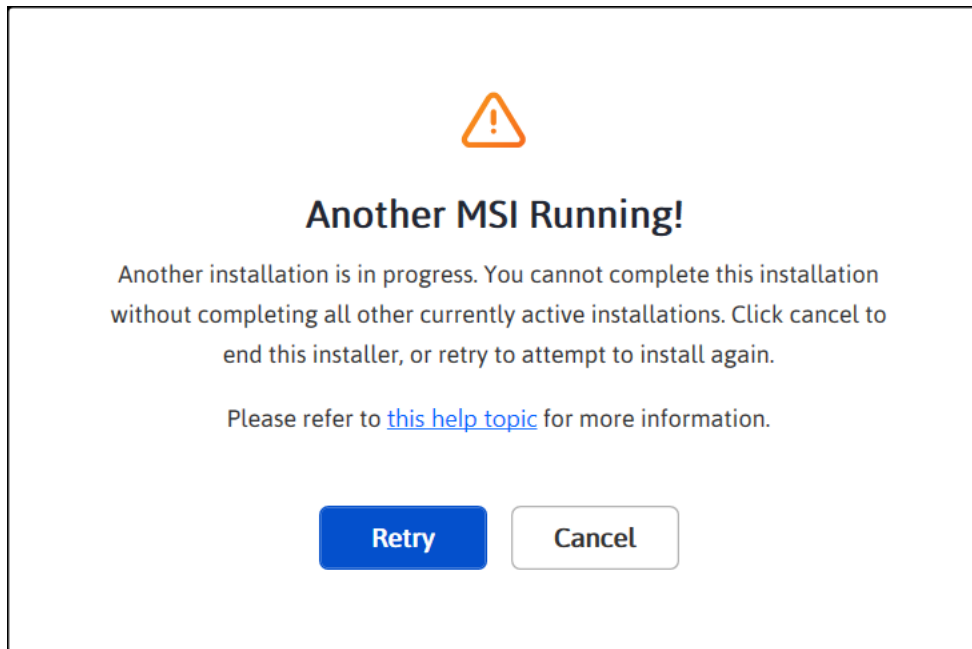
- When your trial period expired
- When you don't have a license or an active trial
- You are not the license holder of your license
- Your account administrator has not yet assigned you a license.

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

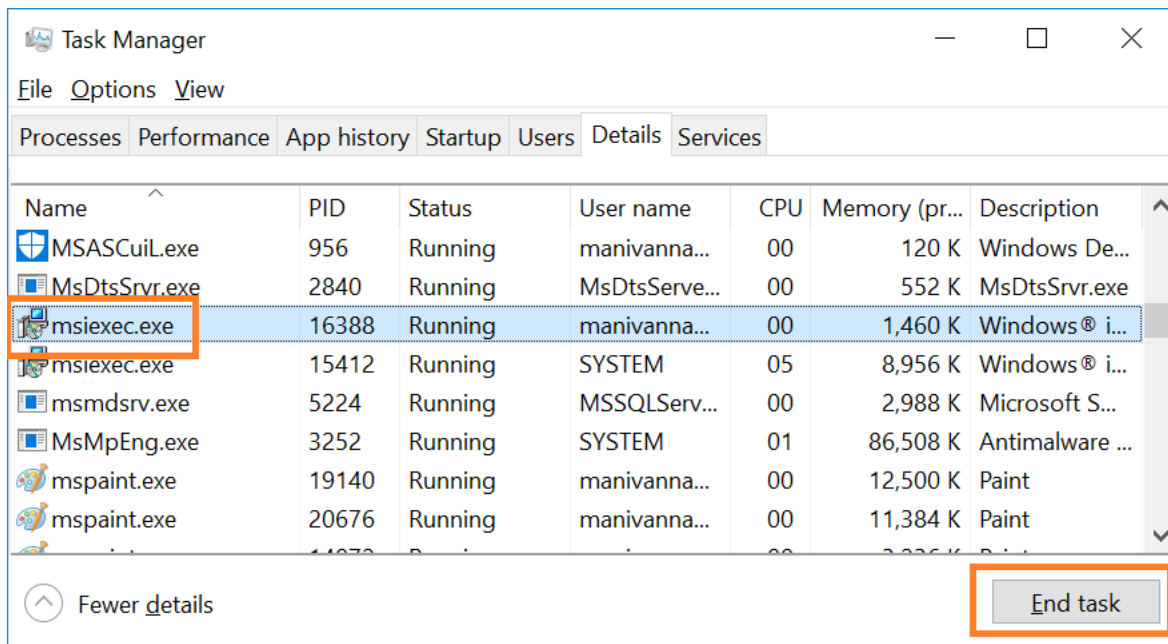
Error Message: Another installation is in progress. You cannot start this installation without completing all other currently active installations. Click cancel to end this installer or retry to attempt after currently active installation completed to install again.



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.

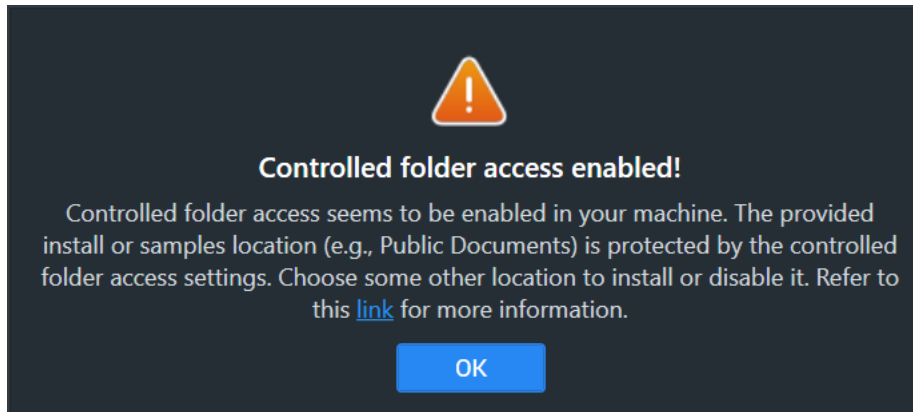
1. Open the Windows Task Manager.
2. Browse the Details tab.
3. Select the msixec.exe and click **End task**.



Unable to install due to controlled folder access

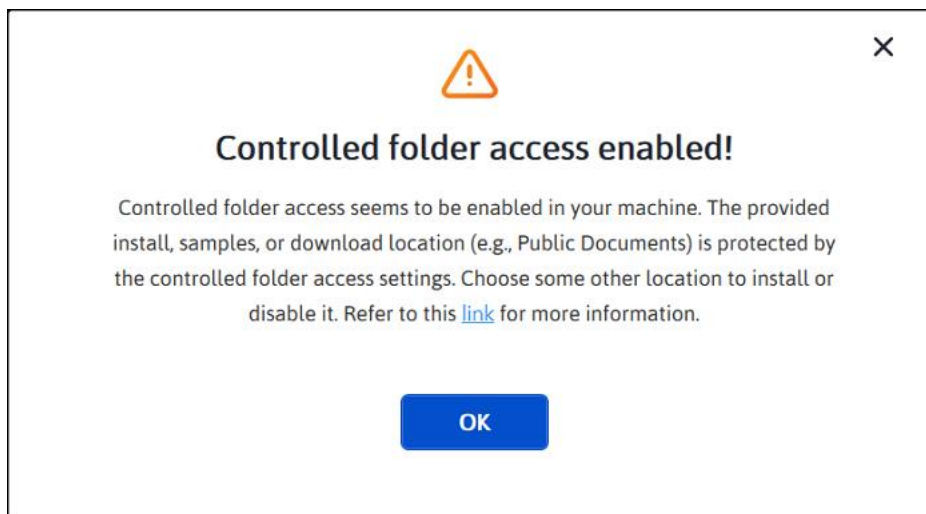
Offline

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)

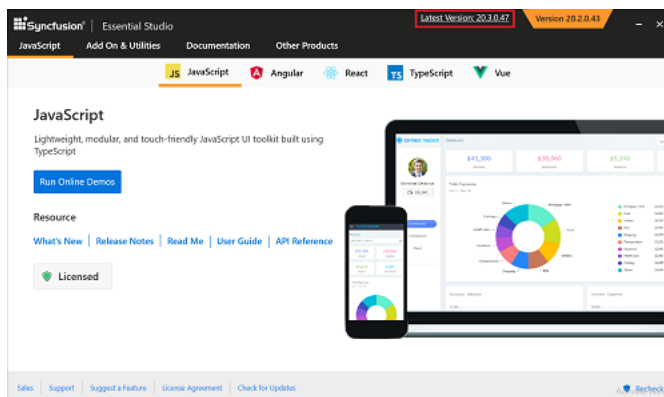
Syncfusion releases new volumes once every three months, with exciting new features. There will be one Service Pack release for these volume releases. Service Pack releases are provided to address major bug fixes in the volume releases.

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

See our "[Upgrade Guide](#)" for JavaScript – EJ2 to learn more about the “Breaking Changes, Bug Fixes, Features and Known Issues” between your current version and the latest version you are trying to upgrade.

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

Overview in EJ2 JavaScript Licensing control

We have introduced license key validation for Essential JS2 platforms from the 2022 Volume 1 release. This licensing key validation will enforce the developer to register the valid licensing key in an application while referring to any of the latest JavaScript packages, either from npm or CDN or build.

License key can be obtained from the [My Account >> License and downloads](#) section of the Syncfusion website. To obtain a license key, you will need to have an active trial or license or community license.

Before using any JavaScript controls, you must register the obtained license key in the application code. Otherwise you will get license validation error message in application as shown in below

This application was built using a trial version of Syncfusion Essential Studio. Please include a valid license to permanently remove this license validation message. You can also obtain a free 30 day evaluation license to temporarily remove this message during the evaluation period. Please refer to this [help topic](#) for more information.

Difference between unlock key and license key

Please note that this license key is different from the installer unlock key that you might have used in the past and needs to be separately generated from Syncfusion website.

- **Unlock Key** - Syncfusion Unlock Key is used to unlock the Syncfusion installers alone.
- **License Key** - Syncfusion License Key is a string that needs to be registered in your script to avoid licensing warning.

Refer to [this](#) KB article to know more about difference between the Syncfusion Unlock Key and the Syncfusion License Key.

Registering Syncfusion license keys in Build server

| Source of Syncfusion assemblies | Details | License Key needs to be registered? | Where to get license key from |

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

| **NuGet package** | If the Syncfusion assemblies used in Build Server were from the Syncfusion NuGet packages, then no need to install any Syncfusion installer. We can directly use the required Syncfusion NuGet packages at [nuget.org](#).

But, if using NuGet packages from the [nuget.org](#), then we should register the Syncfusion license key in the application. | Yes | Use any developer license to [generate](#) keys for Build Environments as well. |

| **Trial installer** | If the Syncfusion assemblies used in Build Server were from Trial Installer, we should register the license key in the application for the corresponding version and platforms, to avoid trial license warning. | Yes | Use any developer trial license to [generate](#) keys for Build Environments as well. |

| **Licensed installer** | If the Syncfusion assemblies used in Build Server were from Licensed Installer, then there is no need to register the license keys.

You can [download](#) and [install](#) the licensed version of our installer. | No | Not applicable |

See Also

- [How to generate Syncfusion JavaScript \(ES5\) license key?](#)
- [How to register Syncfusion license key in JavaScript \(ES5\) application?](#)
- [Licensing FAQ](#)

License key generation in EJ2 JavaScript Licensing control

License keys can be generated from the [License & Downloads](#) or [Trial & Downloads](#) section of the Syncfusion website.

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio[Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)

[Download](#)

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Code Scan Report](#)

[More Download Options](#)

* Syncfusion license keys are **version and platform specific**. Refer to the [KB](#) to generate the license key for the required version and platform.

* Refer to this [KB](#) to know which version of the Syncfusion license key should be used in the application.

Claim License key

Syncfusion License keys can also be generated from the “**Claim License Key**” page based on the trial or valid license associated with your Syncfusion account.

You can get the license key, based on license availability in your Syncfusion account.

Active License

If you have a Syncfusion account associated with valid license, license key will be generated from claim license key page.

Claim License Key

Essential Studio Enterprise Edition **v20.3.0.56**

Different Platform or Version? [Get License Key](#)

View full key

[COPY TO CLIPBOARD](#) [SEND EMAIL](#)

Note: A license key cannot be generated for our Java platform. If you are looking for a Java license key, please [click here](#).

Please refer to this [help topic](#) to learn how to register your license key.

Active Trial

If you have a Syncfusion account associated with valid trial license, license key will be generated from claim license key page with expiry date.

Claim License Key

Essential Studio Enterprise Edition **Trial v20.3.0.56**

View full key

[COPY TO CLIPBOARD](#) [SEND EMAIL](#)

Note: A license key cannot be generated for our Java platform. If you are looking for a Java license key, please [click here](#).

Your license key expires on January 20, 2023.

Please refer to this [help topic](#) to learn how to register your license key.

If you are looking for valid licenses, try one of the below options.

Purchase Essential Studio

The world's best UI component suite for building powerful web, desktop, and mobile apps.

[BUY NOW](#)

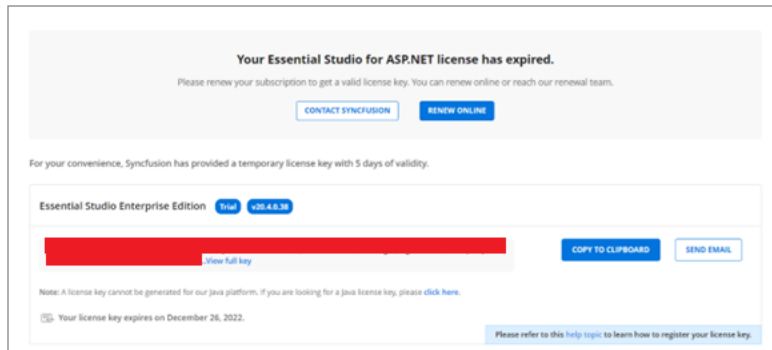
Free Community License

Eligibility: Companies and individuals with less than \$1 million USD in annual gross revenue and 5 or fewer developers.

[CLAIM YOUR FREE LICENSE](#)

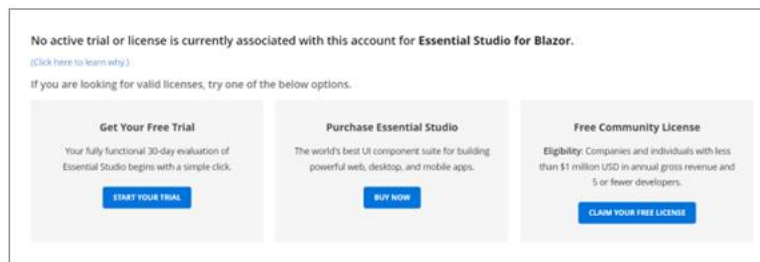
Expired License

If you have a Syncfusion account with an expired license, your license subscription must be renewed in order to obtain a valid license key for the latest Essential Studio version. Meanwhile, a temporary license key with a five day validity period will be generated.



No Trial or No License or Expired trial

If the Syncfusion account is not associated with a trial, license, or expired trial, you can try to claim either a trial or a valid license from claim license page.



See Also

- [How to register Syncfusion license key in the application?](#)
- [Licensing FAQ](#)

License key registration in EJ2 JavaScript Licensing control

Syncfusion license key should be registered, if your project using Syncfusion JavaScript (ES5) packages reference. The generated license key is a string that needs to be registered after any [Syncfusion JavaScript \(ES5\) reference](#).

Note: Syncfusion license validation is done offline during application execution and does not require internet access. Apps registered with a Syncfusion license key can be deployed on any system that does not have an internet connection.

The following code is used to register the license.

Javascript es5

Register the license key by using 'registerLicense' method after the [Syncfusion JavaScript script](#) file reference as below.

Note: As we have mandated License registration for the Syncfusion JavaScript components, we realize there is a possibility to view the registered Syncfusion License key from your application by others. We

acknowledge this is taking place in the JavaScript platform, as we have minimal scope for hiding the License key and we also recommend not to advertise it.

```
`ts
```

```
// Registering Syncfusion license key
```

```
ej.base.registerLicense('Replace your generated license key here');
```

```
,
```

Note: Only from 2022 Vol 1 v20.1.0.47, license key registration required for Essential JavaScript 2 products.

See also

- [Licensing FAQ](#)

Licensing errors in EJ2 JavaScript Licensing control

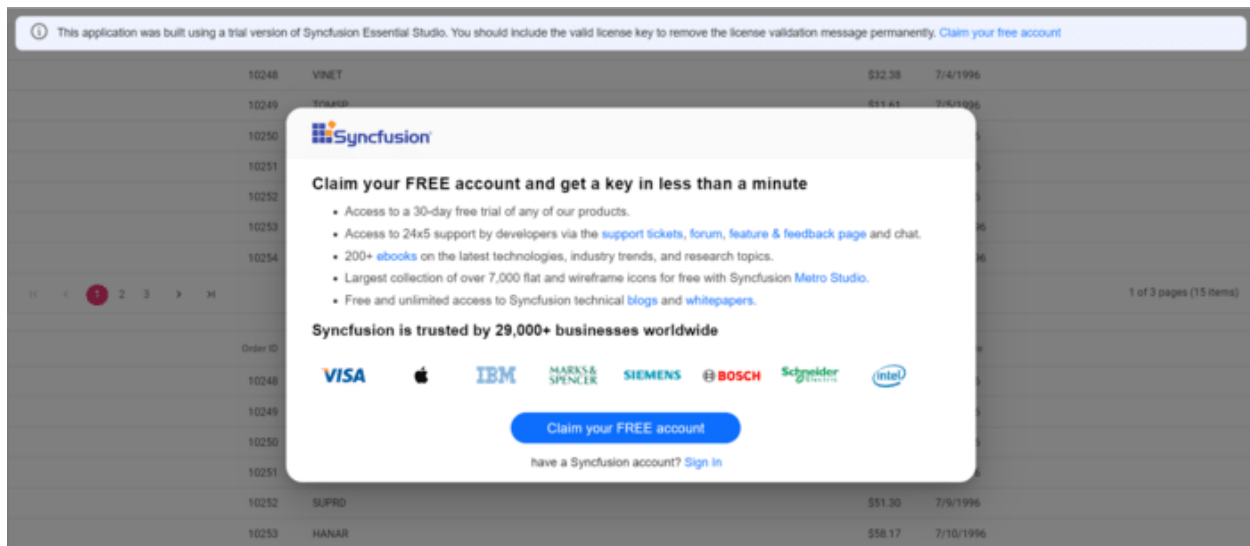
Licensing error popup is displayed with various messages under different circumstances. Here are some ways to resolve different issues.

Licensing errors

License key not registered\Trial Expired

The following error message will be shown if a Syncfusion license key has not been registered in your application or if the trial key has expired after 30 days.

Error message :
 This application was built using a trial version of Syncfusion Essential Studio. You should include the valid license key to remove the license validation message permanently.



Solution:

- If you use JavaScript(ES5) components through syncfusion installer, you can choose from the options listed below
 1. If you **have a valid Syncfusion license**, you can **generate a license key for a specific version and product** from [this page](#).

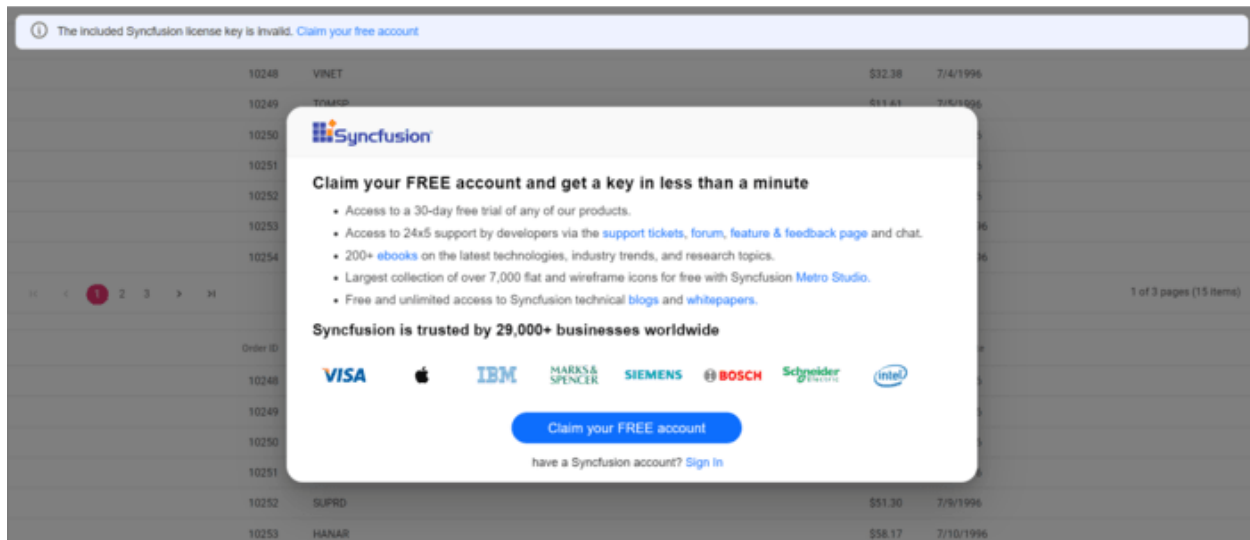
Essential Studio Enterprise Edition - Community license	
Essential Studio Enterprise Edition Binary with TestStudio	Download Older Versions
Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)	
What's New Release Notes Get License Key Get Unlock Key Code Scan Report	Download More Download Options

- If you **have a Syncfusion account and an active trial**, you can **generate the trial license key for a specific version and platform** from [this page](#).
- If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can generate the trial license key for a **specific version and platform** from [this page](#).
- If you **do not already have a Syncfusion account**, you can create one [here](#) and [purchase a license](#) or start your 30-day free trial. Then you can **generate the trial license key for a specific version and platform** from [this page](#).
- Also, you can generate the license key from claim license key page by clicking the “**Claim your FREE account**” click from the licensing warning message. Refer to this [help topic](#) for more details.
 - In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Invalid key

If the application is registered with an invalid key, another version of license key, or another platform's license key, the following error message will pop up when launching the application.

Error Message:
 The included Syncfusion license key is invalid.



Solution:

- If you use JavaScript(ES5) components through syncfusion installer, you can choose from the options listed below

1. If you have a valid Syncfusion license, you can **generate a license key for a specific version and product** from [this page](#).

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)
 [Download](#)

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Code Scan Report](#)
[More Download Options](#)

2. If you have a Syncfusion account and an active trial, you can **generate the trial license key for a specific version and product** from [this page](#).
3. If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
4. If you **do not already have a Syncfusion account**, you can create one here and [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
5. Also, you can generate the license key from claim license key page by clicking the **“Claim your FREE account”** click from the licensing warning message. Refer to this [help topic](#) for more details.
 - o In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Licensing errors from version 16.2.0* to 20.3.0*

License key not registered

The following error message will be shown if a Syncfusion license key has not been registered in your application.

Error message:
 This application was built using a trial version of Syncfusion Essential Studio. Please include a valid license to permanently remove this license validation message. You can also obtain a free 30 day evaluation license to temporarily remove this message during the evaluation period. Please refer to this [help topic](#) for more information.

This application was built using a trial version of Syncfusion Essential Studio. Please include a valid license to permanently remove this license validation message. You can also obtain a free 30 day evaluation license to temporarily remove this message during the evaluation period. Please refer to this [help topic](#) for more information.

10248	Buyer/ 10248	32.38
10249	Buyer/ 10249	11.61
10250	Buyer/ 10250	65.83
10251	Buyer/ 10251	41.34
10252	Buyer/ 10252	51.3
10253	Buyer/ 10253	58.17

Solution:

- If you use JavaScript (ES5) components through syncfusion installer, you can choose from the options listed below
 1. If you **have a valid Syncfusion license**, you can **generate a license key for a specific version and product** from [this page](#).

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)
 [Download](#)

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Code Scan Report](#)
[More Download Options](#)

- If you **have a Syncfusion account and an active trial**, you can **generate the trial license key for a specific version and platform** from [this page](#).
- If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can generate the trial license key for a **specific version and platform** from [this page](#).
- If you **do not already have a Syncfusion account**, you can create one [here](#) and [purchase a license](#) or start your 30-day free trial. Then you can **generate the trial license key for a specific version and platform** from [this page](#).
 - In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Invalid key

If the application is registered with an invalid key, another version of license key, or another platform's license key, the following error message will pop up when launching the application.

Error message:
 The included Syncfusion license is invalid. Please refer to this [help topic](#) for more information.

The included Syncfusion license is invalid. Please refer to this help topic for more information.		
Order ID	Customer Name	Freight
10248	Buyer/ 10248	32.38
10249	Buyer/ 10249	11.61
10250	Buyer/ 10250	65.83
10251	Buyer/ 10251	41.34
10252	Buyer/ 10252	51.3
10253	Buyer/ 10253	58.17

Solution:

- If you use JavaScript (ES5) components through syncfusion installer, you can choose from the options listed below
 - If you have a valid Syncfusion license, you can **generate a license key for a specific version and product** from [this page](#).

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)
 [Download](#)

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Code Scan Report](#)
[More Download Options](#)

2. If you have a Syncfusion account and an active trial, you can **generate the trial license key for a specific version and product** from [this page](#).
3. If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
4. If you **do not already have a Syncfusion account**, you can create one here and [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
 - In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Trial expired

The following error message will be shown if the trial key has expired after 30 days.

Error message:
 Your Syncfusion trial license has expired. Please refer to this [help topic](#) for more information.

Your Syncfusion trial license has expired. Please refer to this help topic for more information.		
Order ID	Customer Name	Freight
10248	Buyer/ 10248	32.38
10249	Buyer/ 10249	11.61
10250	Buyer/ 10250	65.83
10251	Buyer/ 10251	41.34
10252	Buyer/ 10252	51.3
10253	Buyer/ 10253	58.17

Solution:
 Purchase from [here](#) to get a valid Syncfusion license.

Platform Mismatch

If the application is registered with another platform's license key, the following error message will pop up when launching the application.

Error message:
 The included Syncfusion license is invalid (Platform mismatch). Please refer to this [help topic](#) for more information.

The included Syncfusion license is invalid (Platform mismatch). Please refer to this help topic for more information.

Solution:

- License keys are version and product specific. So, if you use JavaScript (ES5) components through syncfusion installer, you can choose from the options listed below
 1. If you have a valid Syncfusion license, you can **generate a license key for a specific version and product** from [this page](#).

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)
 [Download](#)

[What's New](#) | [Release Notes](#) | [Get License Key](#) | [Get Unlock Key](#) | [Code Scan Report](#)
[More Download Options](#)

2. If you have a Syncfusion account and an active trial, you can **generate the trial license key for a specific version and product** from [this page](#).
3. If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
 - o In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Version mismatch

If the application is registered with another version's license key, the following error message will pop up when launching the application.

Error message:
 The included Syncfusion license ({Registered Version}) is invalid for version {Required version}. Please refer to this [help topic](#) for more information.

The included Syncfusion license (v19.1.0.44) is invalid for version 20.1.x. Please refer to this help topic for more information.		
Order ID	Customer Name	Freight
10248	Buyer/ 10248	32.38
10249	Buyer/ 10249	11.61
10250	Buyer/ 10250	65.83
10251	Buyer/ 10251	41.34
10252	Buyer/ 10252	51.3

Solution:

- License keys are version and product specific. So, if you use JavaScript (ES5) components through syncfusion installer, you can choose from the options listed below
 1. If you have a valid Syncfusion license, you can **generate a license key for a specific version and product** from [this page](#).

Essential Studio Enterprise Edition - Community license

Essential Studio Enterprise Edition Binary with TestStudio [Download Older Versions](#)

Latest Official Release : 20.3.0.47 (Volume 3 2022 - September 29, 2022)

[What's New](#) | [Release Notes](#) | [Get License Key](#) ⓘ | [Get Unlock Key](#) ⓘ | [Code Scan Report](#)

[Download](#)

[More Download Options](#)

2. If you have a Syncfusion account and an active trial, you can **generate the trial license key for a specific version and product** from [this page](#).
3. If you **have a Syncfusion account but no active trials**, [purchase a license](#) or [start your 30-day free trial](#). Then you can **generate the trial license key for a specific version and product** from [this page](#).
 - o In your application, register the generated license key. Please refer to this [help topic](#) for information on registering the license key.

Licensing troubleshoot in EJ2 JavaScript Licensing control

Is an internet connection required for license validation

No, Internet connection is not required for the Syncfusion Essential Studio license validation. The Syncfusion license validation is done offline during application execution. Apps registered with a Syncfusion license key can be deployed on any system that does not have an internet connection.

Upgrade from the trial version after purchasing a license

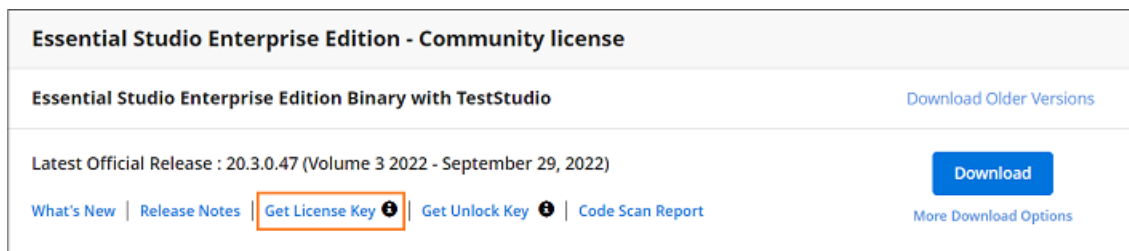
To upgrade from the trial version, there are two possible solutions:

- Uninstall the trial version and install the fully licensed build from the [License & Downloads](#) section of the Syncfusion website.
- If you are using Syncfusion controls from the [npm](#), replace the currently used trial license key with a paid license key that can be generated from the [License & Downloads](#) section of Syncfusion website. Refer to [this](#) topic for more information regarding registering the license in the application.

The license registration is not required if you reference Syncfusion scripts from the Licensed installer. These licensing changes apply to all evaluators who refer to the Syncfusion scripts from the evaluation installer and those who use the Syncfusion NuGet packages form [nuget.org](#).

Where can I get a license key

License keys can be generated from the [License & Downloads](#) or the [Trial & Downloads](#) section of the Syncfusion website.



The Syncfusion license keys are the **version and platform-specific**, refer to the [KB](#) to generate the license key for the required version and platform. Also, refer to this [KB](#) to know which version of the Syncfusion license key should be used in the application.

While using the ASP.NET Core controls with the Javascript(ES5) components, you need to register the license key in both the Javascript(ES5) and the [ASP.NET core](#). Since the license is validated at the client side for Javascript(ES5) components and server-side for the ASP.NET core components.

Refer EJ2 scripts without registering the license key

Registering the Syncfusion license key in the application is mandated from version 20.1 for all the Syncfusion EJ2 platforms to avoid licensing warnings. The end-users can easily inspect the License key registered in the script due to the nature of the JavaScript. If you are a licensed customer and don't want to expose your key due to security reasons or other reasons, follow the following steps to ej2 scripts without registering the license key in the application.

Using scripts from the licensed installer

License registration is not required for licensed users if referring to the script from a [licensed installer](#). The following steps will guide you to get the installed script from your machine.

- After installing the licensed build from a licensed installer, go to the installed location and navigate to the **EJ2/Installed Version/Web (Essential JS 2)/JavaScript/ej2-js-es5**.

Local Disk (C:) > Program Files (x86) > Syncfusion > Essential Studio > JavaScript - EJ2 > 20.1.0.55 > Web (Essential JS 2) > JavaScript

<input type="checkbox"/> Name	Date modified	Type	Size
ej2-heatmap	5/12/2022 4:30 PM	File folder	
ej2-inplace-editor	5/12/2022 4:30 PM	File folder	
ej2-inputs	5/12/2022 4:30 PM	File folder	
<input checked="" type="checkbox"/> ej2-js-es5	5/12/2022 4:30 PM	File folder	
ej2-kanban	5/12/2022 4:30 PM	File folder	
ej2-layouts	5/12/2022 4:30 PM	File folder	

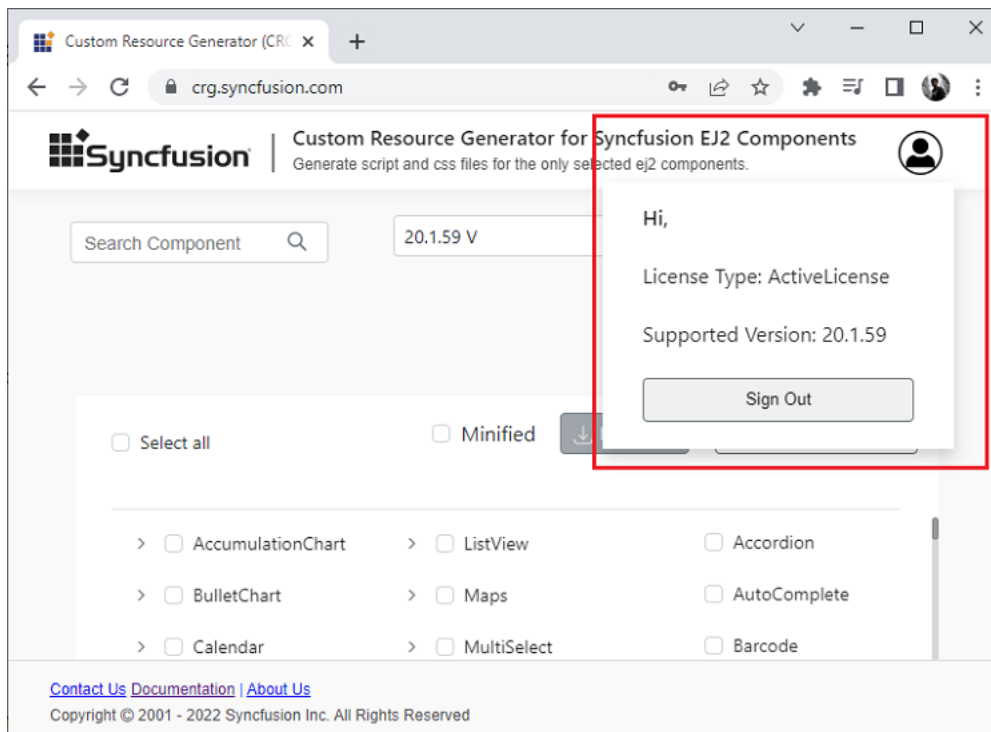
- Navigate to the ej2-js-es5 package folder and access the **ej2.min.js** from the **scripts** folder and utilize the minified ej2 script in the application.

Disk (C:) > Program Files (x86) > Syncfusion > Essential Studio > JavaScript - EJ2 > 20.1.0.55 > Web (Essential JS 2) > JavaScript > ej2-js-es5 > scripts

<input type="checkbox"/> Name	Date modified	Type	Size
ej2.d.ts	10/26/1985 1:45 PM	TS File	4 KB
ej2.min.js	10/26/1985 1:45 PM	JavaScript File	16,105 KB

Using scripts from the CRG

If you are a licensed customer, you can download scripts from the [CRG](#) which doesn't require the license registration by signing in with the Syncfusion account. You can check your license status by clicking my account icon as follows.



Refer to the [CRG documentation](#) to know more about how to refer to scripts.

Will the registered license key expire

No, the Syncfusion license keys won't expire for a particular version and you can continue to use it. So, you won't face any problems on the live site. If you have used the trial key, it will expire in 30 days and we don't recommend using it in production.

If you upgrade to newer versions of the Syncfusion packages, you have to generate new license keys and use them.

When to generate new license key while upgrading

You don't have to generate and change license keys for minor version upgrades. If you upgrade from one major version and another major version, you have to generate new license keys and register in the application.

For example,

- If you upgrade to weekly releases or the SP release in the same major version, you don't have to change license such as if you upgrade from the 20.1.47 version to 20.1.* you don't have to change the license keys.
- If you upgrade from one major version to another major version, you have to generate new license keys for the latest version and change in the application, such as if you upgrade from the 20.1. version to 20.2., you have to generate new license keys for the latest version and change in the application.

License registration for multiple developers on your project

Syncfusion license key is a version based and it's not based on the developer. You don't have to register different keys for each developer. Just register one valid license key when developing and publishing the software.

Can I use the same key for all the web apps under the project

Yes, you can use the same license key for all the web apps.

Does the license registration access any resources or data

No, the license registration doesn't access any data or resources.

License & Downloads shows the "Essential Studio Enterprise Edition Binary with Test Studio" and the "Project License". Which license to use

Use any licenses shown on the [accounts & downloads](#) page. It shows two licenses because if you are part of your company's enterprise portal Global license and an individual license is also assigned to your account, on your account & downloads page, the individual license and your enterprise portal Global license are shown.

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.1.0.47 (Volume 1 2022 - April 04, 2022)

[What's New](#) |
 [Release Notes](#) |
 [Get License Key](#) |
 [Get Unlock Key](#) |
 [Code Scan Report](#)

Your license expires on June 30, 2023.

[Download](#)

[More Download Options](#)

Project License

Essential Studio Enterprise Edition Binary with TestStudio
[Download Older Versions](#)

Latest Official Release : 20.1.0.47 (Volume 1 2022 - April 04, 2022)

[What's New](#) |
 [Release Notes](#) |
 [Get License Key](#) |
 [Get Unlock Key](#) |
 [Code Scan Report](#)

Your license expires on June 30, 2023.

[Download](#)

[More Download Options](#)

Refer to the [KB](#) article which explains the Licenses offered by Syncfusion.

Appearance

Theme in EJ2 JavaScript 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/<themenam>.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/{DEPENDENCYPACKAGENAME}/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)
--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)

Name	Value (Default Theme)	Value (Dark Theme)
--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)
--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)

Name	Value (Default Theme)	Value (Dark Theme)
--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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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

Name	Value
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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%)

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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

Name	Value
\$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
\$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

Name	Value
\$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
\$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

Name	Value (Default Theme)	Value (Dark Theme)
\$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
\$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	

Name	Value (Default Theme)	Value (Dark Theme)
\$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 EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize the Button component.
let button: Button = new Button({ content: 'Button' });
// Render initialized 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>
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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { Calendar } from '@syncfusion/ej2-calendars';
enableRipple(true);
// Initialize the touch button.
let touchButton: Button = new Button({ content: 'Touch Mode' });
touchButton.appendTo('#touch');
touchButton.element.onclick = (): void => {
    document.body.classList.add('e-bigger');
}
// Initialize the mouse button.
let mouseButton: Button = new Button({ content: 'Mouse Mode' });
mouseButton.appendTo('#mouse');
mouseButton.element.onclick = (): void => {
    document.body.classList.remove('e-bigger');
}
let calendarObject: Calendar = new Calendar();
//Render initialized calendar.
calendarObject.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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { Calendar } from '@syncfusion/ej2-calendars';
enableRipple(true);
// Initialize the touch button.
let touchButton: Button = new Button({ content: 'Touch Mode' });
touchButton.appendTo('#touch');
touchButton.element.onclick = (): void => {
    let controls = document.querySelectorAll('.control');
    for (let index: number = 0; index < controls.length; index++) {
        controls[index].classList.add('e-bigger');
    }
}
// Initialize the mouse button.
let mouseButton: Button = new Button({ content: 'Mouse Mode' });
mouseButton.appendTo('#mouse');
mouseButton.element.onclick = (): void => {
    let controls = document.querySelectorAll('.control');
    for (let index: number = 0; index < controls.length; index++) {
        controls[index].classList.remove('e-bigger');
    }
}
let calendarObject: Calendar = new Calendar();
//Render initialized calendar.
calendarObject.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>
</html>

```

```

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

```

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

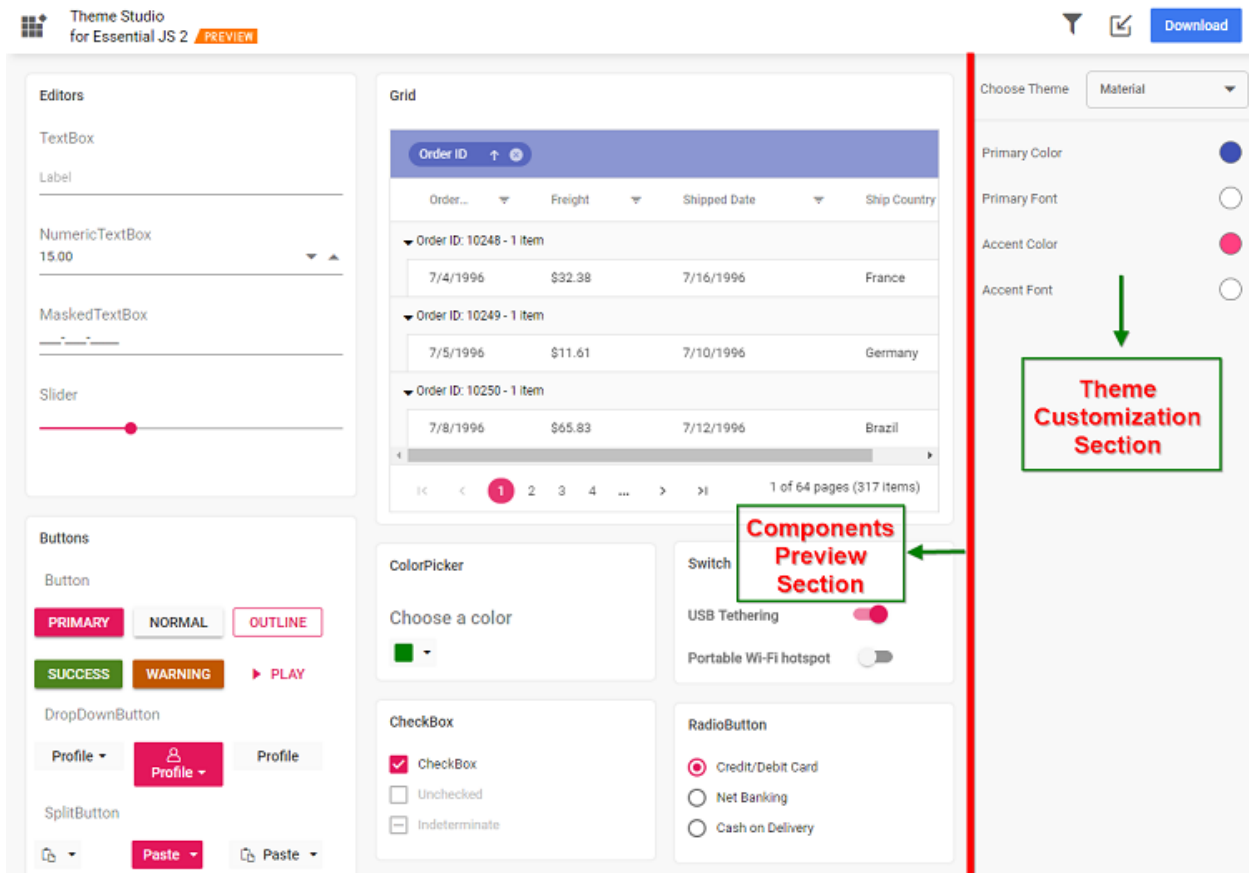
Theme Studio for Essential JS 2 can be used to customize a new theme from an existing theme. It doesn't support with Data visualization controls like Chart, Diagram, Gauge, Range Navigator, Maps.

Customizing theme color from theme studio

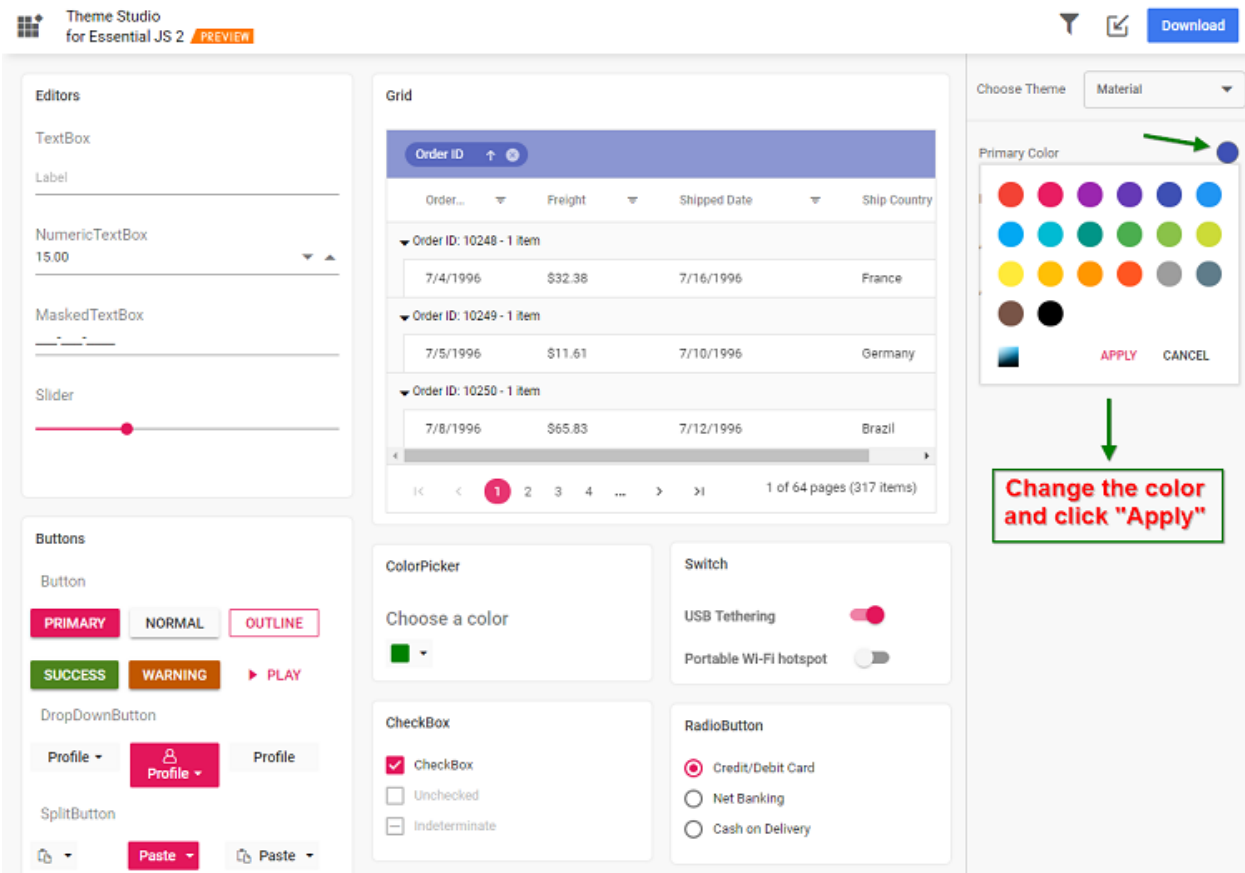
The Essential JS 2 themes are developed under the SCSS environment. Each theme has a unique common variable list. When you change the common variable color code value, it will reflect in all the Syncfusion JavaScript controls. All Syncfusion JavaScript control styles are derived from these [theme-based 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/>.

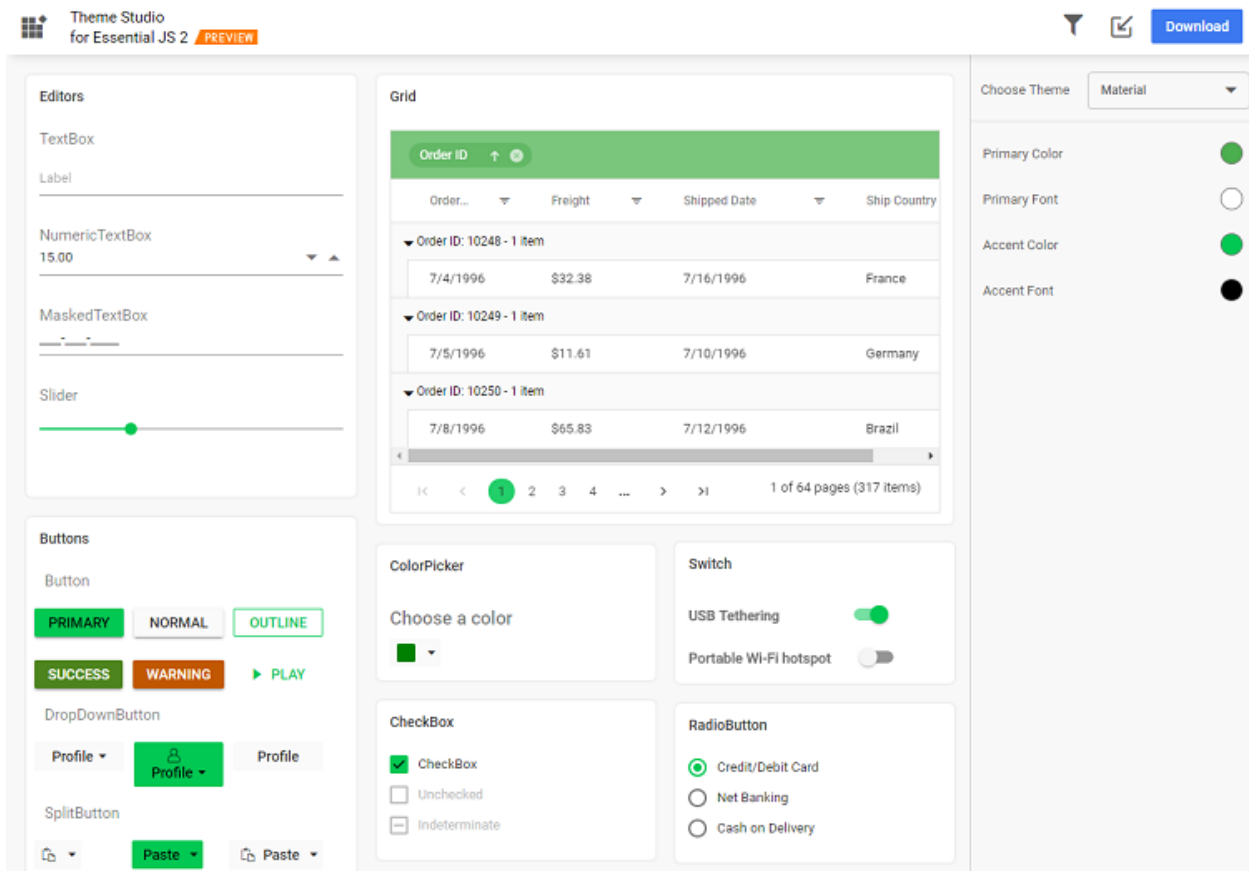
Step 2: The theme studio application page can be divided into two sections: the controls preview section on the left, and the theme customization section on the right.



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



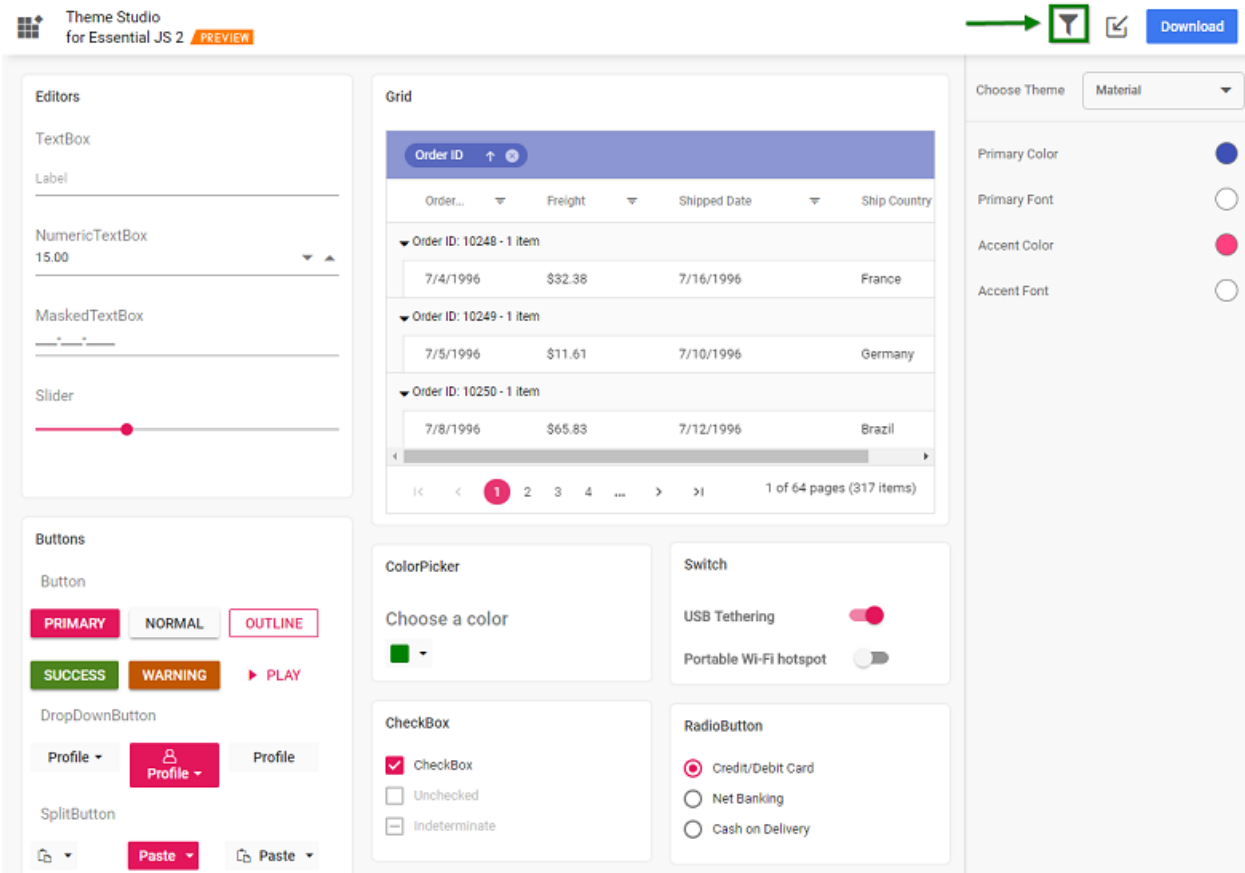
Step 4: The Syncfusion JavaScript UI controls will be rendered with the newly selected colors in the preview section, after selecting the custom color from pickers.



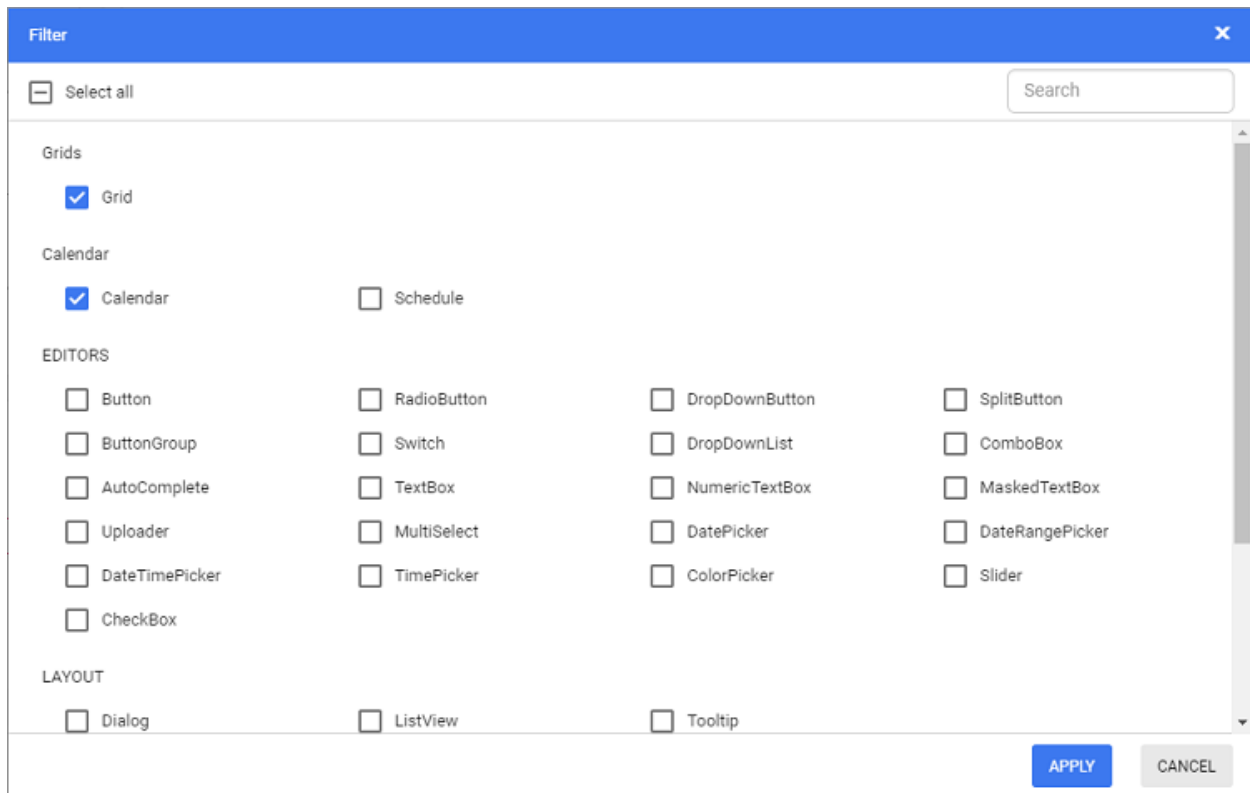
Filtering a specific list of controls

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.

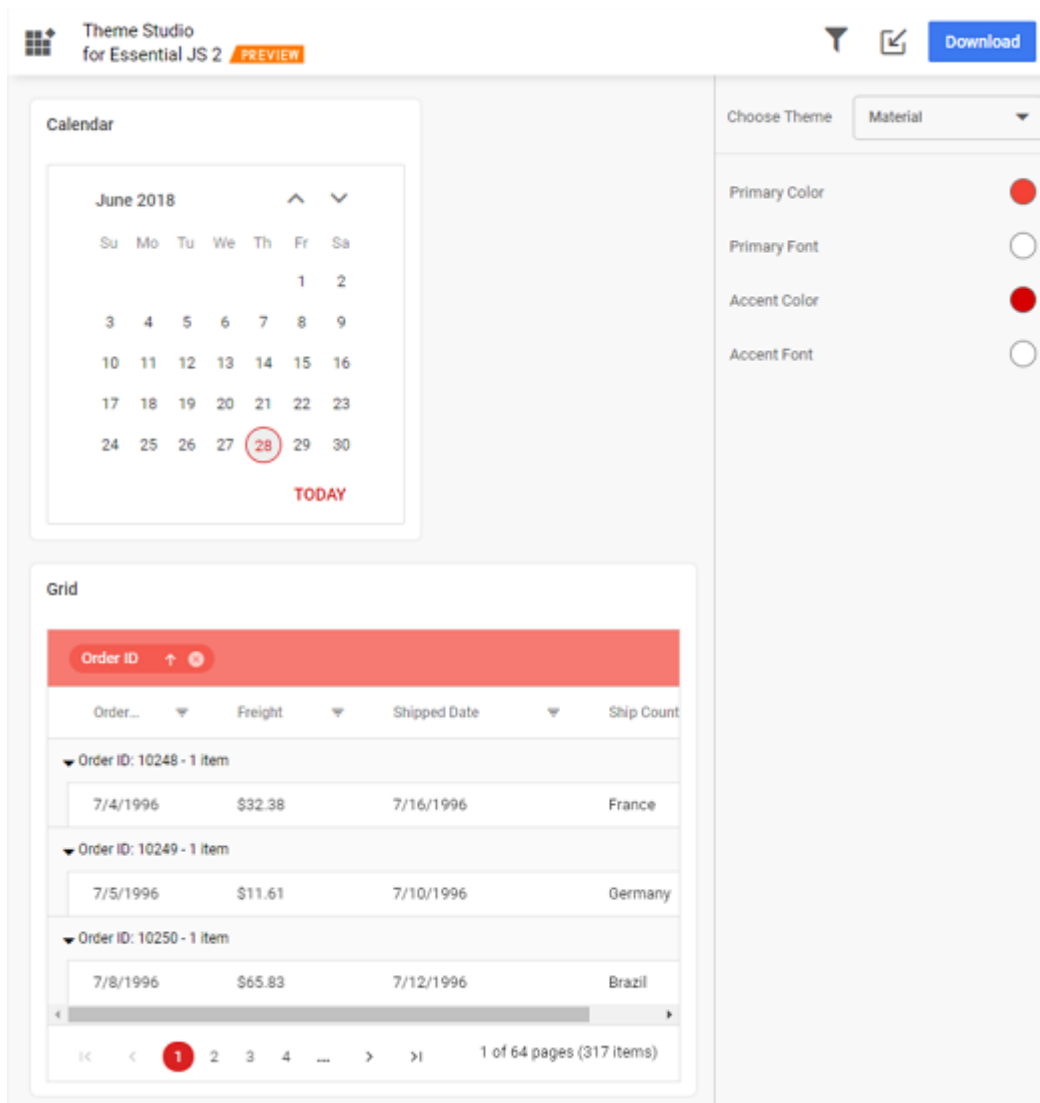
Step 1: Click the Filter icon in the top right corner and select the controls whose theme you want to customize.



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



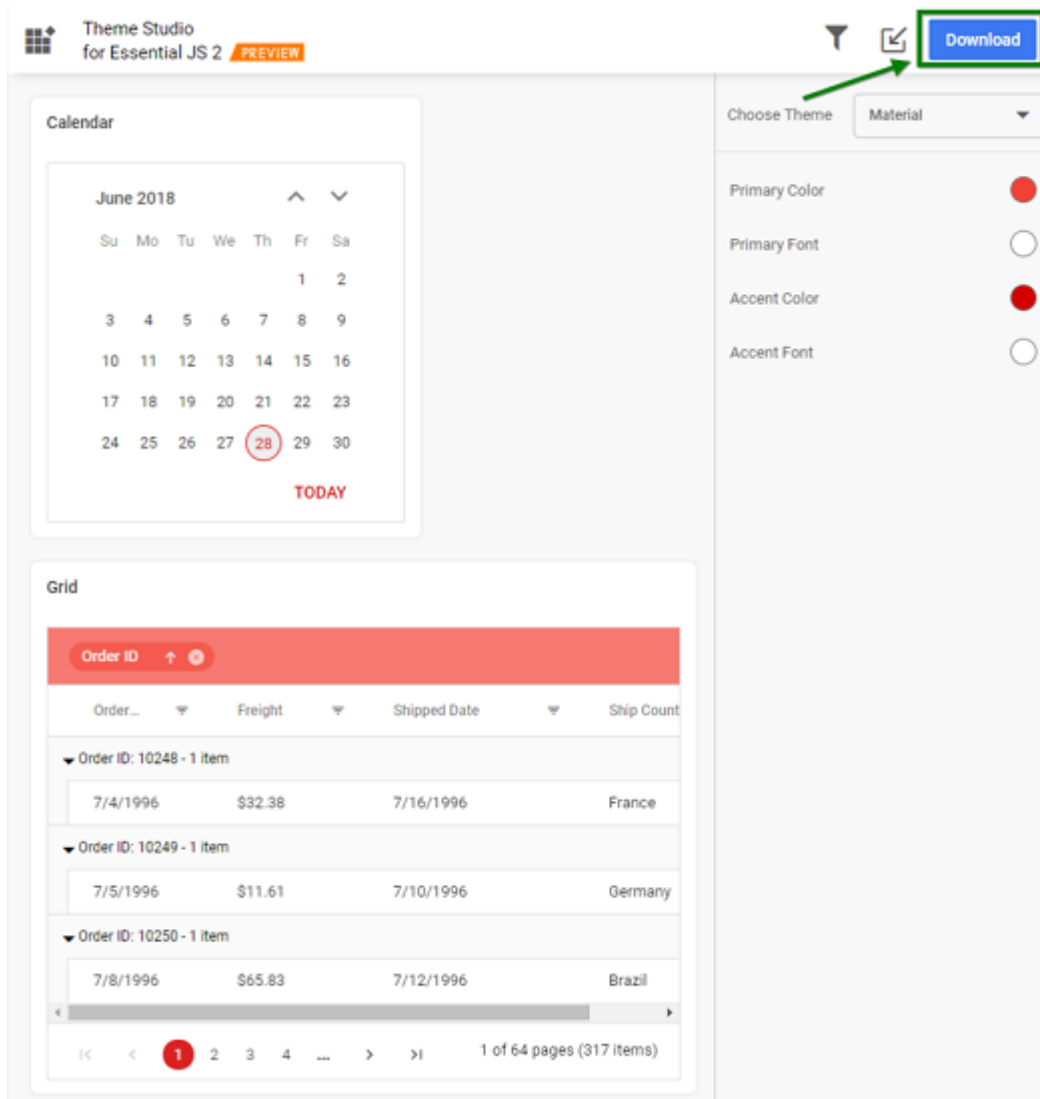
Step 3: Now you can customize the colors in the theme customization section for the controls you selected.



[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.

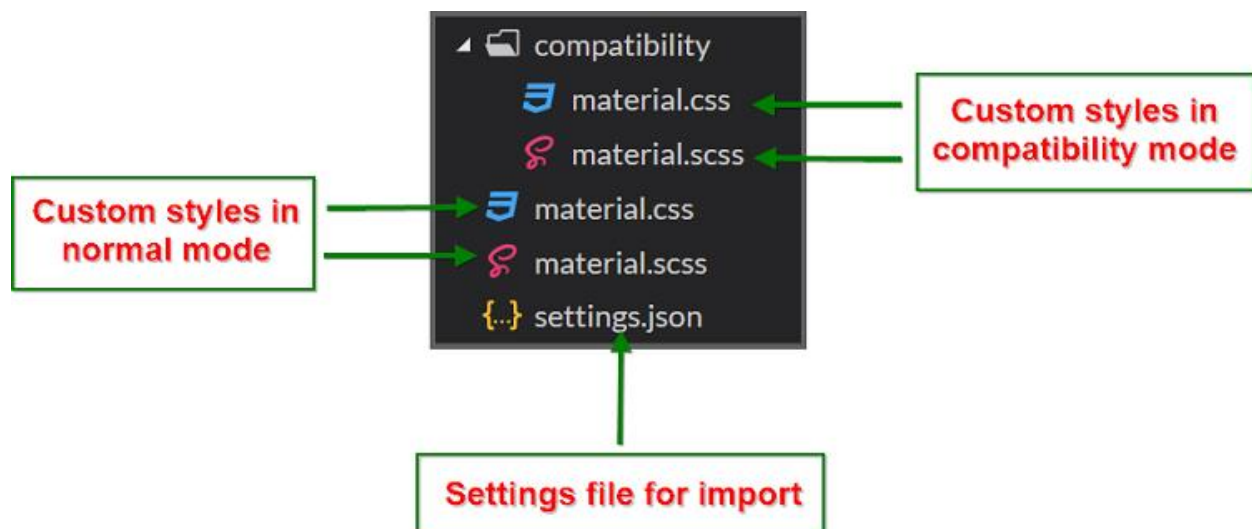
Download [X]

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.

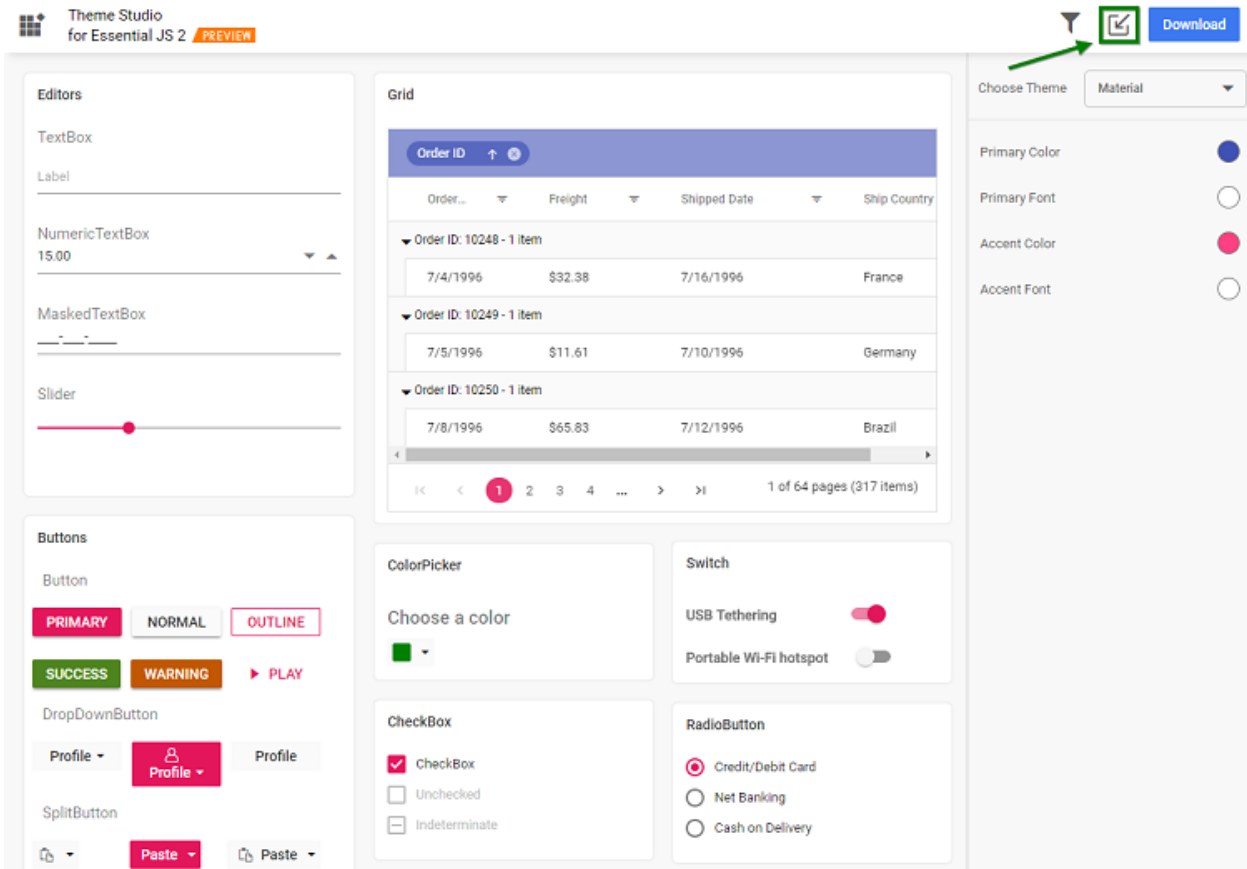
```
<head>
<link href="styles/{file-name}.css" rel="stylesheet"/>
</head>
```

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.

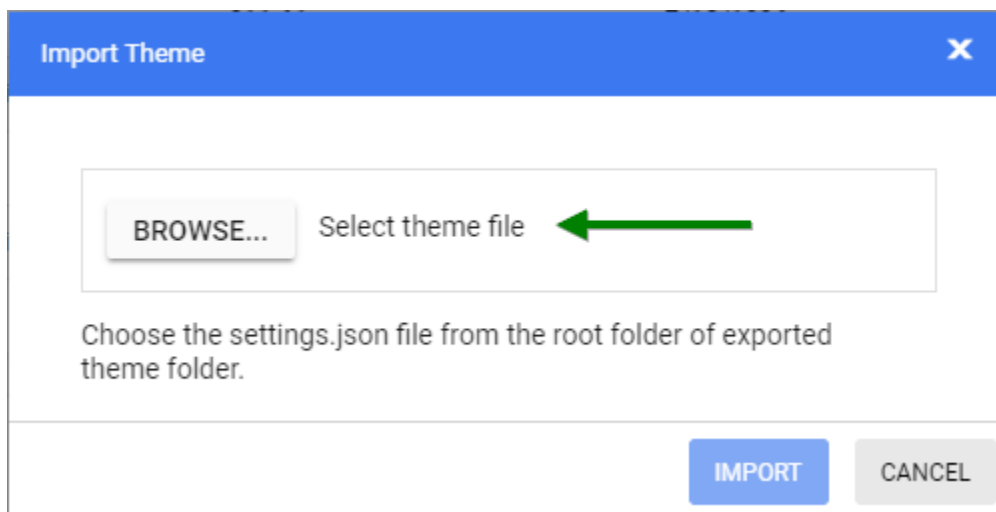
Import previously changed settings into the theme studio

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.

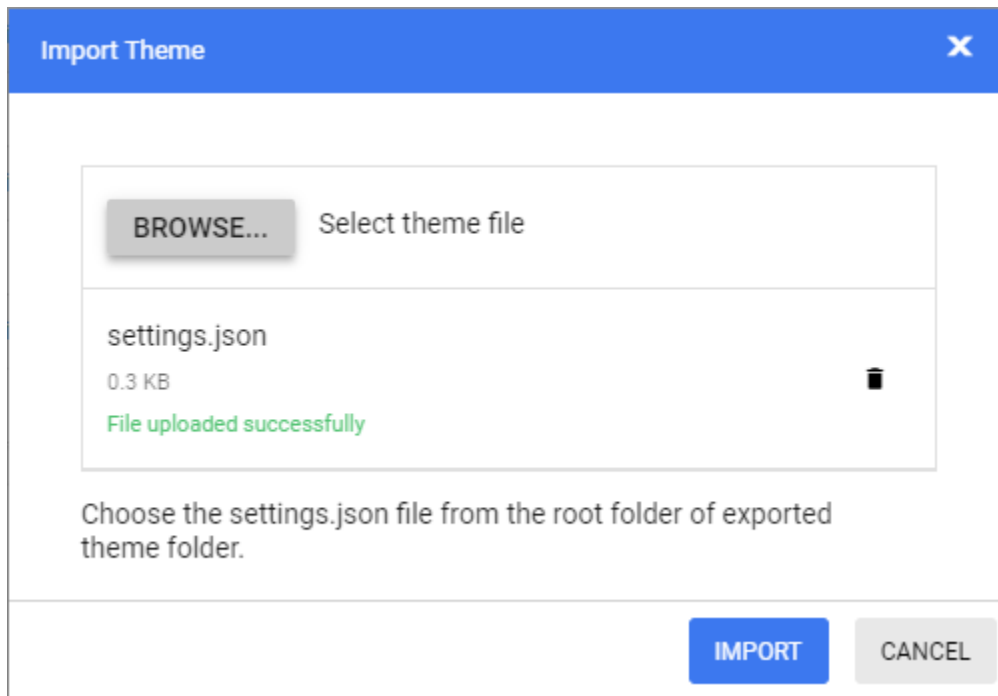
Step 1: Click the Import icon in the top right corner.



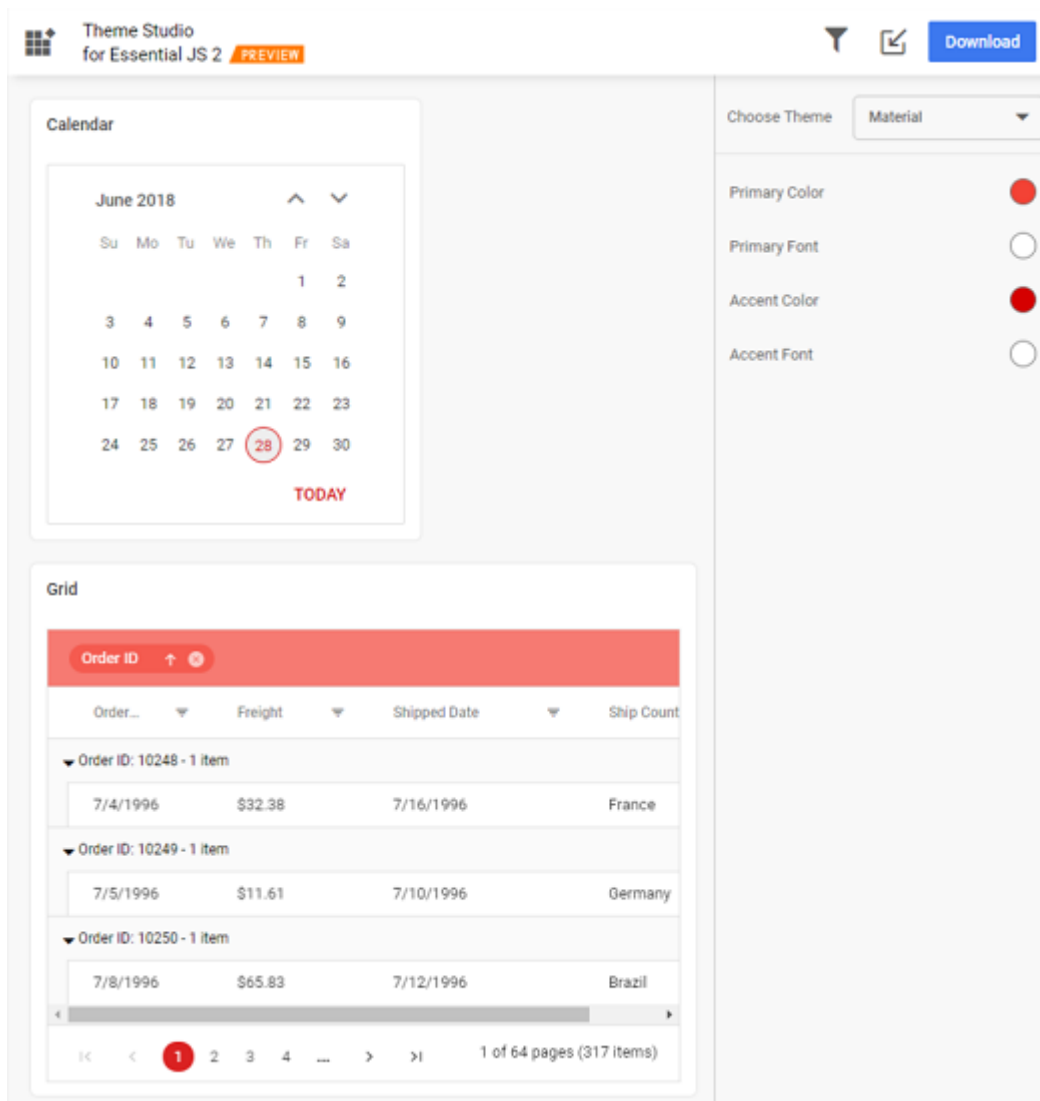
Step 2: The Import Theme dialog will open. Click the Browse button and select a `settings.json` file you exported previously.



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;
```

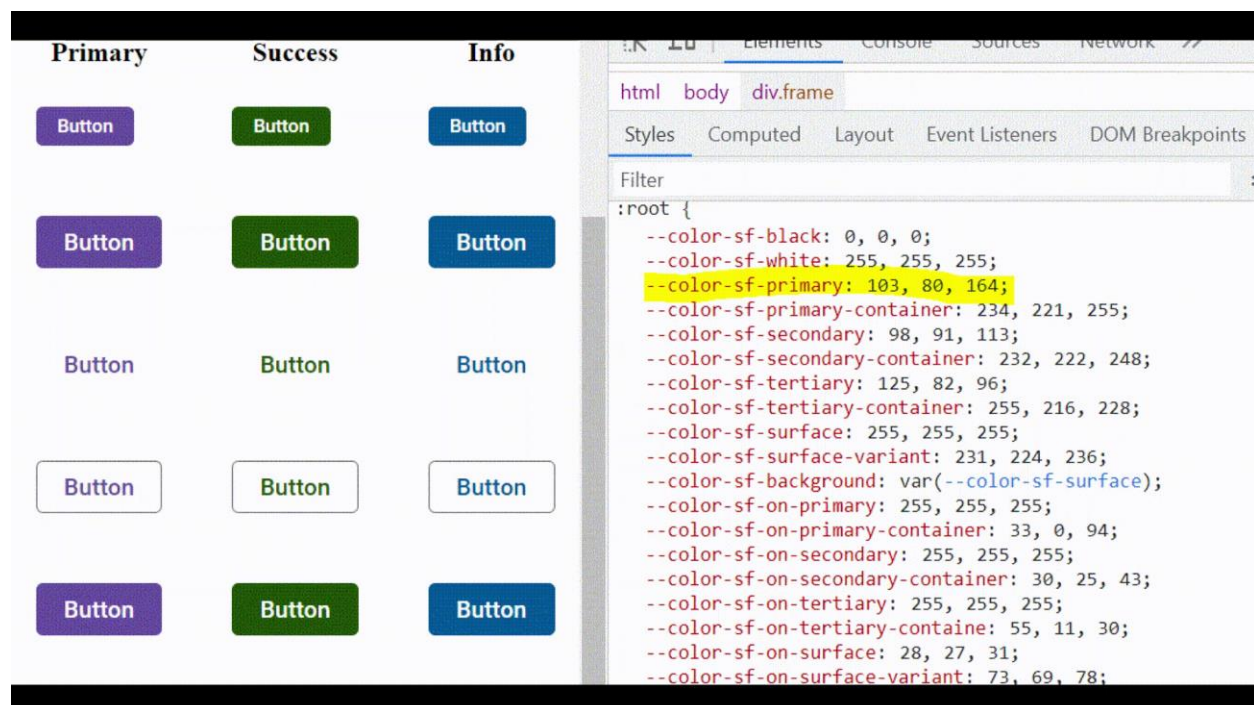
Customized primary value

```
--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.TS

```
import { Button, CheckBox } from '@syncfusion/ej2-buttons';
var checkBoxObj = new CheckBox({ label: 'Enable DarkMode', change:
onDarkMode });
checkBoxObj.appendTo('#darkmode');
function onDarkMode(e: any) {
    e.checked ? document.body.classList.add('e-dark-mode') :
document.body.classList.remove('e-dark-mode');
}
var btnObj = new Button({ isPrimary: true });
btnObj.appendTo('#normal4');
btnObj = new Button({ cssClass: 'e-success' });
btnObj.appendTo('#normal5');
btnObj = new Button({ cssClass: 'e-info' });
btnObj.appendTo('#normal6');
btnObj = new Button({ cssClass: 'e-warning' });
btnObj.appendTo('#normal7');
btnObj = new 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>
```

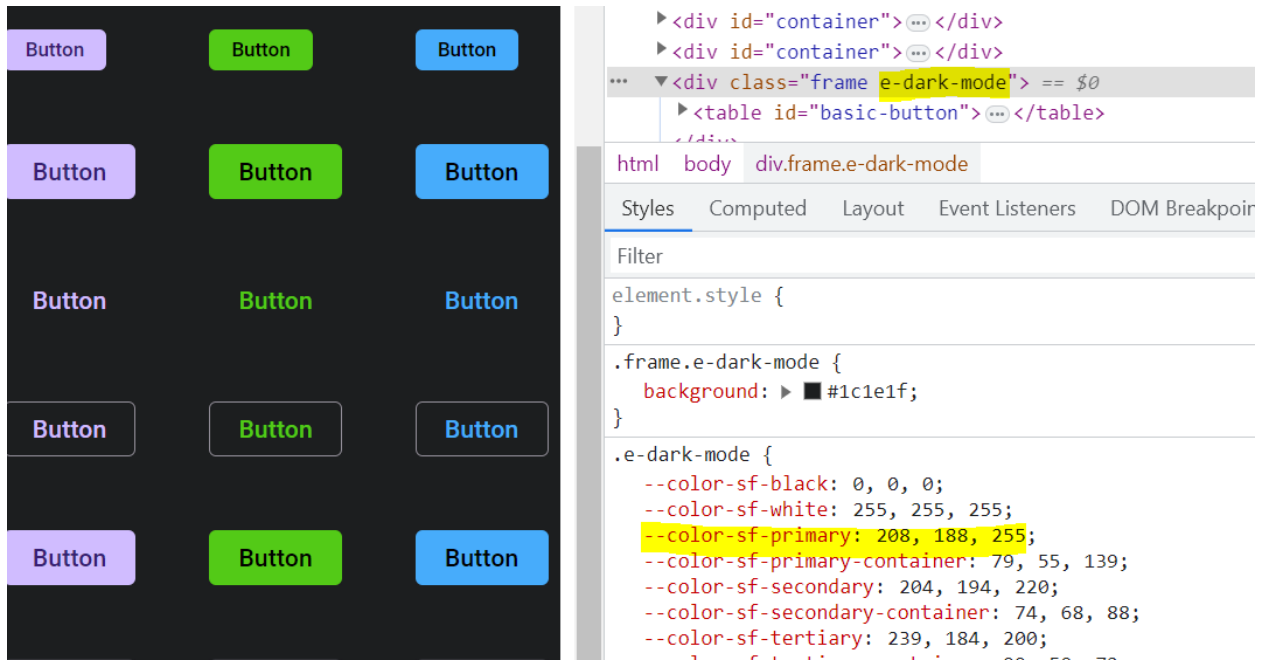
The above example demonstrates the appearance of an application in dark mode. By using CSS variable support, the transition between light and dark mode is smooth and visually pleasing.

`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 EJ2 JavaScript 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 EJ2 JavaScript controls adhere to these established standards.

Accessibility compliance

The accessibility support provided by Syncfusion EJ2 JavaScript 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 EJ2 JavaScript controls, such as:

- The [WAI-ARIA patterns](#) are followed by the Syncfusion EJ2 JavaScript controls to enable appreciable accessibility.
- Each EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript controls support the Right-to-Left feature. Refer to the [Right-to-Left documentation](#) to learn more about this support.

Color contrast

Syncfusion EJ2 JavaScript controls come equipped with [predefined themes](#) that guarantee the presence of satisfactory color contrast, benefiting individuals with visual impairments.

Mobile device support

Syncfusion EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript controls. During manual accessibility testing, testers will ensure accessibility using the screen readers such as [Narrator](#) for Windows and [Embedded VoiceOver](#) for MAC.

Syncfusion EJ2 JavaScript 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 EJ2 JavaScript 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 |
| ListView | Sparkline Charts | Numeric TextBox | Carousel |
| | TreeMap | Masked TextBox | Context Menu |
| | Bullet Chart | | |

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

Animation in EJ2 JavaScript

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.TS

```
import {Animation} from '@syncfusion/ej2-base';
let animation: Animation = new 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.TS

```
import {Animation} from '@syncfusion/ej2-base';
let animation: Animation = new 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.TS

```
import {Animation} from '@syncfusion/ej2-base';
let animation: Animation = new 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 EJ2 JavaScript 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.TS

```
import { GlobalAnimationMode, setGlobalAnimation } from "@syncfusion/ej2-base";
setGlobalAnimation(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.TS

```
import { enableRtl } from '@syncfusion/ej2-base';
import { ListView } from '@syncfusion/ej2-lists';
import { data } from './datasource.ts';
// Enables Right to left alignment for all controls
enableRtl(true);
let rtlListObj: ListView = new 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.TS

```
import { ListView } from '@syncfusion/ej2-lists';
import { data } from './datasource.ts';
let rtlListObj: ListView = new 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.TS

```

import { Grid, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Page);
let grid: Grid = new 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 scrollTop
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 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

	<ul style="list-style-type: none"> • 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.TS

```
import { compile } from '@syncfusion/ej2-base';
// data
let data: object = { name: 'Aston Martin' };
// Compiling template string into executable function
let getDOMString: (data: object) => NodeList =
  compile('<div>${name}</div>');
// Using generated function to get output element collection
let output: NodeList = 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>
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>\${covertToCurrency()}</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.TS

```
import { compile } from '@syncfusion/ej2-base';
let customHelper: Object = {
  upperCase: (str: string) => {
    return str.toUpperCase();
  }
};
let data: object = { name: 'Aston Martin' };
let getDOMString: (data: object) => HTMLCollection =
compile('<div>${upperCase(name)}</div>', customHelper);
let opElem: HTMLCollection = 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 EJ2 JavaScript 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.TS

```
import { Grid } from '@syncfusion/ej2-grids';
let grid: Grid = new 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>
<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.TS

```

import { Grid } from '@syncfusion/ej2-grids';
import { employeeData } from '../datasource.ts';

```



```

let grid: Grid = new 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.TS

```
import { Grid } from '@syncfusion/ej2-grids';
//function template
```

```

var customTemplate = (data: any) => `<div
class="template">${data.ShipCountry}</div>`
let grid: Grid = new 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.TS

```
import { Grid } from '@syncfusion/ej2-grids';
import { employeeData } from './datasource.ts';
//function template
let customTemplate = (data: any) => `<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>`;
let grid: Grid = new 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.TS

```
import { L10n } from '@syncfusion/ej2-base';
import { Grid, Group, Page } from '@syncfusion/ej2-grids';
import { data } from './datasource.ts';
Grid.Inject(Group, Page);
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 Erstaustauslastung
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'
    }
  }
});
let grid: Grid = new 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: 210
});
```

```
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 1. N - denotes numeric type. 2. C - denotes currency type. 3. P - 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.
---	---------------------------------------	--

6	<code>useGrouping</code>	Indicates whether to use grouping. Possible values are <code>true</code> or <code>false</code> . Default value is <code>true</code> .
---	--------------------------	---

| 5 | `maximumSignificantDigits` | Indicates the maximum number of significant digits. . Possible values are 1 to 21. **Note:** If `maximumSignificantDigits` is given it is mandatory to give `minimumSignificantDigits` |

| 6 | `useGrouping` | Indicates whether to enable the group separator or not . By default grouping value will be true. |

| 7 | `minimumIntegerDigits` | Indicates the minimum number of the integer digits to be placed in the value. Possible values are 1 to 21. |

| 8 | `currency` | Indicates the currency code which needs to be 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>	'0123'
---	---	---	--------

#	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>	'1234'
---	---	--	--------

.	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>	'546321.000'
---	---	---	--------------

%	Percent specifier denotes the percentage type format.	<code>instance.formatNumber(1,{format: '0000 %' })</code>	'0100 %'
---	---	---	----------

\$	Denotes the currency type format based on the global currency code specified.	<code>instance.formatNumber(13,{format: '\$ ###.00' })</code>	'\$ 13.00'
----	---	---	------------

;	Denotes separate formats for positive, negative and zero values.	<code>instance.formatNumber(-120,{format: '###.##;(###.00);-0'})</code>	'(120.00)'
---	--	---	------------

'String' (single Quotes)	Denotes the characters enclosed within single Quote(') to be replaced in the resultant string.	<code>instance.formatNumber(-123.44,{format: '####.## '@'})</code>	'123.44 @'
--------------------------	--	--	------------

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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let nParser: Function = intl.getNumberParser({ format: 'P2' , useGrouping:
false });
let val: number = 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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
```

```
let val: number = 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>
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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let nFormatter: Function = intl.getNumberFormat({ skeleton: 'C3', currency:
'USD',minimumIntegerDigits:8});
let formattedValue: string = 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>
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.TS

```

import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let formattedString: string = 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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let dParser: Function = intl.getDateParser({skeleton: 'full', type:
'datetime'});
let val: Date = 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.TS

```
import {Internationalization} from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let val: Date = 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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let dFormatter: Function = intl.getDateFormat({ skeleton: 'full', type:
'datetime' });
let formattedString: string = 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.TS

```
import { Internationalization } from '@syncfusion/ej2-base';
let intl: Internationalization = new Internationalization();
let date: Date = new Date();
let formattedString: string = 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 EJ2 JavaScript

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.TS

```
import {Draggable} from '@syncfusion/ej2-base';
let dragElement: HTMLElement = document.getElementById('element1');
let draggable: Draggable = new 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.TS

```
import {Draggable} from '@syncfusion/ej2-base';
let dragElement: HTMLElement = document.getElementById('element1');
let draggable: Draggable = new Draggable(dragElement, { clone: true });
```

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.TS

```

import { Draggable, Droppable, DropEventArgs } from '@syncfusion/ej2-base';
let draggable: Draggable = new
Draggable(document.getElementById('element1'), {
  clone: false, dragArea: "#droppable"
});

```

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.TS

```

import { Draggable, Droppable, DropEventArgs } from '@syncfusion/ej2-base';
let draggable: Draggable = new
Draggable(document.getElementById('element1'), {
    clone: false
});
let droppable: Droppable = new
Droppable(document.getElementById('droppable'), {
    drop: (e: DropEventArgs) => {
        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.

[View the JavaScript sample enabled with strict CSP in Github](#)

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.

In your application, utilizes string or external templates, it is advisable to rewrite them using the [function template](#) approach for template properties.

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

Syncfusion license banner utilize the image from **base64**, which is not allowed on strict CSP-enabled sites. To overcome this restriction, it is necessary to add the [img-src data:](#) directive in the meta tag or consider [registering the license key](#).

3D Chart

<!-- 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
import { DataManager, Query } from '@syncfusion/ej2-data';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let dataManager: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/orders'
});
let query: Query = new Query().take(5).where('Estimate', 'lessThan', 3,
false);
let chart: Chart3D = new Chart3D({
  //Initializing Primary X and Y Axis
  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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
import { DataManager, Query, ODataAdaptor } from '@syncfusion/ej2-data';
let dataManager: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/orders',

```

```

    adaptor: new ODataAdaptor()
  });
let query: Query = new Query();
let chart: Chart3D = new Chart3D({
  //Initializing Primary X and Y Axis
  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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    //Initializing Primary X and Y Axis
    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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new Chart3D({
  //Initializing Primary X and Y Axis
  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.

```

`javascript
<div id='container'>

<div id='element' style="width:650px; height:350px;"></div>

</div>
`

```

INDEX.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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: Chart3D = new 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>
  <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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
```

```
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
```

```

Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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>
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

The [numeric axis](#) can be used to represent the numeric values of data in 3D chart. By default, the `valueType` of an axis is **Double**.

INDEX.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];

let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';

```

```

Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [{ 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 }];
let chart: Chart3D = new 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.TS


```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    useGroupingSeparator: true,
    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>
```

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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    primaryYAxis: {
        // Custom label format
        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>
<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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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 `Date3D` module using `Chart3D.Inject(Date3D)` method and set the `valueType` of axis to **Date**.

Date category axis

Date 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.TS

```

import {Chart3D, DateCategory3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateCategory3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    primaryXAxis: {
        valueType: 'DateCategory',
        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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
let chart: Chart3D = new 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.TS

```
import {Chart3D, DateTimeCategory3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTimeCategory3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];

```

```
let chart: Chart3D = new 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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, DateTime3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
  primaryXAxis: {
    valueType: 'DateTime'
  },
  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: 10^8).

INDEX.TS

```
import { Chart3D, DateTime3D, Legend3D, Logarithmic3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Logarithmic3D, Legend3D,
Tooltip3D, Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import { Chart3D, DateTime3D, Legend3D, Logarithmic3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Logarithmic3D, Legend3D,
Tooltip3D, Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import { Chart3D, DateTime3D, Legend3D, Logarithmic3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Logarithmic3D, Legend3D,
Tooltip3D, Highlight3D);
let chartData: any[] = [
  { 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: Chart3D = new 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.TS

```

import { Chart3D, DateTime3D, Legend3D, Logarithmic3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, DateTime3D, Logarithmic3D, Legend3D,
Tooltip3D, Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
]

```

```

];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';

```

```

Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
</html>

```

```

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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D, I3DAxisLabelRenderEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let series1: Object[] = [];
let point1: Object;
let value: number = 80;
let i: number;
for (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);
}
let chart: Chart3D = new 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>
<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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
<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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }

```

```

];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [

```

```

    { 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 }
];
let chart: Chart3D = new 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

To render a [column series](#), use series [type](#) as `Column` and inject `ColumnSeries3D` module using `Chart3D.Inject(ColumnSeries3D)` method.

INDEX.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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>
<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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
```



```
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let stackedData: object[] = [
    { 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 }
];
let chart: Chart3D = new 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>
</body>
<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.TS

```
import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let stackedData: object[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let stackedData: object[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let stackedData: object[] = [
    { 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 }
];
let chart: Chart3D = new 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

To render a [100% stacked column](#) series, use series [type](#) as `StackingColumn100` and inject `StackingColumnSeries3D` module using `Chart3D.Inject(StackingColumnSeries3D)` method.

INDEX.TS

```

import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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

To render a [bar series](#), use series [type](#) as `Bar` and inject `BarSeries3D` module using `Chart3D.Inject(BarSeries3D)` method.

INDEX.TS

```
import {Chart3D, Category3D, Legend3D, BarSeries3D, Tooltip3D, Highlight3D,
Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(BarSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, BarSeries3D, Tooltip3D, Highlight3D,
Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(BarSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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>
</body>
</html>

```

```

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.TS

```

import {Chart3D, Category3D, Legend3D, BarSeries3D, Tooltip3D, Highlight3D,
Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(BarSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D);
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, BarSeries3D, Tooltip3D, Highlight3D,
Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(BarSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, BarSeries3D, Tooltip3D, Highlight3D,
Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(BarSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D);
let numData: object[] = [
  { 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 }];

```



```
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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

To render a [100% stacked bar](#) series, use series [type](#) as `StackingBar100` and inject `StackingBarSeries3D` module using `Chart3D.Inject(StackingBarSeries3D)` method.

INDEX.TS

```
import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        {
            interval: 25
        },
    },
    //Initializing Chart Series
    series: [
        {
            dataSource: chartData, xName: 'x', yName: 'y',
            type: 'StackingBar100',
            name: 'General Motors'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y1',
            type: 'StackingBar100', name: 'Honda'
        }, {
            dataSource: chartData, xName: 'x', yName: 'y2',
            type: 'StackingBar100', 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.TS

```
import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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,
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import {Chart3D, Category3D, Legend3D, StackingBarSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(StackingBarSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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>
<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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```
import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, I3DTextRenderEventArgs } from '@syncfusion/ej2-
charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new Chart3D({
  textRender: (args: I3DTextRenderEventArgs) => {
    if (args.point.index === 2) {
      args.text = 'Label';
    }
  }
});

```

```

    }
    else {
        args.cancel = true;
    }
},
primaryXAxis: {
    valueType: 'Category',
},
series:[{
    dataSource: chartData,
    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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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>
</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 size

The size of the legend items can be customised by using the [shapeHeight](#) and [shapeWidth](#) properties.

INDEX.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [

```

```

    { 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 }
  ];
  let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D, Chart3DLoadedEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let colors: string[] = ['#00BDAE', '#404041', '#357CD2'];
let chart3D: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D, Selection3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D, Highlight3D,
Selection3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
  <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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';

```

```

Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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: Chart3D = new 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>
    </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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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: Chart3D = new 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.TS

```
import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
```

```

let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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>
<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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS


```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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>
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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D} from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', 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.TS

```

import { DataLabel3D, Chart3D, Category3D, Legend3D, ColumnSeries3D,
Tooltip3D, Highlight3D, Selection3D, Export3D} from '@syncfusion/ej2-
charts';
Chart3D.Inject(ColumnSeries3D, Category3D, DataLabel3D, Legend3D, Tooltip3D,
Highlight3D, Selection3D, Export3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new Chart3D({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        xName: 'country', yName: 'gold',
        name: 'Gold', 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D, I3DPointRenderEventArgs } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let colors: string[] = ['#00bdae', '#404041', '#357cd2', '#e56590',
'#f8b883',
'#70ad47', '#dd8abd', '#7f84e8', '#7bb4eb', '#ea7a57'];
let chart: Chart3D = new Chart3D({
    pointRender: (args: I3DPointRenderEventArgs) => {
        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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];

```



```
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
```

```

Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```
import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
    { 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 }
];
let chart: Chart3D = new 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.TS

```

import {Chart3D, Category3D, Legend3D, ColumnSeries3D, Tooltip3D,
Highlight3D } from '@syncfusion/ej2-charts';
Chart3D.Inject(ColumnSeries3D, Category3D, Legend3D, Tooltip3D,
Highlight3D);
let chartData: any[] = [
  { 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 }
  ];
let chart: Chart3D = new 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

Getting started in EJ2 JavaScript Accordion control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript components.

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

Syntax:

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

Styles: **(installed location)**/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css

Example:

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

Styles: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.30/Essential JS 2/ej2-navigations/styles/material.css

Step 3: Create a folder `myapp/resources` and copy/paste the global scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (index.html) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Accordion's global script -->
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```


Step 5: Now, add the **Accordion** element and initiate the **Essential JS 2 Accordion** component in the **index.html** by using following code

Initialize the Accordion using JSON items collection

The Accordion can be rendered by defining an array of [items](#).

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Accordion's global script -->
<script src="resources/ej2-navigations.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the HTML <div> element -->
<div id="element"></div>
<script>
//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');
</script>
```

```
</body>
```

```
</html>
```

```
,
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Accordion** component.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

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

Styles: `http://cdn.syncfusion.com/ej2/{PACKAGE_NAME}/styles/material.css`

Example:

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

Styles: <http://cdn.syncfusion.com/ej2/ej2-navigations/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `Accordion` element and initiate the `Essential JS 2 Accordion` component in the `index.html` by using following code.

INDEX.HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
navigations/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 all 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>
    <!-- Add the HTML <div> element -->
    <div id="element"></div>
    <script>
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');
</script>
</body>
</html>

```

Step 4: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Accordion** component.

Initialize the Accordion using HTML elements

The Accordion component can be rendered based on the given HTML element using `id` as `target` property.

You need to follow the below structure of HTML elements to render the Accordion.

```

,
<div id='accordion/html/markup'> --> Root Accordion Element
<div> --> Accordion Item Container
<div> --> Accordion Header Container
<div> </div> --> Accordion Header
</div>
<div> --> Accordion Panel Container
<div> </div> --> Accordion Content
</div>
</div>
</div>
,

```

INDEX.JS

```

ej.base.enableRipple(true);
//Initialize Accordion component
var accordion = new ej.navigations.Accordion({});
//Render initialized Accordion component
accordion.appendTo('#accordion_html_markup');

```

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="accordion_html_markup">
      <div>
        <div>
          <div> ASP.NET </div>
        </div>
        <div>
          <div> Microsoft ASP.NET is a set of technologies in the
Microsoft .NET Framework for building Web applications
and XML Web services </div>
        </div>
      </div>
      <div>
        <div>
          <div> ASP.NET MVC </div>
        </div>
        <div>
          <div> The Model-View-Controller (MVC) architectural
pattern separates an application into three main components:
the model, the view, and the controller </div>
        </div>
      </div>
      <div>
        <div>
          <div> JavaScript </div>
        </div>
        <div>
          <div> 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 </div>
        </div>
      </div>
    </div>
  </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>
```

You can add the custom class into Accordion component using [cssClass](#) property which is used to customize the Accordion component.

See Also

- [How to load accordion items dynamically](#)

Expand mode in EJ2 JavaScript Accordion control

The [JavaScript Accordion](#) supports the two listed types of expand modes while expanding or collapsing the item.

- 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.TS

```
import {Accordion} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let accordion: Accordion = new 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.' },
    ]
});
```

```
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.TS

```
import {Accordion} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let accordion: Accordion = new 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.' },
  ]
});
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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Accordion, AccordionClickArgs, ExpandEventArgs } from
 '@syncfusion/ej2-navigations';
let nestAcrdn_vid: Accordion;
let nestAcrdn_mus: Accordion;
let nestAcrdn_musNew: Accordion;
let nestAcrdn_img: Accordion;
let accordion: Accordion = new 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>' },
  ]
});
accordion.appendTo('#element');
function clicked(e: AccordionClickArgs): void {
  let ele: HTMLElement = e.originalEvent.target;
  if (ele.querySelectorAll('.e-accordion').length > 0) {
    return;
  }
  if (!document.getElementById("nested_musicNew").classList.contains("e-accordion")) {
    nestAcrdn_musNew = new Accordion({
      items: [
        { header: 'New Track1' },
        { header: 'New Track2' }
      ]
    }, '#nested_musicNew');
  }
}
function expanding(e: ExpandEventArgs): void {
  if (e.isExpanded && [].indexOf.call(this.items, e.item) === 0) {
    if (e.element.querySelectorAll('.e-accordion').length > 0) {
      return;
    }
    nestAcrdn_vid = new 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 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 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 EJ2 JavaScript 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.TS

```

import {Accordion} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { Ajax } from '@syncfusion/ej2-base';
enableRipple(true);
let ajax: Ajax = new Ajax('./ajax.html', 'GET', true);
ajax.send().then();
ajax.onSuccess = (data: string): void => {
  let ctn2: string = data;
  let accordion: Accordion = new Accordion({
    items: [
      { header: 'Department', content: '#acrdnContent1' },

```

```

        { header: 'Platform', content: '#acrdnContent2' },
        { header: 'Employee Details', content: ctn2 }
    ]
});
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 EJ2 JavaScript 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.TS

```
import {Accordion, AccordionEffect} from '@syncfusion/ej2-navigations';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let accordion: Accordion = new Accordion({
    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.' },
    ]
});
accordion.appendTo('#element');
let listObjExpand: DropDownList = new DropDownList({
    index: 0,
    placeholder: 'Select a animate type',
    popupHeight: '150px',
    change: () => { valueChange(); }
});
listObjExpand.appendTo('#expandAnimation');
valueChange();
let listObjCollapse: DropDownList = new DropDownList({
    index: 1,
    placeholder: 'Select a animate type',
    popupHeight: '150px',
    change: () => { valueChange1(); }
});
listObjCollapse.appendTo('#collapseAnimation');
valueChange1();
function valueChange(): void {
    accordion.animation.expand.effect =
    <AccordionEffect>(listObjExpand.value);
}
function valueChange1(): void {
```

```

    accordion.animation.collapse.effect =
    <AccordionEffect>(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 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>
<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>

```

To keep single pane open always in EJ2 JavaScript 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.TS

```

import {Accordion, ExpandEventArgs, AccordionClickArgs} from
 '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let clickEle: HTMLElement;
let accordion: Accordion = new 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.' },
  ]
});
accordion.appendTo('#element');
function clicked (e: AccordionClickArgs): void {
  clickEle = (e.originalEvent.target as Element).closest('.e-acrdn-header');
}

```

```
function beforeExpand (e: ExpandEventArgs):void {
    let expandCount: number = this.element.querySelectorAll('.e-selected').length;
    let ele: HTMLElement= this.element.querySelectorAll('.e-selected')[0];
    if (ele) {
        ele = ele.firstChild as Element;
    }
    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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
import { DatePicker } from '@syncfusion/ej2-calendars';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
import { Accordion, ExpandEventArgs } from '@syncfusion/ej2-navigations';
enableRipple(true);
let success: string = 'Your payment successfully processed';
let email_alert: string = 'Enter valid email address';
let mobile_alert: string = 'Mobile number length should be 10';
let card_alert: string = 'Card number length should be 16';
let cvv_alert: string = 'CVV number length should be 3';
let datePicker: DatePicker = new DatePicker({
    width: '100%',
    format: 'MM/yyyy',
    floatLabelType: 'Auto',
    placeholder: 'Expiry Date'
});
datePicker.appendTo("#expiry");
let cardNum: NumericTextBox = new NumericTextBox({
    placeholder: 'Card No',
    floatLabelType: 'Auto',
    format: '0',
    showSpinButton: false
});
cardNum.appendTo("#cardNo");
let cvv: NumericTextBox = new NumericTextBox({
    placeholder: 'CVV',
    floatLabelType: 'Auto',
    format: '0',
    showSpinButton: false
});
cvv.appendTo("#CVV");
let mobile: NumericTextBox = new NumericTextBox({
    placeholder: 'Mobile',
    floatLabelType: 'Auto',
    format: '0',
    showSpinButton: false
});
mobile.appendTo("#mobile");
let alertDialog: Dialog = new 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();
let acrdnObj: Accordion = new Accordion({
  expanding: expand,
  created: () => {
    acrdnObj.enableItem(1, false);
    acrdnObj.enableItem(2, false);
  },
  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: string): void {
  if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/ .test(mail)) {
    return (true);
  } else {
    alertDialog.content = email_alert;
    alertDialog.show();
    return (false)
  }
}
function checkMobile(mobile: number): void {
  if (mobile.match(/^\d{10}$/)) {
    return (true);
  } else {
    alertDialog.content = mobile_alert;
    alertDialog.show();
    return (false)
  }
}
function checkCardNo(cardNo: number): void {
  if (cardNo.match(/^\d{16}$/)) {
    return (true);
  } else {
    alertDialog.content = card_alert;
    alertDialog.show();
    return (false)
  }
}
function checkCVV(cvv: number): void {
  if (cvv.match(/^\d{3}$/)) {
    return (true);
  } else {
    alertDialog.content = cvv_alert;
    alertDialog.show();
    return (false)
  }
}

```

```

}
function expand(e: ExpandEventArgs): void {
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 0) {
        document.getElementById('Continue_Btn').onclick = (e : Event) => {
            let email: string = document.getElementById('email');
            let password: string = 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 : Event) => {
            let name: string = document.getElementById('name');
            let address: string = document.getElementById('address');
            let mobile: number = 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 : Event) => {
            acrdnObj.enableItem(1, true);
            acrdnObj.enableItem(2, false);
            acrdnObj.expandItem(true, 1);
        }
        document.getElementById('Save_Btn').onclick = (e : Event) => {
            let cardHolder: string = document.getElementById('cardHolder');
            let expiry: number = document.getElementById('expiry');
            let cardNo: number = document.getElementById('cardNo');
            let cvv: number = 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>
            <div style="text-align: center">
                <button class="e-btn" id="Continue_Btn">Continue</button>
                <div id="err1">* Please fill all fields</div>
            </div>
        </form>
    </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>
    <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 EJ2 JavaScript 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.TS

```

import { Accordion } from '@syncfusion/ej2-navigations';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from '@syncfusion/ej2-data';
let itemsData: any = [];
let mapping = { header: 'FirstName', content: 'Notes' };
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Employees';
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor,
crossDomain: true })
    .executeQuery(new Query().range(1, 4)).then((e: ReturnOption) => {
        let result: any = e.result;
        for(let i: number = 0; i < result.length; i++) {
            itemsData.push({ header: result[i][mapping.header], content:
result[i][mapping.content] });
        }
        let accordion: Accordion = new Accordion({
            items: itemsData
        });
        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 EJ2 JavaScript 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.TS

```

import { Accordion, ExpandEventArgs, AccordionClickArgs, AccordionItemModel
} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { accordion } from './datasource.ts';
enableRipple(true);
let dbFlag: number = 0;
let dynamciAcrdnCount: number = 2;
let acrdnObj: Accordion = new 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.' },
    ]
});
acrdnObj.appendTo('#Accordion_default');
function expanded( e: ExpandEventArgs) {
    //Find the expanded state item in accordion
    let 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)) {
        let array: AccordionItemModel[] = accordion as
AccordionItemModel[];
        for(let i: number = 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 EJ2 JavaScript 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.TS

```

import { Accordion, ExpandEventArgs, AccordionClickArgs, AccordionItemModel } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { accordion } from './datasource.ts';
enableRipple(true);

//Initialize Accordion component
let acrdnObj: Accordion = new 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
acrdnObj.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>
    </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,AAEAAAAKAIAAAwAgTlMvMj0gSSSYAAAEoAAAAVmNtYXDNn+g2AAAB5AAAAGRnbHlmslg
PRQAAAnwAABvMaGVhZA6+wXwAAADQAAAAANmhoZWEHfaOBAAAArAAAAACRobXR4YcP/xgAAAYAAAAAB
kbG9jYU2IVXoAAAJIAAAANG1heHABLwC3AAABCAAAACBuYW1lNl/OpQAAHkgAAAKFcG9zdBxr6o4
AACDQAAABawABAAADUv9qAFoEAP/i//0D6wABAAAAAAAAAAAAAAAAAAGQABAAAAQAAXGQXJ18
PPPUACwPoAAAAANXpvlcAAAAA1em+V//iAAD6wPpAAAAACAACAAAAAAAAAAAAEAAAAZAKsADAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQpPAAABQAAAnoCvAAAAIwCegK8AAAB4AAxQIAAAIABQMMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnFwNS/2oAWgPpAJYAAAAABAAAAAAAAAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+j/9wPo//AD6AAAA+gAAAPo/+ID6AAAA+gAAAPoAAAD6AAAA+gAAAP
oAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6P/9A+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQ
AUAAAAAQABAABADnF///AADnAP//AAAAQAEEAAAAQAACAAMABAAFAAYABwAIAAKACgALAAWADQA
OAA8AEAARABIAEwAUABUAFgAXABgAAAAAFgAqgFGAkACjAL0A0AEGASOBPgFjgZeBrIHiggWCMI
JOAoUCpILWgWUDI4NXA3mAAIAAAAAA3sD5wANADcAAAEeARcWNicuAScmJw4BBRc3FxEUDwEOAR8
BHgE/ATUfARY7ATI2LwImPQEZmJY3NS4BIyEnNycBxQUnHDNDBwUnGwkJLTf+Qha70geWBQMDMwQ
MBtZ5HgQLYAgIAStpBc8HCAEBcAf+O5OsFgL/HCKGCEEzHCgGAgEBP+4WutP+3AgFYgQMBk4FAwS
Mi3KJDAsHxGIFBs0IBz4HCZOsFgAABAAAAAADhwPoAAMAFAAAdAC8AADcXNy8BHgEHNzEXBzYWFx4
BNwEGFiUOASImNDYyFiUGBwE+ATc+AScuAjEmBw4BAVjxVyMCAwUjgyIKGA4lTyj+rgcDAjkBK0A
rK0Ar/gkkCgFfGjEYRaNML1M2NzRekOJY8lg5DxkII4QjAwECBwIGAVInUFQgKytAKYuDMjH+oQY
WEizNfkJQJyUCB4cAAAAFAAAAAAPpA+cACwAXACMAWABhAAAlDgEHLgEnPgE3HgElDgEHLgEnPgE
3HgEFHgEXPgE3LgEnDgEDBg8BDgEfARYPASyGBw4BFx4BNz4BPwE+AS8BJjY/AjYWHwEeATczJy4
BIwciLwEuASciNx4BMjY0JiIGAUICSjk4SwEBSzg5SgJ3AU54OEsCAks4OEv+XwJmTk1nAgJnTU5
mhRENfCESEkEGCD9EstQUARMzsEUaHQF4EwYOPwQEBkwECBAFIw83IpgBASIA5SQ0GKRA7IhIGATF
JMDBJMcY4SwEBSzg4SwEBS0k5SgICSjk4SwEBSzhOZWICZ05NZwICZwGbbGpGk4mhQsLU0EXWCZ
XJ1kaQBLcJJ4aQB5/Bw4FOgIEBQLHHIIBQBoiAQtsHIUBWYQxMUKwMAEAAAAAAPiA+cAaAB1AJ4
AqgAAEYIHDgEHFT4BNj4BMjY3HgEYnJceATI2Nx4BMjY3HgEYnJceArc1LgEnJiM1ByIHFAYHLgE
1JiM1ByIHFAYHLgEnJiMnFSIHFAYHLgE1JiMnFSIHFAYHLgE1JiMnFSIHFAYHLgE1JiM1AQYWFxY
+ASynJiMOAQUeAR8BHgEzPgE1FBYXPgE1FBYXPgE3LgEnLgEvAS4BJzc2NycmJw4BEx4BFz4BNy4
BJw4BUggEASiIhigLCyk+KQsLKT4pCwsqPSoKCyO9KgsLKT0qCwspHiIjAQQIAQcFJCIjJAQIAQc
FJCIjIwEECAEIBCQiIyQECAEIBCQjIiQECAEIBCQjIiQFCAELDTA0NE4fJzMTEy1C/t0NfB4LCh0
TKikqKikqKSoaJAoMaxMOTCpQMHQFBgYBCS0fECg4ATosLDoBATosLDoBGgYDIwMrArkMDRgYDQ0
YGA0NGBgNDRgYDQ0YGA0MGQErAyIEBgEBBgMkAgIkAwYBAQYDJAICiWQGAQEGAYQCAiQDBgEBBgM
kAgIkAwYBAQYDJAICJAMGAQGCNfKRDtZnVBAFATGUHII2JgkMAyoDAyoDAyoDAyoDARULdMkNBg8
GDA5SBTo2MgEFHzOPAScsOgEBOiwsOgEBOgAAAAAMAAAAAA+ID5wALABcAKQAAARY2NxY2Ny4BJxY
GAx4BFz4BNy4BJw4BBQclFwUWNjCWNjceATCWNjCBAsgqYQQgUxksRBwCotABQzIyQwEBQzIyQ/3
ADwE6Lv66G2wkI10eGF8oJ1kb/sACgjYaBhUHJwMTHwhNAQYyQwEBQzIyQwEBQzU7SjukSQIuNAw
rKhI8OxUgAWsAAv/3AAADEgPnAAwAQQAQYWFxY2NzYmJyYOASUGFhcWNjCXAwyfAgcGFB8BFjY
/ArChBhY7ATY/ASc2LwE3FRQWOWEYnJURNcclLgEnJgYCYBYXJy1TGryXJxs6MP28Fis3NmYaYJQ
DAgNlfaQFRgYMBHECHQEJB18MAykBAgVJQAKGPgcJCP6kFlssIDMDhihUGBYWKChUGA8BHBgpYSE
eCSU3/wAGBwZ1jwUMBT0EAQSCARUHCwEM+AQGB1VvdAYJCQYBGQkFySc0AQEYAAL/8AAAAxID5wA
DACoAACUXJScBJgYXPgEXHgEjLgEHIGYXHGEXJScuAQcmNjceArc3NgInLgEnJgYBI2MBjWb+EA2
eCgaHOQMBIAJcQwguOQATQAFIA2uNag8KAXt9EGAHgn8BcWIGr9vWtNMCogOVxgcNVQIaCUkqVU8
GdCSbBpU7BwQaAQQuNS8OATyABUsGAAAAAABAAAAADtWpNAEwAUACJAIAABMGJxU2Nz4BFzY
XHGEzBYPYAQ4BFBYXFjI/AT4Bjz8BNhc2FjI3PgEzNhYn4BMzYXHGEzNSImJyYHJgYiJy4BIyY
GIicImYiJgciAQcjPwEvASYHBg8BFSEVMjY3NjcyFhcWMjc2MhceATMyNzYyFyYzYmJc2NzY2Nz4
BNzUnITcXfY3NjY3JyUeATI2NCYiBkscGxscDh4OHxsKFWlGJiFMCgkJCRItEVAOBAlfExsfHDC
3HA4eDhw3NxxwOHg4gGw81EBA1DxsGzC4Gw4fDh82OBsXQDc3NiAbAgN/GTghyBALHSAIOv7oERc
LICUSJRAULxghTCILFQsUGCFNIRQYExggJQsTBww1Ezr+lXkMEhsFBAXcF/6NAS9ILy9ILwFzDQF
ZAQ4HBgEBDQUHShcgVgcVFxYHEhJTDiMTYgkNAQEbDgcFARsOBwUBDQcKWQcHDQEBGg0HBQEaDQ0
aGgEBH4asFk4JChMSINChEQQFDwEHCQkJDw8FBAkPDwkJDwEBBwkEDgszAXwFCRINfHFeE1kkLy9
ILy8ABAAAAAAD4gPnAAMABwA8AEkAACUzNSMFMzUjAQcxBxUnJgYPAQYWHwIHBg8BDgEfAR4BPwE
2PwEXFh8BFj8BNiYvASYvAQM3PgEvASYjIiUOARceATc+AScuAgMou7v82bq6ApGt15UHCgIPAgc

```

G7z5NBQOABQEEQgQNBX8EBIqLBgafCwYtAwYHkgYEQpmsBgMDHwUJBp49JxYWGfQoKBYWDzE5JJy
cnALgZHwBJgEGBjwHCwE9aiACA2sEDQVTBQIEZgMCOGAEEASICC2MHDAIpAQmxAQhkBAwFNgg3GVM
pJxcWGVMPghwBAAAF/+IAAAPrA+cACAARAC0APwBHAAABDgEiJjQ2MhY3FAYiJjQ2MhY1FTMyFhQ
GKwEVfAYiJj0BIyImNDY7ATU0NjIWBxYXFjY3Mx4BNzY3NiYnIQ4BARUzNS4BIgYC5QETHhMTHhN
0FB0UFB0U/ZBGCg0NCKYNEw5FCg0NCKUOEw3GFYd4vxR3Fb54JxdBe3v94nx7Aec+AREbEQHTDxM
THhQUZQ8UFB4TFBRFDRQNRQoNDQpFDRQNRQoNDfAnFjxjbW1jPBYNsgfBcgBr8PDCw4OAAAYAAAA
AA+ID5wALABcAIwAvAFQAXQAAAQ4BBY4BJz4BNx4BBQ4BBY4BJz4BNx4BFx4BFz4BNy4BJw4BBR4
BFz4BNy4BJw4BAQcGFRQWHwEVHgE3PgE9ATYmLwE3Fx4BOWE2NCcjJzQmJyMiBjceATI2NCYiBgO
rAlU+QVMBAVU/QFP9ugJVPkBUAQJVPj5V6AJ1V1hzAgN0V1Z1/bYcC1hYcwICdVZYcwGXqhASD60
BIBgQFAIHGJpNagRCJMiIoFXGhsDBhxAASxBLcXBLAF2QVMBAVU/QFMCA1NAQVMBAVU/QFMCA1N
AWHMCAnVWHMCAnNYVnMEAnVWHMCAnMBjp4UGBIeCFqnFhgDaxkRyA1LETpTwoJAzcefwIYAgc
yISwsQSwSAAAAAUAAAAA+ID5wBBAEWAVQCEAI0AADcyFhceARcxMzE+ATc+ATczMTIWFx4BFz4
BNz4BMhYXHgEXNSImJy4BJw4BBw4BIiYnLgEnIzUOAQcOASImJy4BJxMGfH4BNycmByIGJR4BMjY
0JiIGJQcOAQcGFj8BFwcGDwEjIgYUFhchMjY/AhcWFzM+ATQmJyMvASyXlWwEzJyYnIhceATI2NCY
iBgESGBETPDafMTwTDxgPBBMaERQ/NDRAExIZJRORFEA0FhwSEz0xMT0UER0qHBIToi4HMj8UEX0
pHRMTPzMBw8UziXOCgoOGQM+ASK/Kio/Kf451Q8YAgIkGn9IeAcEptUUGhoUAQMNfGZYgFQOEQA
UGhoUg24pAQMFAn0OEAbKASo+Kio+KqMQEHUpAgIpFREQARESFskCAigWEhAQEHUpAmISEhQoAQE
nFRISEhIUJgMBAigWEhISEhYpAQHNEYQJULxSBAEQWYApKT8qKlsdBBgQGyAEGTd4BwqbGigaAQ4
MeIBLDAEBGigaAXMiAQIDYgoBGR8qKj8pKQAAAwAAAAADCQpNAAQACQAxAAABFgIHEQMMAj8CNSM
VJgADPgEXBzEhHgEXIT4BNyE1PgEXEAInNRY2JwYmNT4BJwYmAZ5bRhwhfHfTxBh8fGf6xAwFzghb
+qA9gSwGUTWSc/o4Fqmn7HTsqARobJhECKTKDZav+qS0CNf1VMAGh0QlqEHcC/rf+nBNxhwM/XR4
mjAh2FoubAskBCQYnCCsEGggCFCMBJwoABgAAAAAD4gPnACQARwBTAF0AfACWAAATBiMVMjYyFjI
2MhYyNjIWMjYyFjM1IiYiBiImIgYiJiIGIiYiEwYiJyYnFRYXFjI2MhYyNjIWMjYzMhc1JiMiBiI
mIgYiJiIFFBYXPgE3LgEnDgElFBYXPgE0JiIGJQUOAR8CBRYXFjI2MhYyNjIWFzI3AzC+ATUuASc
iJQcOAR8CBxUWFxYyNjIWMzI3Jzc+ATQmIz4cISE5RDpCOUQ5QjLEOkE6RDkhITLEOkE6RD1COUQ
5QjpEARIpEgcHBwcSKSQqJCKkKiQpJBYPDg4PFiQpJCokKSQqAm9BNTJDAQFDMjJD/fYoIh8qKj8
qAlP+nhUfDQnc/qUODB1COUQ5QjLEOSEYFsXlGiABJhwE/dfdDRMHAjPHDgWskSMrJBQPDnuPEBQ
YEGEDV0bGxsbGxsbG10bGxsbGxsbAUICAMCOgIDCBERERERBTOfEREREU4zQwICQzMzQwICQ7k
gKgEBKkAqKkNBTAjB6nBBAUNGhoaGgEIAWcrBSQaHSYBbikDHhYEaidQAgUIEREE4RoEFiMYAAA
AAwAAAAADCAPnABIATQBbAAATDwIGFh8BFj8DNI8BJiMiEwYxByWYHwEWNj8BFwMGHwEHBhYfARY
2PwE2JyYvATcXMDExfjY/ATYmLwEmBg8BJyImLwEiMSYjJwY3BhYXFjY3NiYnJiMiBpc9AlQEAWd
UCwhhAigECEgFBgkuAUMCBAY5BgwDLCRzBAi1PwMFB1UGDAJqAwUBAlNaLDaFDQSCBQIFMAUMBFE
+AQICxAEDA4gJyRQkMDFcFxQkMBGyJT4Ba4oEawYOBQCFCoMDWwsHQgQBpgGXBGwDGQIEBmUD/vw
LCKiPBwsDJQMEB+8IBwMCTc0lKAQBBZwFDQqOBAEFYTUBAVcCDAF7MVwXFCQwMVwXCiYAAAMAAAA
AA8MD5wBRAFOAcAAANw4BBxUyNjc+ATM2Fx4BMj4CMzYXHGEzFjc+ATM2Fx4BMxY3PgEzNhceATM
1IicuASMmBw4BBYInLgEjJgcOAQciJy4BIyYHDgEHIicuASIGEx4BMjY0JiIGAQMHBgcDBh4BNj8
BJTY3EzYmJy4BBjwOHRAQHAWOHxEhHQ0dIRwcIBAhHQ4dEB8cDh8RIRwOHRAgGw4gECEdDh0QIBs
OIBAhHQ4dEB8cDh8QIhwOHRAgGw4gECEdDhwRIBcOICEfggEuQy8vQy4Cf6XyEgg6CQghIwqDAR4
SCsMMDBUOHRtsBQcBWQcFBQcBDQUHBwOHAQ0FBwENBQcBDQUHAQ0FBwENBQcdZDQUHAQ0FBwENBQc
BDQUHAQ0FBwENBQcBDQUHBwGLIS4uQy4uAa3+6ZIJE/69ESMUCRDjsQkTAU8WLAwFAQ4AAAAAAwA
AAAAD4gPnACQAMABOAAATBiMVMjYyFjI2MhYyNjIWMjYyFjM1IiYiBiImIgYiJiIGIiYiARQWFz4
BNy4BJw4BEwUOAR8CBRYXFjI+ARYyPgEWFzI3AzC+ATUuASc+HCEhOUQ6QjLEOUI5RD1COU5ISE
5RT1COUQ5QjLEOUI6RAKcQTUyQwEBQzIyQ0n+nhUfDQnc/qUODB1COUQ5QjLEOSEYFsXlGiABJhw
BhAlDgXsbGxsbGxtdGxsbGxsbGwEGM0MCAkMzM0MCAkMBHEEFMCMHqcEDBg0aARsaARsBCAFnKwU
kGh0mAQAEAAAAAONA+cAEABQAGUAKQAAJQcxBgCGHgI/ATY0LwEmIjCHfzcXBxc3FwcXNxcHFzc
2JiMuAQcOAQcnNz4BMzYmIyYGBw4BBYc3PgEXNiYjJgYVDgEXByc+ATM2JjUmBgCBHgEHDgEHDgE
nLgE3PgE3PgE3MhYFDgEHBhcPAicmIg8BBhQfARYyPwE2NC8BPwEWMzI2Nz4BNzYmJy4BIyIGAzm
fBAIPBiBgECUDA00ECVdDDgkMCwsPCg8NEQsSDJALAgEMJAiiIwMLCBQrAhUEAgshAh8cAwkBEC0
CEAICDRcoFgEBCw8vAgEJCQsDATQXEWfKSGJGDUXEWfKSIpXi0eNf7NKTiHCiYLCLoFChkJwk
JJAoZCY8JCQXAMjI9OnQyKTIHBx0hHUSqOXbgHQIDdi8qCg4iAwkDVAOVkBAUDREMDgsODgsLCg0
6Bg4MAQEKJgUMBBoUFRAFCAEMKAgKAh8eARYQBACBFjACAQwnIgwXAQ0BBAIGF0InKk8hRBgxFOI
nKk8iKCBFBIPyJRTPiUWuwUJCY8JGQokCQmPChgKBb8SHTIyKWE0NlwiHRwyAAUAAAAA+ID6QA
DAACAcWATAEWAAADchNSEnITUhJyE1ISUOAQcXNQYmAQYWFwUeARcFDgEfAR4BPwEVDgEHLgE3PgE
3NhYXNZUuAScOAQceARc+ATc1NzMWnXElLgEHIGY3fQEY/ug+ARj+6D4BN/7JA3gEfGKGKzv93zI
fBgE7WmAT/uwPDQUDbhkPbgU+LTFAAQEfGCY/GmcDZz1efAMdFf5dfwpBCS8i/ssjOQ8FBwK9Hx8
fHx85J0MCCnkIBAJFFjgDriRiK0sFGRANEBAEHworOAEbQjEcMA8WDBMeCAZVBgN9XV19AgJ9XRM
dAQ4Bfis7JAICAgAAAAAYAAAAA+ID5wAoAFEAWgBjAHcAiWAAExYXHgMyPgIyHgIyPgI3NjQnDgM

```
iLgIiDgIiLgInBicWFzIeAjI+AjIeAjI+AjM2NCciDgIiLgIiDgIiLgIjBiUeATI2NCYiBgUeATI
2NCYiBiUHBh8BFjMyNz4BLwE3NiYnJiMGJQcGHwEWMzI3PgEvATc2JicmIyIBAqSqNSs6XTksNFU
0LDpdOSw0KwsLLzorNFU0LDpdOSw0VTQsOS8LAQELKjUrO105LDRVNCw6XTksNCsLCy86KzRVNCw
6XTksNFU0LDkvCwLYAS1DLsXELf31ATBILy9IMAGwUAKKUgWTCgkNBwhHRgcIDQkJFP31VgoLWAw
VCgoOBwhMSwcJDgkJFgG4CwEBEBcSEhcQEBCSEhcQAQEWAEQESFxAQFxiSFxAQFxiBAWkLARAXEhI
XEBAXEhIXEAEWARIXEBAXEhIXEBAXEgF5IS0tQy0tHSMwMEcwMOSVEhKHEQUJHA11gg4cBwUBBJ4
TE5ASBQkeDn2KDx0JBAAGAAAAAaOBA+cAdwAoAD8AUgBiAHUAAAEGBw4BBx4BMz4BNy4BJwYlBgC
GBYMiJiMeARc+ATc+ATcmJy4BJw4BExQXHgEXHgEXFhc+ATU0Jy4BJy4BJyIFFBYXNTIWPwEuASc
mNz4BNw4BJQYHPgEXHgEXFhcUAScOAScOAhYXPG3JicmNjc+ATcOAQI6UGkuUSAzcD12wD0ZOR0
z/q5HUBcWJxUgCxVNMhdvRk+nNSonLkIOH0pWIRBCLCJBG0IuGRwID0EuJmtBM/4LBggBVEIzERU
HDAQCCARGUAHDBQEgVywoUig4Ii7PiAUT+gIPChMnMWgwGQgEAgUEEQdCcwEQMR0MDwMcHwFpWSI
0FUx2GACCAQNBbCoBDxQUXFECBUWAhclARFWVQXQFBEjFS43L2o5KywLIxIPFwG+GTEVCAUFAYB
EKEtXIjwXP6vGISoIAgUEEQ8VFXygDa1QFQ5XfJ5JCiYhRFAMTygwSxMCJQACAAAAAANuA+cACAB
OAAATHgEyNjQmIgYlASc+ATUuAScOAR0BIgYHbHqfARYyPwI2PwEWHwEWHwEOAQcRHgEyNjc1NwE
eATc+AS8BNzYmLwEmLwI3NjQnLgEjInsBNlE2NlE2AZf+kCKMDwEbFhIaDBQHDw9tESkQBGMEBUw
JEjUBBT4JIgIBGigaARcBAA0iDxQCEfQMHWckgQcIDgaqEBAHEwkSAz8pNjZRNzdx/o4nBhYQFB0
BARYQBgyJECKRbQ8PCQMBBU0SETMEBUYRPAb+7RQaHhX7JP8ADQQKDzAS8wwjWCF4BwUJA60QJw8
HCAAB//9AAAD6APnABYAJQAzAEIAUQB2AIIAADcGBw4BBxUFLgEnLgUnLgEjJgYBMhceAQcOASc
uATc+ATcHBhYXFjY3NiYnJiMOASUWFx4BBw4BJy4BNz4BNycOAcGfHCWNjc2JicmIyUHBgcGFh8
BBwYeATY/AT4BLwE3F4BHwEWNi8CNCYvASYiBjCGHgE+ASYNJiMiBpY3PQwUAQPFCLFGP0YBJkN
CHxg7Ix9DalkPDz4/DQ9mPT4/DQ10M8URWFRXiRURWFQXF0du/oAPDj1ADBB1PT8+DA1PMgNHARE
RVlVXiHURWVMXFWFGvRMGBAoMiiUEEiUdBS4CAhdQdyAFDgiIIAwfeTMSGAMDDxNVBiA9Mg4gHgc
IGIRKIBMBwN8BgVEBAImBBcQGQ8LEQEQAARMDEGQ/Pj8NEGU9NT4BaFaKFRFYVFeJFQUBVqABAw9
kPz4/DBB1PTU+AjgDVUUhIhCRWFRWihUF3XgPFxMhCnukExwJEhPOASQVRUTTCw0BHWQzCxuOAh0
HAQEEIR4zDiA9MwgBHWAAAAwAAAAA8MD5wADAACcWAPABMAFwAbAB8AIwAnAEoAVwAAJTM1Iwc
zNSMHMzUjBzMLIwcZNSMHMzUjBTMLIwUzNSMFMzUjBTMLIwEHBhYXFhcWMYUXFj4BJi8BBzCXNRU
3PgEuAQ8BLwEmIyYGNwYWFxY2NzYmJyYOAQNIHx9eHx/ZHx9dHx/aICBdHx8DCT4+/ug+Pv7KPj7
+yT4+ARiBGRktGxwJCAFnPR0zFBoc3r1DZKcbIAMkGoPdBBMTGSt9FhcnKVMZFhcnGzowJD8/Pz9
BQUFBQUBQZycnJycnJwCJN4vXxwPAQEZOAKZOjMLS11HwEBCgIlnCEBB0MBCAEYtltTGRYXJyl
TGQ8CGwAAAAAEgDeAAEAAAAAAAAAAAAAEEADwABAAEAAAAAAAAAIBwAQAAEAAAAAAAAAM
ADwAXAAEAAAAAAAAQADwAmAAEAAAAAAAAUAwA1AAEAAAAAAAAAYADwBAAEAAAAAAAAALABPAAEAAAA
AAAsAEgB7AAMAAQQAAMAAAgCNAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQA
AAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQA
AMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQQAAMAAQ
DIEFjY29yZGlvbl9JY29uc1JlZ3VsYXJBY2NvcnRpb25fSWNvbNBY2NvcnRpb25fSWNvbNwZlXJ
zaW9uIDEuMEFjY29yZGlvbl9JY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV
0cm8gU3R1ZGlvb3d3LnN5bmNmdXNpb24uY29tACAAQQBjAGMABwByAGQAaQBvAG4AXwBJAGMABwB
uAHMAUgBlAGcAdQBsAGEAcgBBAGMAYwBvAHIAZABpAG8AbgBfAEkAYwBvAG4AcwBBAGMAYwBvAHIA
ZABpAG8AbgBfAEkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAQQBjAGMABwByAGQAaQB
vAG4AXwBJAGMABwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAQZQBkACAAdQBzAGkAbgBnACA
AUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAAdAB1AGQAaQBvAHcAdwB3AC4AcwB
5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAA
AAAAAQECAQMBAEFAQYBBwEIAQkBCgELAQwBDQEQAQ8BEAERARIBEWUARUBFgEXARGBGQEaAA1
KYYZlBgluX1Rocm93DfRhYmxlX1Rlbn5pcwtDeWNsaW5nX0JNwApXYXRlc1l9Qb2xvCFN3aW1taW5
nDERpc2Nlc1l9UaHJvdwZCb3hpbmcGUm93aW5nCUhpZ2hfSnVtcAxJbmRvb3JfR2FtZXMKQ3ljbGV
fUmFjZQtXYXRlc1l9HYW1lcwDfTYWlsaW5nEU1hcmF0aG9uX1N3aW1taW5nCE1hcmF0aG9uBkRpdml
uZw1Td21tbWluZzEJQmFkbWludG9uBlJhY2luZxVTEw5jaHJvbml6ZWRFU3dpcW1pbmcLVm9sbGV
5X0JhbGwJQXRobGV0aWNzFEN5Y2xpbmdfTW91bnRhaW5CaWtlCUxvbmddSnVtcAAAAA==)
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 EJ2 JavaScript 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.TS

```
import {Accordion} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let accordion: Accordion = new 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.' }
    ]
});
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>
<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 EJ2 JavaScript 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.TS

```
import {Accordion} from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let expandIndex;
let isCollapsed = false;
let initialLoad = true;
let accordion: Accordion = new Accordion({
    expandMode: 'Single',
    expanding: onExpanding,
    expanded: onExpanded,
    created: onCreate,
    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. 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.' }
    ]
});
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 EJ2 JavaScript 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.TS

```
import { Accordion, TreeView } from '@syncfusion/ej2-navigations';
```

```
import { DocDB, DownloadDB, PicDB } from 'datasource.ts'
//Initialize Accordion component
let acrdnObj: Accordion = new 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
acrdnObj.appendTo('#element');
//Expanding Event function for Accordion component.
function expand(e: ExpandEventArgs): void {
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 0 &&
e.element.querySelector('#treeDoc').childElementCount === 0) {
        //Initialize TreeView component
        let treeObj: TreeView = new TreeView({
            fields: { dataSource: DocDB, id: 'nodeId', text: 'nodeText', child:
'nodeChild', iconCss: 'icon', imageUrl: 'image' },
            sortOrder: 'Ascending'
        });
        //Render initialized TreeView component
        treeObj.appendTo('#treeDoc');
    }
    if (e.name === 'expanding' && [].indexOf.call(this.items, e.item) === 1
&& e.element.querySelector('#treeDownload').childElementCount === 0) {
        let treeObj: TreeView = new 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: TreeView = new 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>
```

Ej1 api migration in EJ2 JavaScript Accordion control

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Keyboard Navigation | **Property:** *allowKeyboardNavigation*

 \$("#accordion").ejAccordion({
 allowKeyboardNavigation: false
 });
 | **Not Applicable** |

| Localization | **Not Applicable** | **Property:** *locale*

 var accordion = new ej.navigations.Accordion({
 locale: 'en-US'
 });
 accordion.appendTo('#ej2Accordion');
 |

| RTL | **Property:** *enableRTL*

 \$("#accordion").ejAccordion({
 enableRTL: false
 });
 | **Property:** *enableRtl*

 var accordion = new ej.navigations.Accordion({
 enableRtl: false
 });
 accordion.appendTo('#ej2Accordion');
 |

AjaxSettings

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Default | **Property:** *ajaxSettings*

 \$("#accordion").ejAccordion({
 ajaxSettings: { type: 'GET' }
 });
 | **Not Applicable** |

| Asynchronous | **Property:** *ajaxSettings.async*

 \$("#accordion").ejAccordion({
 ajaxSettings: { async: true }
 });
 | **Not Applicable** |

| Browser Cache | **Property:** *ajaxSettings.cache*

 \$(" #accordion").ejAccordion({
 ajaxSettings: { cache: false }
 });
 | **Not Applicable** |

| Request type | **Property:** *ajaxSettings.contentType*

 \$(" #accordion").ejAccordion({
 ajaxSettings: { contentType: "html" }
 });
 | **Not Applicable** |

| Data | **Property:** *ajaxSettings.data*

 \$(" #accordion").ejAccordion({
 ajaxSettings: { data: {} }
 });
 | **Not Applicable** |

| Response type | **Property:** *ajaxSettings.dataType*

 \$(" #accordion").ejAccordion({
 ajaxSettings: { dataType: "html" }
 });
 | **Not Applicable** |

| HTTP request type | **Property:** *ajaxSettings.type*

 \$(" #accordion").ejAccordion({
 ajaxSettings: { type: 'GET' }
 });
 | **Not Applicable** |

| AjaxBeforeLoad | **Event:** *ajaxBeforeLoad*

 \$(" #accordion").ejAccordion({
 ajaxBeforeLoad: function(args) {}
 });
 | **Not Applicable** |

| AjaxError | **Event:** *ajaxError*

 \$(" #accordion").ejAccordion({
 ajaxError: function(args) {}
 });
 | **Not Applicable** |

| AjaxLoad | **Event:** *ajaxLoad*

 \$(" #accordion").ejAccordion({
 ajaxLoad: function(args) {}
 });
 | **Not Applicable** |

| AjaxSuccess | **Event:** *ajaxSuccess*

 \$(" #accordion").ejAccordion({
 ajaxSuccess: function(args) {}
 });
 | **Not Applicable** |

Animation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Default | **Not Applicable** | **Property:** *animation*

 var accordion = new ej.navigations.Accordion({
 animation: { expand: {} , collapse: {} }
 });
 accordion.appendTo('#ej2Accordion');
 |

| EnableAnimation | **Property:** *enableAnimation*

 \$(" #accordion").ejAccordion({
 enableAnimation: false
 });
 | **Not Applicable** |

| Expand animation | **Not Applicable** | **Property:** *animation.expand*

 var accordion = new ej.navigations.Accordion({
 animation: {
 expand: {
 duration: 400,
 easing: 'ease-in',
 effect: 'SlideLeft'
 }
 }
 });
 accordion.appendTo('#ej2Accordion');
 |

| Collapse animation | **Not Applicable** | **Property:** *animation.collapse*

 var accordion = new ej.navigations.Accordion({
 animation: {
 collapse: {
 duration: 400,
 easing: 'ease-in',
 effect: 'SlideLeft'
 }
 }
 });
 accordion.appendTo('#ej2Accordion');
 |

| Duration
 [expand / collapse] | **Not Applicable** | **Property:** *animation.collapse.duration*

 var accordion = new ej.navigations.Accordion({
 animation: {
 expand: { duration: 400 }
 }
 });
 accordion.appendTo('#ej2Accordion');
 |

| Easing
 [expand / collapse] | **Not Applicable** | **Property:** *animation.collapse.easing*

var accordion = new ej.navigations.Accordion({
 animation: {

expand: { easing: 'ease-in' }
 }
 });
 accordion.appendTo('#ej2Accordion');

|

| Effect
 [expand / collapse] | **Not Applicable** | **Property:** *animation.collapse.effect*

var accordion = new ej.navigations.Accordion({
 animation: {

expand: { effect: 'SlideLeft' }
 }
 });
 accordion.appendTo('#ej2Accordion');

</> |

Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Default | **Not Applicable** | **Property:** *items*

 var accordion = new ej.navigations.Accordion({

 items: []
 });
 accordion.appendTo('#ej2Accordion');
 |

| Content | **Not Applicable** | **Property:** *items[0].content*

 var accordion = new
ej.navigations.Accordion({
 items: [{ content: 'Contents' }]
 });

accordion.appendTo('#ej2Accordion');
 |

| Custom class | **Not Applicable** | **Property:** *items[0].cssClass*

 var accordion = new
ej.navigations.Accordion({
 items: [{ cssClass: 'customClass' }]
 });

accordion.appendTo('#ej2Accordion');
 |

| Header | **Not Applicable** | **Property:** *items[0].header*

 var accordion = new
ej.navigations.Accordion({
 items: [{ header: 'Header1' }]
 });

accordion.appendTo('#ej2Accordion');
 |

| HeaderSize | **Property:** *headerSize*

 \$("#accordion").ejAccordion({

headerSize: '50px'
 });
 | **Not Applicable** |

| Icon class | **Not Applicable** | **Property:** *items[0].iconCss*

 var accordion = new
ej.navigations.Accordion({
 items: [{ iconCss: 'e-icons' }]
 });

accordion.appendTo('#ej2Accordion');
 |

| IsExpand | **Not Applicable** | **Property:** *items[0].expanded*

 var accordion = new
ej.navigations.Accordion({
 items: [{ expanded: true }]
 });

accordion.appendTo('#ej2Accordion');
 |

| Collapse Item | **Method:** *collapsePanel(index)*

 \$("#accordion").ejTab();
 var
accordion = \$("#accordion").data("ejAccordion");
 accordion.collapsePanel(0);
 | **Method:**
expandItem(index, false)

 var accordion = new ej.navigations.Accordion();

accordion.appendTo('#ej2Accordion');
 accordion.expandItem(0, false);
 |

| Expand Item | **Method:** *expandPanel(index)*

 \$("#accordion").ejTab();
 var accordion
= \$("#accordion").data("ejAccordion");
 accordion.expandPanel(0);
 | **Method:**
expandItem(index, true)

 var accordion = new ej.navigations.Accordion();

accordion.appendTo('#ej2Accordion');
 accordion.expandItem(0, true);
 |

| CollapseAll | **Method:** *collapseAll()*

 \$("#accordion").ejTab();
 var accordion =
\$("#accordion").data("ejAccordion");
 accordion.collapseAll();
 | **Not Applicable** |

| ExpandAll | **Method:** *expandAll()* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.expandAll();
 | Not Applicable |`

| Get ItemsCount | **Method:** *getItemsCount()* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.getItemsCount();
 | Not Applicable |`

| AddItem | **Method:** *addItem(text, content, index)* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.addItem("New item", "The accordion content", 2);
 | Method: addItem(items, index)

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.addItem({ header: 'App', content: 'text' }, 0)
 |`

| Remove Item | **Method:** *removeItem(index)* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.removeItem(0);
 | Method: removeItem(index)

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.removeItem(0)
 |`

| Disable Items | **Property:** *disableItems* `

 $("#accordion").ejTab({
 disabledItems: [0, 1]
 });
 | Not Applicable |`

| Enable Items | **Property:** *enableItems* `

 $("#accordion").ejTab({
 enableItems: [0, 1]
 });
 | Not Applicable |`

| Disable Item | **Method:** *disableItems([index])* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.disableItems([1]);
 | Method: enableItem(index, false)

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.enableItem(0, false)
 |`

| Enable Item | **Method:** *enableItems([index])* `

 $("#accordion").ejTab();
 var accordion = $("#accordion").data("ejAccordion");
 accordion.enableItems([1]);
 | Method: enableItem(index, true)

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.enableItem(0, true)
 |`

| Hide Item | **Not Applicable** | **Method:** *hideItem(index, true)* `

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.hideItem(0, true)
 |`

| SelectedItemIndex | **Property:** *selectedItemIndex* `

 $("#accordion").ejAccordion({
 selectedItemIndex: false
 });
 | Not Applicable |`

| Select | **Not Applicable** | **Method:** *select(index)* `

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.select(0)
 /> |`

| BeforeActivate | **Event:** *beforeActivate* `

 $("#accordion").ejAccordion({
 beforeActivate: function(args) {}
 });
 | Event: expanding

 var accordion = new ej.navigations.Accordion({
 expanding: function(e: Event): void {}
 });
 accordion.appendTo('#ej2Accordion');
 |`

| Activate | **Event:** *activate* `

 $("#accordion").ejAccordion({
 activate: function(args) {}
 });
 | Event: expanded

 var accordion = new ej.navigations.Accordion({
 expanded: function(e: Event): void {}
 });
 accordion.appendTo('#ej2Accordion');
 |`

| beforeInActivate | **Event:** *beforeInactivate*

 \$("#accordion").ejAccordion({

beforeInactivate: function(args) {}
 });
 | **Not Applicable** |

| InActive | **Event:** *inActivate*

 \$("#accordion").ejAccordion({
 inActivate:
function(args) {}
 });
 | **Not Applicable** |

| Clicked | **Not Applicable** | **Event:** *clicked*

 var accordion = new ej.navigations.Accordion({

 clicked: function(e: Event): void {}
 });
 accordion.appendTo('#ej2Accordion');

 |

Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Collapsible | **Property:** *collapsible*

 \$("#accordion").ejAccordion({

collapsible: false
 });
 | **Not Applicable** |

| Collapse speed | **Property:** *collapseSpeed*

 \$("#accordion").ejAccordion({

collapseSpeed: 500
 });
 | **Not Applicable** |

| Custom class | **Property:** *cssClass*

 \$("#accordion").ejAccordion({
 cssClass:
'custom'
 });
 | **Not Applicable** |

| CustomIcon class | **Property:** *customIcon*

 \$("#accordion").ejAccordion({

customIcon: {
 header: "header-close",
 selectedHeader:
"header-expand"
 }
 });
 | **Not Applicable** |

| Enabled | **Property:** *enabled*

 \$("#accordion").ejAccordion({
 enabled: false

 });
 | **Not Applicable** |

| Events | **Property:** *events*

 \$("#accordion").ejAccordion({
 events:
"mouseover"
 });
 | **Not Applicable** |

| Expand speed | **Property:** *expandSpeed*

 \$("#accordion").ejAccordion({

expandSpeed: 300
 });
 | **Not Applicable** |

| Height | **Property:** *height*

 \$("#accordion").ejAccordion({
 height: 400

});
 | **Property:** *height*

 var accordion = new ej.navigations.Accordion({

height: 700
 });
 accordion.appendTo('#ej2Accordion');
 |

| HeightAdjustMode | **Property:** *heightAdjustMode*

 \$("#accordion").ejAccordion({

 heightAdjustMode: "content"
 });
 | **Not Applicable** |

| HtmlAttributes | **Property:** *htmlAttributes*

 \$("#accordion").ejAccordion({

htmlAttributes: { title: "Demo" }
 });
 | **Not Applicable** |

| MultipleOpen | **Property:** *enableMultipleOpen*

 \$("#accordion").ejAccordion({

 enableMultipleOpen: true
 });
 | **Property:** *expandMode*

 var accordion =
new ej.navigations.Accordion({
 expandMode: 'Multiple'
 });

accordion.appendTo('#ej2Accordion');
 |

| Persistence | **Property:** *enablePersistence*

 \$("#accordion").ejAccordion({

enablePersistence: false
 });
 | **Property:** *enablePersistence*

 var accordion = new
ej.navigations.Accordion({
 enablePersistence: true
 });

accordion.appendTo('#ej2Accordion');
 |

| ShowRoundedCorner | **Property:** *showRoundedCorner*

 \$("#accordion").ejAccordion({
 showRoundedCorner: false
 });
 | **Not Applicable** |

| Width | **Property:** *width*

 \$("#accordion").ejAccordion({
 width: 600
 });
 | **Property:** *width*

 var accordion = new ej.navigations.Accordion({
 width: 500
 });
 accordion.appendTo('#ej2Accordion');
 |

| Enable | **Method:** *enable()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.enable();
 | **Not Applicable** |

| Disable | **Method:** *disable()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.disable();
 | **Not Applicable** |

| Show | **Method:** *show()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.show();
 | **Not Applicable** |

| Hide | **Method:** *hide()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.hide();
 | **Not Applicable** |

| Destroy | **Method:** *destroy()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.destroy();
 | **Method:** *destroy()*

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.destroy();
 |

| Refresh | **Method:** *refresh()*

 \$("#accordion").ejAccordion();
 var accordion = \$("#accordion").data("ejAccordion");
 accordion.refresh();
 | **Method:** *refresh()*

 var accordion = new ej.navigations.Accordion();
 accordion.appendTo('#ej2Accordion');
 accordion.refresh();
 |

| Created | **Event:** *create*

 \$("#accordion").ejAccordion({
 create: function(args) { }
 });
 | **Event:** *created*

 var accordion = new ej.navigations.Accordion({
 created: function(e: Event): void { }
 });
 accordion.appendTo('#ej2Accordion');
 |

| Destroyed | **Event:** *destroy*

 \$("#accordion").ejAccordion({
 destroy: function(args) { }
 });
 | **Event:** *destroyed*

 var accordion = new ej.navigations.Accordion({
 destroyed: function(e: Event): void { }
 });
 accordion.appendTo('#ej2Accordion');
 |

AccumulationChart

<!-- markdownlint-disable MD036 -->

Getting started in EJ2 JavaScript Accumulation chart control

In EJ2, accumulation chart is implemented as a separate control to avoid the axis related logics. The dependencies for accumulation chart is same as chart control.

Dependencies

The list of minimum dependencies required to use an accumulation chart are follows:

```
`javascript
```

```
|-- @syncfusion/ej2-charts
```

```
|-- @syncfusion/ej2-base
```

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

Installation and Configuration

- To get started with accumulation chart 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
\
```

- Map the Accumulation chart packages in system.config.js configuration file.

```
`javascript
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-charts": "syncfusion:ej2-charts/dist/ej2-charts.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-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",
    "@syncfusion/ej2-navigations": "syncfusion:ej2-navigations/dist/ej2-navigations.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-inputs": "syncfusion:ej2-inputs/dist/ej2-inputs.umd.min.js",
"@syncfusion/ej2-svg-base": "syncfusion:ej2-svg-base/dist/ej2-svg-base.umd.min.js",
"@syncfusion/ej2-splitbuttons": "syncfusion:ej2-splitbuttons/dist/ej2-splitbuttons.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 Accumulation Chart to the Project](#)

Add HTML div elements in your `index.html`. [src/index.html] for accumulation chart.

```

`html
<!DOCTYPE html>
<html lang="en">
<head>
<title>EJ2 Animation</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-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 Accumulation chart-->
<div id='element'>
</div>
</body>

```

```
</html>
```

Now, import an accumulation chart component in your `app.ts` to instantiate the accumulation chart, and append the accumulation chart instance to the `#element` [src/app/app.ts]

```
`javascript
import { AccumulationChart } from '@syncfusion/ej2-charts';
// initialize Accumulation Chart component
let chart: AccumulationChart = new AccumulationChart();
// render initialized Accumulation Chart
chart.appendTo('#element');
```

Pie Series

By default, the pie series will be rendered when assigning the JSON data to the series using the [dataSource](#) property. Map the field names in the JSON data to the [xName](#) and [yName](#) properties of the series.

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: 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 }],
      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>

```

Now, use the `npm run start` command to run the application in the browser.

`npm run start`

Pie dough nut in EJ2 JavaScript 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.TS

```

import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let piechart: AccumulationChart = new 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 }],
            type: 'Pie',
            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>

```

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.TS

```

import { AccumulationChart } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: data,
            radius: '100%',
            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>

```

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.TS

```

import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let piechart: AccumulationChart = new 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: '60%', y: '60%' },
    enableSmartLabels: true,
    enableAnimation: false,
    legendSettings: { visible: false },
    // Initialize the tooltip
    tooltip: { enable: true, format: '$ {point.x} : <b>${point.y}%</b>'
},
    title: 'Mobile Browser 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>
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.TS

```

import { AccumulationChart, AccumulationLegend, PieSeries,
AccumulationDataLabel } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationChart, AccumulationLegend, PieSeries,
AccumulationDataLabel );
let piechart: AccumulationChart = new 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>
  <script>
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.TS

```

import { AccumulationChart } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: data,
      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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
import { dataMapping } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: dataMapping,
      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.TS

```
import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
import { dataMapping } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: dataMapping,
      //Map the fill color from data source
      pointColorMapping: 'fill',
      //Map the fill color from data source
      name: '',
      xName: 'x',
      yName: 'y',
      dataLabel: { visible: true, name: 'text' }
    }
  ]
});
```

```

    ]
  }, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccPointRenderEventArgs
} from '@syncfusion/ej2-charts';
import { dataMapping } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: dataMapping,
      xName: 'x',
      yName: 'y',
      border: { color: 'white', width: 1 },
    }
  ],
  pointRender: (args: IAccPointRenderEventArgs) => {
    if ((args.point.x as string).indexOf('Apr') > -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>

```

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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccPointRenderEventArgs
} from '@syncfusion/ej2-charts';
import { dataMapping } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new 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.TS

```
import {
  AccumulationTheme, getElement, AccumulationChart, AccumulationLegend,
  PieSeries,
  AccumulationTooltip, IAccLoadedEventArgs, IAccTextRenderEventArgs,
  chartMouseClicked, IMouseEventArgs, Index, indexFinder,
  AccumulationDataLabel,
  AccumulationChartModel, AccumulationAnnotation
} from '@syncfusion/ej2-charts';
import { EmitType } from '@syncfusion/ej2-base';
AccumulationChart.Inject(AccumulationLegend, PieSeries, AccumulationTooltip,
  AccumulationDataLabel, AccumulationAnnotation);
/**
 * Sample fro Drill Down in Pie chart
```

```


*/
//tslint:disable:max-unc-body-length
let suvs: Object = [{ x: 'Toyota', y: 8 }, { x: 'Ford', y: 12 }, { x: 'GM', y: 17 }, { x: 'Renault', y: 6 }, { x: 'Fiat', y: 3 }, { x: 'Hyundai', y: 16 }, { x: 'Honda', y: 8 }, { x: 'Maruthi', y: 10 }, { x: 'BMW', y: 20 }];
let cars: Object = [{ x: 'Toyota', y: 7 }, { x: 'Chrysler', y: 12 }, { x: 'Nissan', y: 9 }, { x: 'Ford', y: 15 }, { x: 'Tata', y: 10 }, { x: 'Mahindra', y: 7 }, { x: 'Renault', y: 8 }, { x: 'Skoda', y: 5 }, { x: 'Volkswagen', y: 15 }, { x: 'Fiat', y: 3 }];
let pickups: Object = [{ x: 'Nissan', y: 9 }, { x: 'Chrysler', y: 4 }, { x: 'Ford', y: 7 }, { x: 'Toyota', y: 20 }, { x: 'Suzuki', y: 13 }, { x: 'Lada', y: 12 }, { x: 'Bentley', y: 6 }, { x: 'Volvo', y: 10 }, { x: 'Audi', y: 19 }];
let minivans: Object = [{ x: 'Hummer', y: 11 }, { x: 'Ford', y: 5 }, { x: 'GM', y: 12 }, { x: 'Chrysler', y: 3 }, { x: 'Jaguar', y: 9 }, { x: 'Fiat', y: 8 }, { x: 'Honda', y: 15 }, { x: 'Hyundai', y: 4 }, { x: 'Scion', y: 11 }, { x: 'Toyota', y: 17 }];
let clickInstance: AccumulationChartModel = {
  series: [{
    type: 'Pie', dataSource: suvs, xName: 'x', yName: 'y',
    dataLabel: { visible: true, position: 'Outside' }, innerRadius:
'30%',
  }],
  textRender: (args: IAccTextRenderEventArgs) => {
    args.text = args.point.x + ' ' + args.point.y + ' %';
  },
  annotations: [{
    content: '<div id="back" style="cursor:pointer;padding:3px;width:30px; height:30px;">' +
'',
    region: 'Series', x: '50%', y: '50%'
  }],
  chartMouseClick: (args: IMouseEventArgs) => {
    if (args.target.indexOf('back') > -1) {
      let tooltip: Element = document.getElementsByClassName('e-tooltip-wrap')[0];
      if (tooltip) { tooltip.remove(); }
      pie.destroy(); pie.removeSvg(); pie = null; pie = new AccumulationChart(instance);
      pie.appendTo('#container');
      (getElement('category') as HTMLButtonElement).style.visibility = 'hidden';
      (getElement('symbol') as HTMLElement).style.visibility = 'hidden';
      (<HTMLElement>getElement('text')).style.visibility = 'hidden';
    }
  },
  legendSettings: { visible: false }, enableSmartLabels: true,
  tooltip: { enable: false, format: '$ {point.x} <br> $ {point.y} %' },
};
let pointClick: EmitType<IMouseEventArgs> = (args: IMouseEventArgs) => {
  let index: Index = indexFinder(args.target);
  if (getElement(pie.element.id + '_Series_' + index.series + '_Point_' + index.point)) {
    let tooltip: Element = document.getElementsByClassName('e-tooltip-wrap')[0];
    if (tooltip) { tooltip.remove(); }
    pie.destroy(); pie.removeSvg(); pie = null;
  }
}


```

```

        switch (index.point) {
            case 0:
                clickInstance.series[0].dataSource = suvs;
                getElement('text').innerHTML = 'SUV';
                clickInstance.title = 'Automobile Sales in the SUV Segment';
                break;
            case 1:
                clickInstance.series[0].dataSource = cars;
                getElement('text').innerHTML = 'Car';
                clickInstance.title = 'Automobile Sales in the Car Segment';
                break;
            case 2:
                clickInstance.series[0].dataSource = pickups;
                getElement('text').innerHTML = 'Pickup';
                clickInstance.title = 'Automobile Sales in the Pickup
Segment';
                break;
            case 3:
                clickInstance.series[0].dataSource = minivans;
                getElement('text').innerHTML = 'Minivan';
                clickInstance.title = 'Automobile Sales in the Minivan
Segment';
                break;
        }
        pie = new AccumulationChart(clickInstance);
        pie.appendTo('#container');
        (<HTMLElement>getElement('symbol')).style.visibility = 'visible';
        (<HTMLElement>getElement('category')).style.visibility = 'visible';
        (<HTMLElement>getElement('text')).style.visibility = 'visible';
    }
};
let instance: AccumulationChartModel = {
    series: [
        {
            dataSource: [{ x: 'SUV', y: 25 }, { x: 'Car', y: 37 }, { x:
'Pickup', y: 15 }, { x: 'Minivan', y: 23 }],
            dataLabel: {
                visible: true, position: 'Inside', connectorStyle: { type:
'Curve', length: '10%' },
                font: { fontWeight: '600', color: 'white' }
            },
            radius: '70%', xName: 'x', yName: 'y', startAngle: 0, endAngle:
360, innerRadius: '0%',
            explode: false
        }
    ], enableSmartLabels: false, legendSettings: { visible: false },
    chartMouseClick: pointClick,
    textRender: (args: IAccTextRenderEventArgs) => { args.text =
args.point.x + ' ' + args.point.y + ' %'; },
    tooltip: { enable: false, format: '${point.x} <br> ${point.y} %' },
    title: 'Automobile Sales by Category',
    load: (args: IAccLoadedEventArgs) => {
        let selectedTheme: string = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.accumulation.theme =
<AccumulationTheme>(selectedTheme.charAt(0).toUpperCase() +

```

```

        selectedTheme.slice(1)).replace(/-dark/i,
'Dark').replace(/contrast/i, 'Contrast');
    }
};
let pie: AccumulationChart = new AccumulationChart(instance);
pie.appendTo('#container');
(getElement('category') as HTMLElement).onclick = (e: MouseEvent) => {
    let tooltip: Element = document.getElementsByClassName('e-tooltip-
wrap')[0];
    if (tooltip) { tooltip.remove(); }
    pie.destroy(); pie.removeSvg(); pie = null; pie = new
AccumulationChart(instance);
    pie.appendTo('#container');
    (e.target as HTMLButtonElement).style.visibility = 'hidden';
    (getElement('symbol') as HTMLElement).style.visibility = 'hidden';
    (getElement('text') as HTMLElement).style.visibility = 'hidden';
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta 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 EJ2 JavaScript 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.TS

```

import { AccumulationChart, PyramidSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PyramidSeries);
let chart: AccumulationChart = new AccumulationChart({
    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',
    }],
}, '#element');

```

INDEX.HTML

```

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { AccumulationChart, PyramidSeries } from '@syncfusion/ej2-charts';
import { pyramidData } from './datasource.ts';
AccumulationChart.Inject(PyramidSeries);
let pyramidchart: AccumulationChart = new AccumulationChart({
    series: [{
        type: 'Pyramid',
        dataSource: pyramidData,
        xName: 'x', yName: 'y',
        pyramidMode: 'Surface',
    }],
}, '#element');
```

INDEX.HTML

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

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

    <div id="container">
        <div id="element"></div>
    </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 pyramid chart can be customized by using the **width** and **height** properties.

INDEX.TS

```
import { AccumulationChart, PyramidSeries } from '@syncfusion/ej2-charts';
import { pyramidData } from './datasource.ts';
AccumulationChart.Inject(PyramidSeries);
let chart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Pyramid',
    dataSource: pyramidData,
    xName: 'x', yName: 'y',
    //Width of the pyramid will be 100% of the chart area
    width: '60%',
    //Height of the pyramid will be 100% of the chart area
    height: '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>
  <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.TS

```
import { AccumulationChart, PyramidSeries } from '@syncfusion/ej2-charts';
import { pyramidData } from './datasource.ts';
AccumulationChart.Inject(PyramidSeries);
let pyramidchart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Pyramid',
    dataSource: pyramidData,
    xName: 'x', yName: 'y',
    // Defines the gap to be left between the segments
    gapRatio: 0.2,
  }],
}, '#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>
```

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.TS


```
import { AccumulationChart, PyramidSeries, IAccPointRenderEventArgs } from
'@syncfusion/ej2-charts';
import { pyramidData } from './datasource.ts';
AccumulationChart.Inject(PyramidSeries);
let pyramidchart: AccumulationChart = new AccumulationChart({
    series: [{
        type: 'Pyramid',
        dataSource: pyramidData,
        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
    }]
});
pyramidchart.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>
```

Customization

Individual points can be customized using the `pointRender` event.

INDEX.TS

```
import { AccumulationChart, PyramidSeries, IAccPointRenderEventArgs } from
'@syncfusion/ej2-charts';
import { pyramidData } from './datasource.ts';
AccumulationChart.Inject(PyramidSeries);
let pyramidchart: AccumulationChart = new AccumulationChart({
    series: [{
        type: 'Pyramid',
        dataSource: pyramidData,
        xName: 'x', yName: 'y',
        gapRatio: 0.04
    }],
    pointRender: (args: IAccPointRenderEventArgs) => {
        if ((args.point.x as string).indexOf('India') > -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](#)

Funnel in EJ2 JavaScript 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.TS

```
import { AccumulationChart, FunnelSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(FunnelSeries);
let chart: AccumulationChart = new 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',
    }],
}, '#element');
```

INDEX.HTML

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { AccumulationChart, FunnelSeries } from '@syncfusion/ej2-charts';
import { funnelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries);
let chart: AccumulationChart = new AccumulationChart({
    series: [{
        type: 'Funnel',
        dataSource: funnelData,
        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%',
    }],
}, '#element');
```

INDEX.HTML

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { AccumulationChart, FunnelSeries } from '@syncfusion/ej2-charts';
```

```
import { funnelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries);
let chart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: funnelData,
    xName: 'x', yName: 'y',
    //Width of the neck will be set as 25% of the chart area
    neckWidth: '25%', neckHeight: '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>
```

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.TS

```
import { AccumulationChart, FunnelSeries } from '@syncfusion/ej2-charts';
import { funnelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries);
let pyramidchart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: funnelData,
```

```

        xName: 'x', yName: 'y',
        // Defines the gap to be left between the segments
        gapRatio: 0.08,
    }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { AccumulationChart, FunnelSeries, IAccPointRenderEventArgs } from
 '@syncfusion/ej2-charts';
import { funnelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries);
let funnelChart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: funnelData,
    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:3
    }
});
funnelChart.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>

```

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.TS

```

import {
  AccumulationChart, AccumulationLegend, FunnelSeries,
  AccumulationTooltip, IAccLoadedEventArgs,
  AccumulationDataLabel, IAccResizeEventArgs, AccumulationTheme
} from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationLegend, FunnelSeries,
  AccumulationTooltip, AccumulationDataLabel);
let data: object[] = [
  { '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' }];
let chart: AccumulationChart = new AccumulationChart({
    //Initializing Chart Series
    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: (args: IAccLoadedEventArgs) => {
        let selectedTheme: string = location.hash.split('/')[1];
        selectedTheme = selectedTheme ? selectedTheme : 'Material';
        args.accumulation.theme =
        <AccumulationTheme>(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: (args: IAccResizeEventArgs) => {
        let bounds: ClientRect =
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',

```



```
});
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>
```

Customization

Individual points can be customized using the `pointRender` event.

INDEX.TS

```
import { AccumulationChart, FunnelSeries, IAccPointRenderEventArgs } from
'@syncfusion/ej2-charts';
import { funnelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries);
let funnelChart: AccumulationChart = new AccumulationChart({
  series: [{
    type: 'Funnel',
    dataSource: funnelData,
    xName: 'x', yName: 'y', gapRatio: 0.08
  }],
  pointRender: (args: IAccPointRenderEventArgs) => {
    if ((args.point.x as string).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 EJ2 JavaScript Accumulation chart control

Data label can be added to a chart series by enabling the [visible](#)

option in the dataLabel property.

INDEX.TS

```
import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [{ x: 'Jan', y: 13, text: 'Jan: 3' }, { x: 'Feb', y:
13.5, text: 'Feb: 3.5' },
```

```

        { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' }]],
        xName: 'x',
        yName: 'y',
        dataLabel: { 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>

```

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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: labelData,
      xName: 'x',
      yName: 'y',

```

```

        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>

```

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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new 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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  enableSmartLabels: true,
  series: [

```

```

    {
        dataSource: labelData,
        dataLabel: { visible: true, name: 'text', position: 'Inside',
            template: "<div id='templateWrap' style='background-color:#bd18f9;border-radius: 3px; float: right;padding: 2px;line-height: 20px;text-align: center;'>" + "<img src='https://ej2.syncfusion.com/demos/src/chart/images/sunny.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>${point.y}</span></div></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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({

```

```

enableSmartLabels: true,
series: [
  {
    dataSource: labelData,
    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.TS

```
import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-charts';
import { dataMapping } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: dataMapping,
            xName: 'x',
            yName: 'y',
            dataLabel: { visible: true, name: 'text' }
        }
    ]
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS


```
import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs }
from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'Jan', y: 13, text: 'Jan: 3' },
                { x: 'Feb', y: 13, text: 'Feb: 3.5' },
                { x: 'Mar', y: 7, text: 'Mar: 7' },
                { x: 'Apr', y: 13, text: 'Apr: 13.5' } ],
            xName: 'x',
            yName: 'y',
            dataLabel: { visible: true, format: 'n2' }
        }
    ],
}, '#element');
```

INDEX.HTML

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { AccumulationChart, FunnelSeries, IAccTextRenderEventArgs,
AccumulationDataLabel } from '@syncfusion/ej2-charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(FunnelSeries, AccumulationDataLabel);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            dataLabel: { visible: true, name: 'text', position: 'Outside' },
            xName: 'x',
            yName: 'y'
        }
    ],
    textRender: (args: IAccTextRenderEventArgs) => {
        if (args.text.indexOf('Mar') > -1) {
            args.color = 'red';
            args.border.width = 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>
<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.TS

```

import { AccumulationChart, AccumulationDataLabel } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new 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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs }
from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [{ x: 'Jan', y: 13, text: 'Jan: 3' }, { x: 'Feb', y:
13.5, text: 'Feb: 3.5' },
      { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' } ],
      xName: 'x',
      yName: 'y',
      dataLabel: { visible: true }
    }
  ],
  textRender: (args: IAccTextRenderEventArgs) => {
    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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs }
from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new AccumulationChart({
  enableSmartLabels: true,
  series: [
    {
      dataSource: [{ x: 'Jan', y: 13, text: 'Jan: 3' }, { x: 'Feb', y:
13.5, text: 'Feb: 3.5' },
      { x: 'Mar', y: 7, text: 'Mar: 7' }, { x: 'Apr', y: 13.5,
text: 'Apr: 13.5' }],
      xName: 'x',
      yName: 'y',
      dataLabel: { visible: true, template: "<div
id='dataLabelTemplate'>${point.percentage}%</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>

```

Grouping in EJ2 JavaScript 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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs,
IAccPointRenderEventArgs } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationDataLabel);
let piechart: AccumulationChart = new 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'
    }
],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs,
IAccPointRenderEventArgs } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);

```

```

let accChart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: data,
      dataLabel: { visible: true, position: 'Outside' },
      groupTo: '4',
      groupMode: 'Point',
      xName: 'x',
      yName: 'y'
    }
  ],
  textRender: (args: IAccTextRenderEventArgs) => {
    if (args.text.indexOf('Others') > -1) {
      args.text = 'Grouped Slices';
      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>

```


Customization

You can customize the grouped point and its data label using `pointRender` and `textRender` event.

INDEX.TS

```
import { AccumulationChart, AccumulationDataLabel, IAccTextRenderEventArgs,
IAccPointRenderEventArgs } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
AccumulationChart.Inject(AccumulationDataLabel);
let accChart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: data,
      dataLabel: { visible: true, position: 'Outside' },
      groupTo: '11',
      xName: 'x',
      yName: 'y'
    }
  ],
  textRender: (args: IAccTextRenderEventArgs) => {
    if (args.text.indexOf('Others') > -1) {
      args.text = 'Grouped Slices';
      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>

```

Empty points in EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```

import { AccumulationChart, AccumulationAnnotation } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationAnnotation);
let piechart: AccumulationChart = new 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'
    }
  ],
  annotations:[{
    content:'<div style="border: 1px solid black;background-color:#f5f5f5; padding: 5px 5px 5px 5px">13.5</div>',
    region: 'Series',
    coordinateUnits: 'Point',
    x: 'Jan',
    y: 13
  }],
}, '#element');

```

INDEX.HTML

```

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

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

```

import { AccumulationChart, AccumulationAnnotation } from '@syncfusion/ej2-charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationAnnotation);
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y'
        }
    ],
    annotations:[{
        content:'<div style="border: 1px solid black;background-color:#f5f5f5; padding: 5px 5px 5px 5px">13.5</div>',
        region: 'Chart',
        coordinateUnits: 'Pixel',
        x: 150,
        y: 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">

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { AccumulationChart, AccumulationAnnotation } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationAnnotation);
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y'
        }
    ],
    annotations:[{
        content:'<div style="border: 1px solid black;background-
color:#f5f5f5; padding: 5px 5px 5px 5px">Jan : 13.5</div>',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Jan',
        y: 13
    }],
}, '#element');
```

INDEX.HTML

```

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

```

```

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { AccumulationChart, AccumulationAnnotation } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationAnnotation);
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y'
        }
    ],
    annotations:[{
        content:'<div style="border: 1px solid black;background-
color:#f5f5f5; padding: 5px 5px 5px 5px">Jan : 13.5</div>',
        region: 'Series',
        coordinateUnits: 'Point',
        x: 'Jan',
        y: 13,
        verticalAlignment:'Top',
        horizontalAlignment:'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>

```

Tooltip in EJ2 JavaScript Accumulation chart control

Tooltip for the accumulation chart can be enabled by using the [enable](#) property.

INDEX.TS

```

import { AccumulationChart, AccumulationTooltip } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new 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'
    }
  ],
  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.TS

```

import { AccumulationChart, AccumulationTooltip } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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.TS

```

import { AccumulationChart, AccumulationTooltip } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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.TS

```

import { AccumulationChart, AccumulationTooltip } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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='https://ej2.syncfusion.com/demos/src/chart/images/sunny.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.TS

```

import { AccumulationChart, AccumulationTooltip } from '@syncfusion/ej2-
charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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.TS

```

import { AccumulationChart, AccumulationTooltip, IAccTooltipRenderEventArgs
} from '@syncfusion/ej2-charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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.TS

```

import { AccumulationChart, AccumulationTooltip, IAccTooltipRenderEventArgs
} from '@syncfusion/ej2-charts';
import { labelData } from './datasource.ts';
AccumulationChart.Inject(AccumulationTooltip);
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y'
        }
    ],
    tooltip:{
        enable: true
    },
    tooltipRender: (args: IAccTooltipRenderEventArgs) => {
        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 EJ2 JavaScript 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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
let piechart: AccumulationChart = new 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'
    }
  ],
  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>
```

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.TS

```
import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: labelData,
      xName: 'x',
      yName: 'y'
    }
  ],
  legendSettings:{ position:'Top' ,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>

```

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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: labelData,
      xName: 'x',
      yName: 'y',
      legendShape: 'Rectangle'
    }
  ],
  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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y',
            legendShape: 'Rectangle'
        }
    ],
    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

The legend size can be changed by using the **width** and **height** properties of the **legendSettings**.

INDEX.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y',
        }
    ],
    legendSettings: { width: '150', 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` properties.

INDEX.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            xName: 'x',
            yName: 'y',
        }
    ],
    legendSettings:{ shapeHeight: 15, shapeWidth: 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">

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

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.TS

```
import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationLegend);
import { data } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
  series: [
    {
      dataSource: data,
      xName: 'x',
      yName: 'y',
    }
  ],
  legendSettings: { height: '150', 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>
</script>
<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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new 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%',
            type: 'Pie',
        }
    ],
    legendSettings: { visible: true, position: 'Right', textWrap: 'Wrap',
        maximumLabelWidth: 60, height: '44%', width: '64%' }
}, '#element');

```

INDEX.HTML

```

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

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { labelData } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: labelData,
            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.TS

```
import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { data } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: data,
            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.TS

```

import { AccumulationChart, AccumulationLegend } from '@syncfusion/ej2-
charts';
AccumulationChart.Inject(AccumulationLegend);
import { data } from './datasource.ts';
let piechart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: data,
            xName: 'x',
            yName: 'y',
        }
    ],
    legendSettings:{ width: '260px', height: '50px', position: 'Bottom',
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>

```

Center label in EJ2 JavaScript 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.TS

```

import { AccumulationChart } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
let piechart: AccumulationChart = new 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.TS

```

import { AccumulationChart } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
let piechart: AccumulationChart = new 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.TS

```

import { AccumulationChart } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
let piechart: AccumulationChart = new 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: '900',
        size: '15px',
        color: 'grey',
        fontFamily: 'Roboto',
        fontStyle: 'Italic'
    }
}
}, '#element');

```

INDEX.HTML

```

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

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script 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 EJ2 JavaScript Accumulation chart control

Accumulation Chart can be given a title using [title](#) property, to show the information about the data plotted.

INDEX.TS

```
import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let pie: AccumulationChart = new 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',
});
pie.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>
```

Title Customization

Accumulation Chart can be customizing a title using [titleStyle](#) property.

INDEX.TS

```
import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let pie: AccumulationChart = new 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'
    }
});
pie.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>

```

SubTitle

Accumulation Chart can be given a subtitle using [subTitle](#) property, to show the information about the data plotted.

INDEX.TS

```

import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let pie: AccumulationChart = new 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'
  });
  pie.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>
```

SubTitle Customization

Accumulation Chart can be customizing a subtitle using [subTitleStyle](#) property, to show the information about the data plotted.

INDEX.TS

```

import { AccumulationChart, PieSeries } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(PieSeries);
let pie: AccumulationChart = new 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',
    subTitleStyle: {
        fontFamily: 'Arial',
        fontStyle: 'italic',
        fontWeight: 'regular',
        color: '#E27F2D',
        size: '13px'
    }
});
pie.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>
```

Chart print in EJ2 JavaScript Accumulation chart control

Print

The rendered chart can be printed directly from the browser by calling the public method print.

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: 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 = () => {
  piechart.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.TS

```
import { AccumulationChart, Export } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(Export);
let piechart: AccumulationChart = new 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 = () => {
  piechart.exportModule.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 EJ2 JavaScript 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 EJ2 JavaScript controls](#)

Ej1 api migration in EJ2 JavaScript 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	<code>Property:annotations\$("#chart").ejChart({ annotations: [{}]});</code>	<code>Property:annotationslet pie: AccumulationChart = new AccumulationChart({ annotations: [{}]});pie.appendTo('#chart');</code>
background	<code>Property:background\$("#chart").ejChart({ background: '#DDCCEE'});</code>	<code>Property:backgroundlet pie: AccumulationChart = new AccumulationChart({ 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	Property: size.height\$("#chart").ejChart({ size: { height: '400' } });	Property: heightlet pie: AccumulationChart = new AccumulationChart({ height: '400' }); pie.appendTo('#chart');
Multi selection	Property: selectionSettings.type\$("#chart").ejChart({ selectionSettings: { type: 'multiple' } });	Property: isMultiSelectlet pie: AccumulationChart = new AccumulationChart({ isMultiSelect: true }); pie.appendTo('#chart');
legend Settings	Property: legend\$("#chart").ejChart({ legend: { } });	Property: legendSettingslet pie: AccumulationChart = new AccumulationChart({ legendSettings: { } }); pie.appendTo('#chart');
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:

		<code>'SegoeUI'}});pie.appendTo('#chart');</code>
Sub Title text	<code>Property:subTitle.text\$("#chart").ejChart({subTitle: { text: 'Pie Chart' }});</code>	<code>Property:subTitlelet pie: AccumulationChart = new AccumulationChart({subTitle: 'Pie Chart'});pie.appendTo('#chart');</code>
Sub title Styles	<code>Property:title\$("#chart").ejChart({title: { text: 'Pie Chart' }});</code>	<code>Property:subTitleStylelet pie: AccumulationChart = new AccumulationChart({subTitleStyle: { fontFamily: 'SegoeUI'}});pie.appendTo('#chart');</code>
tooltip	<code>Property:series.toolTip\$("#chart").ejChart({series: [{ tooltip: { }}]});</code>	<code>Property:tooltiplet pie: AccumulationChart = new AccumulationChart({ tooltip: { }});pie.appendTo('#chart');</code>
Width of Chart	<code>Property:size.width\$("#chart").ejChart({size: { width: '400' }});</code>	<code>Property:widthlet pie: AccumulationChart = new AccumulationChart({ width: '400'});pie.appendTo('#chart');</code>

Annotation

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
annotations	<code>Property:annotations\$("#chart").ejChart({ annotations: [{}]});</code>	<code>Property:annotationslet pie: AccumulationChart = new AccumulationChart({annotations: [{}});pie.appendTo('#chart');</code>
content	<code>Property:annotations\$("#chart").ejChart({ annotations: [{content: 'Pie Chart'}]});</code>	<code>Property:annotationslet pie: AccumulationChart = new AccumulationChart({annotations: [{content: 'Pie Chart'}]});pie.appendTo('#chart');</code>
coordinateUnits	<code>Property:annotations\$("#chart").ejChart({ annotations: [{coordinateUnit: 'Pixel'}]});</code>	<code>Property:annotationslet pie: AccumulationChart = new AccumulationChart({ annotations: [{coordinateUnit: 'Pixel'}]});pie.appendTo('#chart');</code>

description	Not Applicable	Property: description let pie: AccumulationChart = new AccumulationChart({ annotations: [{ description: 'Pixel' }]; }); pie.appendTo('#chart');
horizontalAlignment for annotation	Property: annotations.horizontalAlignment \$("#container").ejChart({ annotations: [{ horizontalAlignment: "middle" }]});	Property: annotations.horizontalAlignment let 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:

		<code>'100']}]); chart.appendTo('#chart');</code>
--	--	--

Series

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
series	<code>Property:series\$("#chart").ejChart({ series: [{ }]; });</code>	<code>Property:serieslet pie: AccumulationChart = new AccumulationChart({ series: [{ }]; }); pie.appendTo('#chart');</code>
animation for series	<code>Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]; });</code>	<code>Property:animation.enablelet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { enable: true } }]; }); pie.appendTo('#chart');</code>
animation duration for series	<code>Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000 }]; });</code>	<code>Property:animation.durationlet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { duration: 1000 } }]; }); pie.appendTo('#chart');</code>
animation delay for series	Not Applicable	<code>Property:animation.durationlet pie: AccumulationChart = new AccumulationChart({ series: [{ animation: { delay: 100 } }]; }); pie.appendTo('#chart');</code>
Border for series	<code>Property:border\$("#chart").ejChart({ series: [{ border: { color: 'pink', width: 2, dashArray: '10,4' } }]; });</code>	<code>Property:borderlet pie: AccumulationChart = new AccumulationChart({ series: [{ border: { color: 'red', width: 2 } }]; }); pie.appendTo('#chart');</code>
DataLabel for series	<code>Property:dataLabel\$("#chart").ejChart({ series: [{ dataLabel: { } }]; });</code>	<code>Property:labellet pie: AccumulationChart = new AccumulationChart({ series: [{ dataLabel: { } }]; }); pie.appendTo('#chart');</code>

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 }]});pie.appendTo('#chart');
end angle	Property:endAngle\$("#chart").ejChart({ series: [{ endAngle: 80 }]});	Property:endAnglelet pie: AccumulationChart = new AccumulationChart({ series: [{ endAngle: 90 }]});pie.appendTo('#chart');
explode	Property:explode\$("#chart").ejChart({ series: [{ explode: true }]});	Property:explodelet pie: AccumulationChart = new AccumulationChart({ series: [{ explode: true }]});pie.appendTo('#chart');
explodeAll	Property:explodeAll\$("#chart").ejChart({ series: [{ explodeAll: true }]});	Property:explodeAlllet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeAll: true }]});pie.appendTo('#chart');
explodeIndex	Property:explode\$("#chart").ejChart({ series: [{ explodeIndex: 0 }]});	Property:explodeIndexlet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeIndex: 1 }]});pie.appendTo('#chart');
explodeOffset	Property:explodeOffset\$("#chart").ejChart({ series: [{ explodeOffset: '30%' }]});	Property:explodeOffsetlet pie: AccumulationChart = new AccumulationChart({ series: [{ explodeOffset: '30%' }]});pie.appendTo('#chart');

gapRatio	Property:gapRatio\$("#chart").ejChart({ series: [{ gapRatio: 0.6 }]});	Property:gapRatiolet pie: AccumulationChart = new AccumulationChart({ series: [{ gapRatio: 0.6 }]});pie.appendTo('#chart');
gapWidth	Property:gapWidth\$("#chart").ejChart({ series: [{ gapWidth: 0.6 }]});	Not Applicable
inner radius of the accumulation chart	Property:innerRadius\$("#chart").ejChart({ series: [{ innerRadius : '30%' }]});	Property:innerRadiuslet pie: AccumulationChart = new AccumulationChart({ series: [{ innerRadius : '30%' }]});pie.appendTo('#chart');
Legend shape of the series	Not applicable	Property:legendShapelet pie: AccumulationChart = new AccumulationChart({ series: [{ legendShape : 'Rectangle' }]});pie.appendTo('#chart');
name of the series	Property:name\$("#chart").ejChart({ series: [{ name : '30%' }]});	Property:namelet pie: AccumulationChart = new AccumulationChart({ series: [{ name : '30%' }]});pie.appendTo('#chart');
neck height for funnel series	Property:neckHeight\$("#chart").ejChart({ series: [{ neckHeight : '30%' }]});	Property:neckHeightlet pie: AccumulationChart = new AccumulationChart({ series: [{ neckHeight : '30%' }]});pie.appendTo('#chart');
neck width for funnel series	Property:neckWidth\$("#chart").ejChart({ series: [{ neckWidth : '30%' }]});	Property:neckWidthlet pie: AccumulationChart = new AccumulationChart({ series: [{ neckWidth : '30%' }]});pie.appendTo('#chart');
opacity for series	Property:opacity\$("#chart").ejChart({ series: [{ opacity : 0.4 }]});	Property:opacitylet pie: AccumulationChart = new AccumulationChart({ series: [{ opacity : 0.5 }]});pie.appendTo('#chart');
palettes for series	Property:palette\$("#chart").ejChart({ series: [{ palette : [] }]});	Property:paletteslet pie: AccumulationChart = new AccumulationChart({ series: [{ palettes : [] }]});

		<code>});});pie.appendTo('#chart');</code>
point color mapping name for series	<code>Property:pointColorMappingName\$("#chart").ejChart({ series: [{ pointColorMappingName : 'color' }]});</code>	<code>Property:pointColorMappinglet pie: AccumulationChart = new AccumulationChart({ series: [{ pointColorMappingName : 'color' }]});pie.appendTo('#chart');</code>
Mode of pyramid series	<code>Property:pyramidMode\$("#chart").ejChart({ series: [{ pyramidMode : 'Surface' }]});</code>	<code>Property:pyramidModelet pie: AccumulationChart = new AccumulationChart({ series: [{ pyramidMode : 'Linear' }]});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	<code>Property:xName\$("#chart").ejChart({ series: [{ xName : 'x' }]});</code>	<code>Property:xNamelet chart: AccumulationChart = new</code>

property in the datasource that contains x value for the series.		<code>AccumulationChart({ series: [{ xName: 'x' }] }); chart.appendTo('#chart');</code>
Name of the property in the datasource that contains x value for the series.	<code>Property:yName\$("#chart").ejChart({ series: [{ yName : 'x' }] });</code>	<code>Property:yNamelet chart: AccumulationChart = new AccumulationChart({ series: [{ yName: 'x' }] }); chart.appendTo('#chart');</code>
Name of the property in the datasource that contains x value for the series.	<code>Property:yName\$("#chart").ejChart({ series: [{ yName : 'x' }] });</code>	<code>Property:yNamelet chart: AccumulationChart = new AccumulationChart({ series: [{ yName: 'x' }] }); chart.appendTo('#chart');</code>
Width of funnel series	<code>Property:funnelWidth\$("#chart").ejChart({ series: [{ funnelWidth : '100' }] });</code>	<code>Property:widthlet chart: AccumulationChart = new AccumulationChart({ series: [{ width: '100' }] }); chart.appendTo('#chart');</code>
Grouping	Not Applicable	<code>Property:groupTolet chart: AccumulationChart = new AccumulationChart({ series: [{ groupTo: '100' }] }); chart.appendTo('#chart');</code>
GroupMode	Not Applicable	<code>Property:groupModelet chart: AccumulationChart = new AccumulationChart({ series: [{ groupMode: 'Point' }] }); chart.appendTo('#chart');</code>

[DataLabel](#)

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
dataLabel	<code>Property:series.marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel : { visible: true }}}]});</code>	<code>Property:series.dataLabellet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { visible: true }}});pie.appendTo('#chart');</code>
border of dataLabel	<code>Property:series.marker.dataLabel.border\$("#chart").ejChart({ series: [{ marker: { dataLabel : { border: { color: 'red', width: 2 }}} }]});</code>	<code>Property:borderlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { border: { color: 'red', width: 2 }}}});pie.appendTo('#chart');</code>
connector style for dataLabel connector line	<code>Property:connectorLine\$("#chart").ejChart({ series: [{ marker: { dataLabel : { connectorLine: { type: 'Curve', width: 2 }}} }]});</code>	<code>Property:connectorStylelet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { connectorStyle: { type: 'Curve', width: 2 }}}});pie.appendTo('#chart');</code>
Fill for dataLabel	<code>Property:fill\$("#chart").ejChart({ series: [{ marker: { dataLabel : { fill: 'red' }}}]});</code>	<code>Property:filllet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { fill: 'pink' }}});pie.appendTo('#chart');</code>
font for dataLabel	<code>Property:font\$("#chart").ejChart({ series: [{ marker: { dataLabel : { font: { } }}}]});</code>	<code>Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { font: { } }}});pie.appendTo('#chart');</code>
font for dataLabel	<code>Property:font\$("#chart").ejChart({ series: [{ marker: { dataLabel : { font: { } }}}]});</code>	<code>Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { font: { } }}});pie.appendTo('#chart');</code>
position	<code>Property:position\$("#chart").ejChart({ series: [{ marker: { dataLabel : { position: 'Inside' }}}]});</code>	<code>Property:fontlet pie: AccumulationChart = new AccumulationChart({ series: { dataLabel : { position: 'Outside' }}});pie.appendTo('#chart');</code>

Rounde d corner radius X	Not Applicable	Property: dataLabel.rx let chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { rx: 10 } }];});chart.appendTo('#chart
Rounde d corner radius Y	Not Applicable	Property: dataLabel.ry let chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { ry: 10 } }];});chart.appendTo('#chart
HTML templat e in dataLab el	Property: dataLabel.template \$("#chart").ejChart({ series: [{ marker: {dataLabel: { template: ' Chart ' } } }];});	Property: dataLabel.template let chart: AccumulationChart = new AccumulationChart({ series: [{dataLabel: { template: ' Chart ' } }];});chart.appendTo('#chart

Legend

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Default legend	Property: visible \$("#container").ejChart({ legend: { visible: true }});	Property: visible let chart: AccumulationChart = new AccumulationChart({ legendSettings: { visible: true }});
Legend height	Property: size.height \$("#container").ejChart({ legend: { size : { height: 50 } }});	Property: height let chart: AccumulationChart = new AccumulationChart({ legendSettings: { height: '30' }});
Legend width	Property: size.width \$("#container").ejChart({ legend: { size: { width: 20 } }});	Property: width let chart: AccumulationChart = new AccumulationChart({ legendSettings: { width: '30' }});
Legend location in chart	Property: location \$("#container").ejChart({ legend: { location: { x: 3, y: 45} }});	Property: height let chart: AccumulationChart = new AccumulationChart({ legendSettings: { location: { x: 3, y: 45} }});

Legend position in chart	Property:position\$("#container").ejChart({ legend: { position: 'top' } });	Property:positionlet chart: AccumulationChart = new AccumulationChart({ legendSettings: { position: 'Top' } });
Legend padding	Not applicable	Property:paddinglet chart: AccumulationChart = new AccumulationChart({ legendSettings: { padding: 8 } });
Legend alignment	Property:position\$("#container").ejChart({ legend: { alignment: 'center' } });	Property:positionlet chart: AccumulationChart = new AccumulationChart({ legendSettings: { alignment: 'Center' } });
text style for legend	Property:font\$("#container").ejChart({ legend: { font: { fontFamily: '', fontWeight: '400', fontStyle: 'italic', size: '12' } } });	Property:textStylelet chart: AccumulationChart = new AccumulationChart({ legendSettings: { textStyle: { size: '12', 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: AccumulationChart = new AccumulationChart({ legendSettings: { shapeHeight: 20 } });
shape width of legend	Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });	Property:shapeWidthlet chart: AccumulationChart = new AccumulationChart({ legendSettings: { shapeWidth: 20 } });
shape width of legend	Property:itemStyle.width\$("#container").ejChart({ legend: { itemStyle: { width: 20 } } });	Property:shapeWidthlet chart: AccumulationChart = new AccumulationChart({ 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: AccumulationChart = new AccumulationChart({

		legendSettings: { shapePadding: 20 } });
Background of legend	Property:background\$("#container").ejChart({ legend: { background: 'transparent' } });	Property:backgroundlet chart: AccumulationChart = new AccumulationChart({ legendSettings: { background: 'transparent' } });
Opacity of legend	Property:opacity\$("#container").ejChart({ legend: { opacity: 0.3 } });	Property:opacitylet chart: AccumulationChart = new AccumulationChart({ legendSettings: { opacity: 0.4 } });
Toggle visibility of series while legend click	Property:toggleSeriesVisibility\$("#container").ejChart({ legend: { toggleSeriesVisibility: true } });	Property:toggleVisibilitylet chart: AccumulationChart = new AccumulationChart({ 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.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({chartMouseClicked: () => { }});chart.appendTo('#chart');
Fires on chart mouse move	Property:chartMouseMove\$("#chart").ejChart({chartMouseMove: () { }});	Property:chartMouseMovelet chart: AccumulationChart = new AccumulationChart({chartMouseMove: () => { }});chart.appendTo('#chart');
Fires on chart mouse leave	Property:chartMouseLeave\$("#chart").ejChart({chartMouseLeave: () { }});	Property:chartMouseLeavelet chart: AccumulationChart = new AccumulationChart({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: AccumulationChart = new AccumulationChart({chartmouseUp: () => { }});chart.appendTo('#chart');
Fires on chart mouse down	Not Applicable	Property:chartmouseDownlet chart: AccumulationChart = new AccumulationChart({chartmouseDown: () => { }});chart.appendTo('#chart');
Fires during the calculatio	Property:chartAreaBoundsCalculate\$("#chart").ejChart({chartAreaBoundsCalculate: () { }});	Not applicable

n of chart area bounds. You can use this event to customize the bounds of chart area		
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: 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	Property:legendItemRendering\$("#chart").ejChart({ legendItemRendering: () { } });	Property:legendRenderlet chart: AccumulationChart = new AccumulationChart({ legendRender: () => { } });chart.appendTo('#chart');

legend item.		
Fires before loading the chart.	Property:load \$("#chart").ejChart({ load: () { } });	Property:load let 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.	Property:pointRegionMouseMove \$("#chart").ejChart({ pointRegionMouseMove: () { } });	Property:pointMove let chart: AccumulationChart = new AccumulationChart({ pointMove : () => { } });chart.appendTo('#chart');
Fires before rendering chart.	Property:preRender \$("#chart").ejChart({ preRender: () { } });	Not applicable
Fires when point render.	Not Applicable	Property:pointRender let chart: AccumulationChart = new AccumulationChart({ pointRender : () => { } });chart.appendTo('#chart');
Fires before rendering a series.	Property:seriesRendering \$("#chart").ejChart({ seriesRendering: () { } });	Property:seriesRender let chart: AccumulationChart = new AccumulationChart({ seriesRender : () => { } });chart.appendTo('#chart');
Fires before rendering the Chart title.	Property:titleRendering \$("#chart").ejChart({ titleRendering: () { } });	Not applicable
Fires before rendering	Property:subTitleRendering \$("#chart").ejChart({ subTitleRendering: () { } });	Not applicable

the Chart sub title.		
Fires before rendering the tooltip.	Property:tooltipInitialize\$("#chart").ejChart({ tooltipInitialize: () { }});	Property:tooltipRender let chart: AccumulationChart = new AccumulationChart({ tooltipRender : () => { }});chart.appendTo('#chart');

AppBar

Size and color in EJ2 JavaScript 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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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>
</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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Primary',
    mode: 'Prominent',
    cssClass: 'prominent-appbar'
});
defaultAppBarObj.appendTo("#defaultAppBar");

```

```

const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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="prominent">AppBar Component with Prominent
mode</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>
```

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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary',
  mode: 'Dense'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo("#defaultButtonMenu");
const defaultButtonLoginObj: Button = new 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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonLoginObj: Button = new 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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Dark'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo("#defaultButtonMenu");
const defaultButtonLoginObj: Button = new 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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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>
</body>
</html>
```

```

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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Inherit'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonLoginObj: Button = new 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
```

```
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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>
        </div>
      </div>
    </div>
  </div>
```

```

        <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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Primary',
    position: 'Bottom'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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>
          <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>
</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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary',
  isSticky: true
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultButtonLoginObj: Button = new 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 EJ2 JavaScript 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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";

```

```

const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonHomeObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-home' });
defaultButtonHomeObj.appendTo('#defaultButtonHome');
const defaultButtonCutObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-cut' });
defaultButtonCutObj.appendTo('#defaultButtonCut');
const defaultButtonPanObj: Button = new 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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonCutObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-cut' });
defaultButtonCutObj.appendTo("#defaultButtonCut");
const defaultButtonCopyObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-copy' });
defaultButtonCopyObj.appendTo("#defaultButtonCopy");
const defaultButtonPasteObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-paste' });
defaultButtonPasteObj.appendTo("#defaultButtonPaste");
const defaultButtonBoldObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-bold' });
defaultButtonBoldObj.appendTo("#defaultButtonBold");
const defaultButtonUnderlineObj: Button = new Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-underline' });
defaultButtonUnderlineObj.appendTo("#defaultButtonUnderline");
const defaultButtonItalicObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-italic' });
defaultButtonItalicObj.appendTo("#defaultButtonItalic");
const defaultButtonAlignLeftObj: Button = new Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-left' });
defaultButtonAlignLeftObj.appendTo("#defaultButtonAlignLeft");
const defaultButtonAlignRightObj: Button = new Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-right' });
defaultButtonAlignRightObj.appendTo("#defaultButtonAlignRight");
const defaultButtonAlignCenterObj: Button = new Button({ cssClass: 'e-
inherit', iconCss: 'e-icons e-align-center' });
defaultButtonAlignCenterObj.appendTo("#defaultButtonAlignCenter");
const defaultButtonJustifyObj: Button = new 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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo("#defaultButtonMenu");
const defaultButtonHomeObj: Button = new Button({ cssClass: 'e-inherit',
content: 'Home' });
defaultButtonHomeObj.appendTo("#defaultButtonHome");
const defaultButtonAboutObj: Button = new Button({ cssClass: 'e-inherit',
content: 'About' });
defaultButtonAboutObj.appendTo("#defaultButtonAbout");
const defaultButtonProductsObj: Button = new Button({ cssClass: 'e-inherit',
content: 'Products' });
defaultButtonProductsObj.appendTo("#defaultButtonProducts");
const defaultButtonContactsObj: Button = new Button({ cssClass: 'e-inherit',
content: 'Contacts' });
defaultButtonContactsObj.appendTo("#defaultButtonContacts");
const defaultButtonLoginObj: Button = new 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>
                    <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>
<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.TS

```

import { AppBar, Menu, MenuItemModel } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const companyMenuItems: MenuItemModel[] = [
    {
        text : 'Company',
        items: [
            { text: 'About Us' },
            { text: 'Customers' },
            { text: 'Blog' },
            { text: 'Careers' }
        ]
    }
];
const productMenuItems: MenuItemModel[] = [
    {
        text : 'Products',
        items: [
            { text: 'Developer' },
            { text: 'Analytics' },
            { text: 'Reporting' },

```

```

        { text: 'Help Desk' }
    ]
}
];
const aboutMenuItems: MenuItemModel[] = [
    {
        text : 'About Us'
    }
];
const carrerMenuItems: MenuItemModel[] = [
    {
        text : 'Carrers'
    }
];
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultMenuCompanyObj: Menu = new Menu({ cssClass: 'e-inherit', items:
companyMenuItems });
defaultMenuCompanyObj.appendTo('#defaultMenuCompany');
const defaultMenuProductsObj: Menu = new Menu({ cssClass: 'e-inherit',
items: productMenuItems });
defaultMenuProductsObj.appendTo('#defaultMenuProducts');
const defaultMenuAboutObj: Menu = new Menu({ cssClass: 'e-inherit', items:
aboutMenuItems });
defaultMenuAboutObj.appendTo('#defaultMenuAbout');
const defaultMenuCarrersObj: Menu = new Menu({ cssClass: 'e-inherit', items:
carrerMenuItems });
defaultMenuCarrersObj.appendTo('#defaultMenuCarrers');
const defaultButtonLoginObj: Button = new 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>
                    <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>
</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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
const productDropDownButtonItem: ItemModel[] = [
    { text: 'Developer' },
    { text: 'Analytics' },
    { text: 'Reporting' },
    { text: 'E-Signature' },
    { text: 'Help Desk' }
];
const defaultAppBarObj: AppBar = new AppBar({
    colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");

```

```

const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
const defaultDropDownButtonProductObj: DropDownButton = new DropDownButton({
cssClass: 'e-inherit', items: productDropDownButtonItem });
defaultDropDownButtonProductObj.appendTo('#defaultDropDownButtonProduct');
const defaultButtonLoginObj: Button = new 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.TS

```

import { AppBar, Sidebar, TreeView } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
import { TextBox } from "@syncfusion/ej2-inputs";
let data: { [key: string]: Object }[] = [
    {
        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' }
        ]
    }
];
const defaultAppBarObj: AppBar = new AppBar({});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new Button({ cssClass: 'e-inherit',
iconCss: 'e-icons e-menu' });
defaultButtonMenuObj.appendTo('#defaultButtonMenu');
defaultButtonMenuObj.element.addEventListener("click", toggle);
let sidebarMenu: Sidebar = new Sidebar({
    width: '220px',
    target: '.main-content',
    mediaQuery: '(min-width: 600px)',
    isOpen: true
});
sidebarMenu.appendTo('#sideTree');
let inputObj: TextBox = new TextBox({
    placeholder: "Search..."
});
inputObj.appendTo("#resSearch");

```



```

let mainTreeView: TreeView = new TreeView({
    cssClass: "main-treeview",
    fields: { dataSource: data, id: 'nodeId', text: 'nodeText', child:
'nodeChild' },
    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>
    </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>
</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 EJ2 JavaScript 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 |

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

|.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.TS

```
import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary',
  cssClass: 'custom-appbar'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new 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.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>
        </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>

```

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.TS

```

import { AppBar } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const defaultAppBarObj: AppBar = new AppBar({
  colorMode: 'Primary'
});
defaultAppBarObj.appendTo("#defaultAppBar");
const defaultButtonMenuObj: Button = new 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>
<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 EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
'Football', 'Golf', 'Gymnastics', 'Hockey', 'Tennis'];
// initialize AutoComplete component
let atcObject: AutoComplete = new AutoComplete({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to AutoComplete input element
    placeholder: "Find a game"
});
// render initialized AutoComplete
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>
```

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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
  { 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' }
];
//initiate the AutoComplete
let atcObject: AutoComplete = new AutoComplete({
  // bind the sports Data to datasource property
  dataSource: sportsData,
  // maps the appropriate column to fields property
  fields: { value: 'Game' },
  //set the placeholder to AutoComplete input
  placeholder: "Find a game"
});
//render the component
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;">
  <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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let countriesData: { [key: string]: Object }[] = [
  { 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' } }
];
//initiate the AutoComplete
let atcObject: AutoComplete = new AutoComplete({
  // bind the sports Data to datasource property
  dataSource: countriesData,
  // maps the appropriate column to fields property
  fields: { value: 'Country.Name' },
  //set the placeholder to AutoComplete input
  placeholder: "Find a country"
});
//render the component
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;">
    <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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let customers: AutoComplete = new AutoComplete({
  //bind the DataManager instance to dataSource property
  dataSource: new DataManager({

```

```

        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    )),
    //bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    //set the placeholder to AutoComplete input
    placeholder: "Find a customer",
    //sort the resulted items
    sortOrder: 'Ascending'
});
//render the component
customers.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 EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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: "Find 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
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" id="atcelelement">
    </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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, Predicate, DataManager, ODataV4Adaptor } from
 '@syncfusion/ej2-data';
// form predicate to fetch the grouped data
let groupPredicate = new Predicate('City', 'equal', 'london').or('City',
 'equal', 'seattle');
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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: "Find an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to groupTemplate
    groupTemplate: "<strong>${City}</strong>"
  });
  //render the component
  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" 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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: "Find an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the value to header template
    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
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" 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.TS

```

import { AutoComplete, OpenEventArgs } from '@syncfusion/ej2-dropdowns';
let sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf', 'Gymnastics', 'Hockey'];
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: sportsData,
    //set the placeholder to AutoComplete input
    placeholder: "Find a game",
    open: (e: OpenEventArgs) => {
        let count = atcObject.getItems().length;
        //set the value to footerTemplate property
        let ele = document.getElementsByClassName('foot')[0];
        ele.innerHTML = "Total list item: " + count;
    }
    //set the footer template
    footerTemplate: "<span class='foot'></span>"
});
//render the component
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" 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: [],
    //set the placeholder to AutoComplete input
    placeholder: "Find an item",
    //set the value to noRecords template
    noRecordsTemplate: "<span class='norecord'> NO DATA AVAILABLE</span>"
});
//render the component
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>

```

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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        // use the wrong url to display the action failure template
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName']).take(6),
    //map the appropriate columns to fields property
    fields: { value: 'FirstName' },
    //set the placeholder to AutoComplete input
    placeholder: "Find an employee ",
    //set the value to action failure template
    actionFailureTemplate: "<span class='action-failure'> Data fetch get
fails</span>"
});
//render the component

```

```
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>
```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)

Grouping in EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//define the data with category
let vegetableData: { [key: string]: Object }[] = [
  { 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
let vegetables: AutoComplete = new 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>
</body>
</html>
```

```

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 EJ2 JavaScript 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let filter: AutoComplete = new AutoComplete({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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'
});
filter.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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let filter: AutoComplete = new AutoComplete({
```

```

//bind the DataManager instance to dataSource property
dataSource: new DataManager({
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
}),
//bind the Query instance to query property
query: new Query().select(['ContactName', 'CustomerID']),
//set the suggestionCount to show the maximum suggestion list item
suggestionCount: 5,
//map the appropriate columns to fields property
fields: { value: 'ContactName' },
//set the placeholder to AutoComplete input
placeholder: "Find a customer",
//sort the resulted items
sortOrder: 'Ascending'
});
filter.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>

```

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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let filter: AutoComplete = new AutoComplete({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().select(['ContactName', 'CustomerID']),
    //set the minLength to restrict the remote request until search key
    //contains 3 characters.
    minLength: 3,
    //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'
});
filter.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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
//initiates the component
let filter: AutoComplete = new AutoComplete({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url:
        'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Customers').select('ContactName').take(7),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    //set the placeholder to AutoComplete input
    placeholder: "Find a customer",
    //set the ignoreCase as false to enable the case sensitive filtering
    ignoreCase: false,
    //set the filterType to searching operation
    filterType: 'StartsWith',
    //sort the resulted items
    sortOrder: 'Ascending'
});
filter.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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
// create local data
let data: string[] = [
  'Aeróbics',
  'Aeróbics en Agua',
  'Aerografía',
  'Aeromodelaje',
  'Águilas',
  'Ajedrez',
  'Ala Delta',
  'Álbumes de Música',
  'Alusivos',
  'Análisis de Escritura a Mano'];

```

```
// initialize AutoComplete component
let atcObj: AutoComplete = new 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.TS

```
import { AutoComplete, VirtualScroll } from '@syncfusion/ej2-dropdowns';
AutoComplete.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i: number = 1; i <= 150; i++) {
    let item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the dataSource 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
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">
  <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.TS

```

import { AutoComplete, VirtualScroll } from '@syncfusion/ej2-dropdowns';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
AutoComplete.Inject(VirtualScroll);
//initiates the component
let atcObject: AutoComplete = new AutoComplete({

```

```

//bind the dataSorce property
dataSource: new DataManager({
    url: 'https://ej2services.syncfusion.com/js/development/api/orders',
    adaptor: new 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
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">
    <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.TS

```
import { AutoComplete, VirtualScroll } from '@syncfusion/ej2-dropdowns';
AutoComplete.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i = 1; i <= 150; i++) {
    let item: { [key: string]: Object } = {};
    item.id = 'id' + i;
    item.text = `Item ${i}`;
    // Generate a random number between 1 and 4 to determine the group
    const randomGroup = Math.floor(Math.random() * 4) + 1;
    switch (randomGroup) {
        case 1:
            item.group = 'Group A';
            break;
        case 2:
            item.group = 'Group B';
            break;
        case 3:
            item.group = 'Group C';
            break;
        case 4:
            item.group = 'Group D';
            break;
        default:
            break;
    }
    records.push(item);
}
//initiates the component
let atcObject: AutoComplete = new AutoComplete({
    //bind the data source 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
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">
  <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 EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
// import L10n class for load function
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
let atcObj: AutoComplete = new AutoComplete({
    // bind remote data to showcase actionFailureTemplate in offline.
    dataSource: new DataManager({
        url:
            'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().select(['ContactName', 'CustomerID']).take(0),
    //map the appropriate columns to fields property
    fields: { value: 'ContactName' },
    // set locale culture to AutoComplete
    locale: 'fr-BE',
    // set placeholder to AutoComplete input element
    placeholder: 'Trouver un client'
});
atcObj.appendTo('#atcelement');
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 EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
'Football', 'Golf', 'Gymnastics', 'Hockey', 'Tennis'];
// initialize AutoComplete component
let atcObject: AutoComplete = new AutoComplete({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to AutoComplete input element
    placeholder: "Find a game"
    floatLabelType: "auto"
});
// render initialized AutoComplete
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="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 EJ2 JavaScript 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.TS

```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let gameList: { [key: string]: Object }[] = [
  { Id: 'Game1', Game: 'Badminton' },
  { Id: 'Game2', Game: 'Basketball' },
  { Id: 'Game3', Game: 'Cricket' },
  { Id: 'Game4', Game: 'Football' },
  { Id: 'Game5', Game: 'Golf' },
  { Id: 'Game6', Game: 'Gymnastics' },
  { Id: 'Game7', Game: 'Hockey' },
  { Id: 'Game8', Game: 'Rugby' },
  { Id: 'Game9', Game: 'Snooker' },
  { Id: 'Game10', Game: 'Tennis' },
];
// initialize AutoComplete component
let atcObject: AutoComplete = new AutoComplete({
  //set the data to dataSource property
  dataSource: gameList,
  //map to column to fields
  fields: { value: 'Game' },
  // set placeholder to AutoComplete input element
  placeholder: "Find a game"
});
// render initialized AutoComplete
atcObject.appendTo('#atcelement');
document.onkeyup = function (e) {
  if (e.altKey && e.keyCode === 84 /* t */) {
    // press alt+t to focus the control.
    atcObject.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 EJ2 JavaScript 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.TS


```
import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let searchData: { [key: string]: Object; }[] = [
  { 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: AutoComplete = new 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 EJ2 JavaScript 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
let sortFormatData: { [key: string]: Object }[] = [
    { 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' }
];
let sortFormat: AutoComplete = new 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 EJ2 JavaScript 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let searchData: { [key: string]: Object; }[] = [
    { 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: AutoComplete = new 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 EJ2 JavaScript 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
import { Query, Predicate } from '@syncfusion/ej2-data';
//define the array of complex data
let searchData: { [key: string]: Object; }[] = [
  { 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: AutoComplete = new 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
  placeholder: "Find a country",
  //set item template
  itemTemplate: "<span><span class='name'>${Name}</span>-<span class='code'>${Code}</span></span></span>",
  filtering: function(e)
  {
    e.preventDefaultAction=true;
    let predicate = new Predicate('Name', 'contains', e.text);
    predicate = predicate.or('Code', 'contains', e.text);
    let query = new 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 EJ2 JavaScript 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.TS

```

import { AutoComplete } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
    crossDomain: true
});

// initialize AutoComplete component
let autoObj: AutoComplete = new AutoComplete({
    // bind the DataManager instance to dataSource property
    dataSource: data,
    // bind the Query instance to query property
    query: new Query().from('Customers').select('ContactName').take(7),
    // map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'ContactName' },
    // set the placeholder to AutoComplete input element
    placeholder: 'Select a customer',

```

```

        // sort the resulted items
        sortOrder: 'Ascending',
        // set the height of the popup element
        popupHeight: '200px',
        actionComplete: function (e: any) {
            let operator: Query = new
Query().from('Customers').select('ContactName');
            let start: number = 7;
            let end: number = 12;
            let listElement: HTMLElement = this.list;
            listElement.addEventListener('scroll', () => {
                if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                    let filterQuery = operator.clone();
                    data.executeQuery(filterQuery.range(start,
end)).then((event: any) => {
                        start = end;
                        end += 5;
                        autoObj.addItem(event.result as { [key: string]: Object
}[]);
                    }).catch((e: Object) => {
                    });
                }
            });
        });
        autoObj.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>
</body>
</html>

```

```

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 EJ2 JavaScript Auto complete control

This article describes the API migration process of AutoComplete component from Essential JS 1 to Essential JS 2.

MultiSelect concept is not present in EJ2-AutoComplete. If you want to use multiselection support in autocomplete, we suggest you to use MultiSelect component.

DataBinding

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Default** | **Property:** *datasource*
 `$('#autocomplete').ejAutocomplete({ dataSource: countriesField,});` | **Property:** *dataSource*
 `var groupObj = new ej.dropdowns.AutoComplete({dataSource: vegetableData,});groupObj.appendTo('#vegetables');` |

| **Fields for mapping** | **Property:** *fields*
 `$('#autocomplete').ejAutocomplete({fields: { key: "index", text: "name" },});` | **Property:** *fields*
 `var groupObj = new ej.dropdowns.AutoComplete({fields: { groupBy: 'Category', value: 'Vegetable' },});groupObj.appendTo('#vegetables');` |

| **Query** | **Property:** *query*
 `$('#autocomplete').ejAutocomplete({query: query,});` | **Property:** *query*
 `var groupObj = new ej.dropdowns.AutoComplete({query: ej.Query().requiresCount(),});groupObj.appendTo('#vegetables');` |

| **Begin event** | **Event:** *actionBegin*
 `$('#autocomplete').ejAutocomplete({actionBegin: "actionBegin",});` | **Event:** *actionBegin*
 `var groupObj = new ej.dropdowns.AutoComplete({actionBegin: "actionBegin",});groupObj.appendTo('#vegetables');` |

| **Complete event** | **Event:** *actionComplete*
 `$('#autocomplete').ejAutocomplete({actionComplete: "actionComplete",});` | **Event:** *actionComplete*
 `var groupObj = new ej.dropdowns.AutoComplete({actionComplete: "actionComplete",});groupObj.appendTo('#vegetables');` |

| **Failure event** | **Event:** *actionFailure*
 `$('#autocomplete').ejAutocomplete({actionFailure: "actionFailure",});` | **Event:** *actionFailure*
 `var groupObj = new ej.dropdowns.AutoComplete({actionFailure: "actionFailure",});groupObj.appendTo('#vegetables');` |

| **Success event** | **Event:** *actionSuccess*
\$('#autocomplete').ejAutocomplete({actionSuccess: "actionSuccess",}); | **Not Applicable** |

Filtering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Case sensitivity** | **Property:** *caseSensitiveSearch*
\$('#autocomplete').ejAutocomplete({ caseSensitiveSearch: true,}); | **Property:** *ignoreCase*
var groupObj = new ej.dropdowns.AutoComplete({ignoreCase: true,});groupObj.appendTo('#vegetables'); |

| **Accent effective search** | **Not applicable** | **Property :** *ignoreAccent*
var groupObj = new ej.dropdowns.AutoComplete({ignoreAccent: true,});groupObj.appendTo('#vegetables'); |

| **Filtering Type** | **Property:** *filterType*
\$('#autocomplete').ejAutocomplete({ filterType: "Contains",}); | **Property:** *filterType*
var groupObj = new ej.dropdowns.AutoComplete({filterType: filtertype,});groupObj.appendTo('#vegetables'); |

| **Autofill** | **Property:** *enableAutoFill*
\$('#autocomplete').ejAutocomplete({ enableAutoFill: true,}); | **Property::** *autoFill*
var groupObj = new ej.dropdowns.AutoComplete({autoFill: true,});groupObj.appendTo('#vegetables'); |

| **Highlight the search word** | **Property:** *highlightSearch* \$('#autocomplete').ejAutocomplete({ highlightSearch: true,}); | **Property:** *highlight*
var groupObj = new ej.dropdowns.AutoComplete({highlight: true,});groupObj.appendTo('#vegetables'); |

| **No of items to be shown** | **Property:** *itemsCount*
\$('#autocomplete').ejAutocomplete({ itemsCount: 3,}); | **Property:** *suggestionCount*
var groupObj = new ej.dropdowns.AutoComplete({suggestionCount: 5,});groupObj.appendTo('#vegetables'); |

| **Minimum characters to enter** | **Property:** *minCharacter*
\$('#autocomplete').ejAutocomplete({ minCharacter: 3,}); | **Property:** *minLength*
var groupObj = new ej.dropdowns.AutoComplete({minLength: 4,});groupObj.appendTo('#vegetables'); |

| **Search** | **Method:** *search*
\$("#autocomplete").ejAutocomplete("search"); | **Not applicable** |

Placeholder

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Watermark text** | **Property:** *watermarkText*
\$('#autocomplete').ejAutocomplete({watermarkText:"select" }); | **Property:** *placeholder*
var groupObj = new ej.dropdowns.AutoComplete({placeholder: "Select",});groupObj.appendTo('#vegetables'); |

| **Floating of watermark text** | **Not applicable** | **Property:** *floatLabelType*
var groupObj = new ej.dropdowns.AutoComplete({floatLabelType: floatLabelType,});groupObj.appendTo('#vegetables'); |

Popup

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **No records text** | **Property:** *emptyResultText*


```
$('#autocomplete').ejAutocomplete({emptyResultText:"no records" });| Property:  
noRecordsTemplate<br/> var groupObj = new ej.dropdowns.AutoComplete({noRecordsTemplate:  
noRecordsTemplate,});groupObj.appendTo('#vegetables');
```

| **No records showing** | **Property:** *showEmptyResultText*


```
$('#autocomplete').ejAutocomplete({showEmptyResultText:true }) | Not applicable |
```

| **Popupbutton** | **Property:** *showPopupButton*


```
$('#autocomplete').ejAutocomplete({ShowPopupButton:true }) | Property:  
showPopupButton<br/> var groupObj = new ej.dropdowns.AutoComplete({showPopupButton:  
true,});groupObj.appendTo('#vegetables');
```

| **Clear button** | **Property:** *showResetIcon*


```
$('#autocomplete').ejAutocomplete({showResetIcon:true }) | Property: showClearButton <br/>var  
groupObj = new ej.dropdowns.AutoComplete({showClearButton:  
true,});groupObj.appendTo('#vegetables');
```

| **Animation** | **Property:** *animateType*


```
$('#autocomplete').ejAutocomplete({animateType:animateType }) | Not Applicable |
```

| **Focusing the list item** | **Property:** *AutoFocus*


```
@Html.EJ().Autocomplete("selectCar").AutoFocus("true") | Not applicable |
```

| **Delaying the popup open time** | **Property:** *delaySuggestionTimeout*


```
$('#autocomplete').ejAutocomplete({delaySuggestionTimeout:300 }) | Not applicable |
```

| **Popup text when there is no popup items** | **Property:** *emptyResultText*


```
$('#autocomplete').ejAutocomplete({emptyResultText:"no Records" })  
| https://ej2.syncfusion.com/javascript/demos/#/material/auto-complete/template.html |
```

| **Enable/disable the duplicate option** | **Property:** *enableDistinct*


```
$('#autocomplete').ejAutocomplete({enableDistinct:true })| Not applicable |
```

| **Popup height** | **Property:** *popupHeight*


```
$('#autocomplete').ejAutocomplete({popupHeight:300px }) | Property: popupHeight <br/> var  
groupObj = new ej.dropdowns.AutoComplete({popupHeight:  
300px,});groupObj.appendTo('#vegetables');
```

| **Popup Width** | **Property:** *popupWidth*


```
$('#autocomplete').ejAutocomplete({popupWidth:300px }) | Property: popupWidth <br/> var  
groupObj = new ej.dropdowns.AutoComplete({popupWidth:  
300px,});groupObj.appendTo('#vegetables');
```

| **Open popup** | **Method:** *open*
 \$('#autocomplete').ejAutocomplete("open"); | **Method:**

```
showPopup<br/>var groupObj = new  
ej.dropdowns.AutoComplete({});groupObj.appendTo('#vegetables');<br/><br/>groupObj.showP  
opup(); |
```

| **Close event** | **Event:** *close*
`$('#autocomplete').ejAutocomplete({close:"onClose" })` | **Event:** *close*
`var groupObj = new ej.dropdowns.AutoComplete({close:"close",});groupObj.appendTo('#vegetables');`|

| **Open event** | **Event:** *open*
`$('#autocomplete').ejAutocomplete({open:"onopen" })` | **Event:** *open*
`var groupObj = new ej.dropdowns.AutoComplete({open:"open",});groupObj.appendTo('#vegetables');`|

CSS

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Default** | **Property:** *cssClass*
`$('#autocomplete').ejAutocomplete({cssClass:"cssClass" })` | **Property:** *cssClass*
`var groupObj = new ej.dropdowns.AutoComplete({cssClass:"cssClass",});groupObj.appendTo('#vegetables');`|

| **Height** | **Property:** *height*
`$('#autocomplete').ejAutocomplete({height:"300px" })` | **Acheivable through the [cssClass](#) property** |

| **showRoundedCorner** | **Property:** *showRoundedCorner*
`$('#autocomplete').ejAutocomplete({showRoundedCorner:true })` | **Acheivable through the [cssClass](#) property.** |

| **Width** | **Property:** *width*
`$('#autocomplete').ejAutocomplete({width:300px })` | **Property:** *width*
`var groupObj = new ej.dropdowns.AutoComplete({width:"300px",});groupObj.appendTo('#vegetables');`|

| **Visibility** | **Property:** *visible*
`$('#autocomplete').ejAutocomplete({visible:true })` | **Acheivable through the [cssClass](#) property.** |

Grouping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Default** | **Property:** *fields*
`$('#autocomplete').ejAutocomplete({fields: { key: "index", groupBy: "name" },});`| **Property:** *fields*
`var groupObj = new ej.dropdowns.AutoComplete({fields: { groupBy: 'Category', value: 'Vegetable' },});groupObj.appendTo('#vegetables');`|

Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| **Default** | **Property:** *Locale*
`$('#autocomplete').ejAutocomplete({lcoale: "fr-FE",});`| **Property:** *Locale*
`var groupObj = new ej.dropdowns.AutoComplete({locale: "fr-FE"});groupObj.appendTo('#vegetables');`|

Template

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

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

| Default | Property: *template* `$('#autocomplete').ejAutocomplete({template: "${FirstName}${City}"})` **| Property:** *itemTemplate* `
 var groupObj = new ej.dropdowns.AutoComplete({itemTemplate: "${FirstName}${City}");groupObj.appendTo('#vegetables');` |

| Group Template | Not Applicable | Property: *groupTemplate* `
var groupObj = new ej.dropdowns.AutoComplete({groupTemplate: "${City}");groupObj.appendTo('#vegetables');` |

| ValueTemplate | Not applicable | Property: *valueTemplate* `
var groupObj = new ej.dropdowns.AutoComplete({valueTemplate: "${FirstName} - ${City}");groupObj.appendTo('#vegetables');` |

| Header Template | Not applicable | Property: *headerTemplate* `
 var groupObj = new ej.dropdowns.AutoComplete({headerTemplate: "NameCity");groupObj.appendTo('#vegetables');` |

| FooterTemplate | Not applicable | Property: *footerTemplate* `
var groupObj = new ej.dropdowns.AutoComplete({footerTemplate: " Total list items: "+ sportsData.length + "");groupObj.appendTo('#vegetables');` |

| No records Template | Not applicable | Property: *noRecordsTemplate* `
var groupObj = new ej.dropdowns.AutoComplete({noRecordsTemplate: " NO DATA AVAILABLE");groupObj.appendTo('#vegetables');` |

| Action failure Template | Not applicable | Property: *actionFailureTemplate* `
var groupObj = new ej.dropdowns.AutoComplete({actionFailureTemplate: " Data fetch get fails");groupObj.appendTo('#vegetables');` |

Sorting

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Default | Property: *allowSorting* `
 $('#autocomplete').ejAutocomplete({allowSorting: true});` **| Achievable through** [sortOrder](#) **property |**

| Order of sorting | Property: *sortOrder* `
$('#autocomplete').ejAutocomplete({sortOrder: "Ascending"});` **| Property:** *sortOrder* `
 var groupObj = new ej.dropdowns.AutoComplete({sortOrder: "sortOrder"});groupObj.appendTo('#vegetables');` |

Accessibility

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| RTL support | Property: *enableRtl* `
$('#autocomplete').ejAutocomplete({enableRtl: true});` **| Property:** *enableRtl* `
var groupObj = new ej.dropdowns.AutoComplete({enableRtl: true});groupObj.appendTo('#vegetables');` |

Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Selecting particular value | Property: *selectValueByKey*

`
$('#autocomplete').ejAutocomplete({selectValueByKey: 1,});` | Achievable through [value](#) property |

| Selecting particular value | Property: *value*`
$('#autocomplete').ejAutocomplete({value: data,});` | Property: *value*`
var groupObj = new ej.dropdowns.AutoComplete({value: "data"});groupObj.appendTo('#vegetables');`

| Selecting particular text | Property: *text*`
 $('#autocomplete').ejAutocomplete({text: "data",});` | Not Applicable |

| Selecting particular value | Method: *selectValueByKey*`
`

`$('#autocomplete').selectValueByKey("key")` | Achievable through [value](#) property |

| Selecting particular text | Method: *selectValueByText* `
`

`$('#autocomplete').selectValueByText("key")` | Not Applicable |

| Select event | Event: *select*`
$('#autocomplete').ejAutocomplete({select: "onSelect",});` | Event: *select* `
var groupObj = new ej.dropdowns.AutoComplete({select: "onSelect"});groupObj.appendTo('#vegetables');`

Miscellaneous

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Enable/disable | Property: *enabled*`
$('#autocomplete').ejAutocomplete({enabled: true,});` | Property: *enabled* `
var groupObj = new ej.dropdowns.AutoComplete({enabled: true});groupObj.appendTo('#vegetables');`

| Enable persistence | Property: *enablePersistence*`
`

`$('#autocomplete').ejAutocomplete({enablePersistence: true,});` | Property: *enablePersistence* `
var groupObj = new ej.dropdowns.AutoComplete({enablePersistence: true});groupObj.appendTo('#vegetables');`

| Loading icon | Property: *showLoadingIcon*

`
$('#autocomplete').ejAutocomplete({showLoadingIcon: true,});` | By default, it is showing |

| Read only | Property: *readOnly* `
 $('#autocomplete').ejAutocomplete({readOnly: true,});` |

Property: *readOnly* `
var groupObj = new ej.dropdowns.AutoComplete({readOnly: true});groupObj.appendTo('#vegetables');` |

| Disable | Method: *disable*`
 $('#autocomplete').ejAutoComplete("disable");` | Achievable through [enabled](#) property |

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Addition of new option watermark text** | **Property:** *addNewText*

\$('#autocomplete').ejAutocomplete({addNewText:"text"}); | **Not applicable** |

| **Addition of new item** | **Property:** *allowAddNew*

\$('#autocomplete').ejAutocomplete({allowAddNew: true}); | **Property:** *allowCustom*
 var groupObj = new ej.dropdowns.AutoComplete({allowCustom: true});groupObj.appendTo('#vegetables'); |

| **Reset the autocomplete** | **Property:** *showResetIcon*

\$('#autocomplete').ejAutocomplete({showResetIcon: true}); | **Property:** *showClearIcon*
 var groupObj = new ej.dropdowns.AutoComplete({allowCustom: true});groupObj.appendTo('#vegetables'); |

| **Destroy** | **Method:** *destroy*
 \$('#autocomplete').ejAutoComplete("destroy"); | **Method:**

destroy
var groupObj = new ej.dropdowns.AutoComplete({allowCustom: true});groupObj.appendTo('#vegetables');

groupObj.destroy(); |

| **Reset the autocomplete** | **Method:**

clearText
\$('#autocomplete').ejAutoComplete("clearText"); | **Property:** *value*
 var groupObj = new ej.dropdowns.AutoComplete({value: ""});groupObj.appendTo('#vegetables'); |

| **Multicolumn** | **Property:** *multiColumnSettings*
 var autocompleteInstance = new

ej.AutoComplete(\$('#selectCar'),
{multiColumnSettings:{enable:true,showHeader:true,stringFormat:"{1}",searchColumnIndices[0
*,1,2],
 columns:[{"field": "EmployeeID", "headerText": "EmployeeID"}, {"field": "FirstName",*
"headerText": "FirstName"}, {"field": "City", "headerText": "City"}]}); | **Not applicable** |

| **Hide the Autocomplete** | **Method:** *hide*
\$('#autocomplete').ejAutoComplete("hide"); | **Acheivable through the [cssClass](#) property.** |

| **Getting particular text** | **Method:** *getActiveText*

\$('#autocomplete').ejAutoComplete("getActiveText"); | **Not Applicable** |

| **Getting particular value** | **Method:** *getValue*

\$('#autocomplete').ejAutoComplete("getValue"); | **Acheivable through the [value](#) property.** |

| **Change event** | **Event:**

change
\$('#autocomplete').ejAutocomplete({change:"change"}); | **Event:** *change*
var groupObj = new ej.dropdowns.AutoComplete({change: "change"});groupObj.appendTo('#vegetables'); |

| **Create event** | **Event:** *create*
\$('#autocomplete').ejAutocomplete({create:"create"}); | **Event:**

created
var groupObj = new ej.dropdowns.AutoComplete({created: "created"});groupObj.appendTo('#vegetables'); |

| **Destroy event** | **Event:** *destroy*
\$('#autocomplete').ejAutocomplete({destroy:"destroy"}); |

| **Event:** *destroyed*
var groupObj = new ej.dropdowns.AutoComplete({destroyed: "destroyed"});groupObj.appendTo('#vegetables'); |

| **Focus out event** | **Event:**

focusOut
\$('#autocomplete').ejAutocomplete({focusOut:"focusOut"}); | **Event:** *blur*
var


```
groupObj = new ej.dropdowns.AutoComplete({blur:
"blur"});groupObj.appendTo('#vegetables'); |
| Focus in event | Event : focusIn<br/>$( '#autocomplete' ).ejAutocomplete({focusIn:"focusIn"}); |
Event: focus <br/>var groupObj = new ej.dropdowns.AutoComplete({focus:
"focus"});groupObj.appendTo('#vegetables'); |
```

Avatar

Types in EJ2 JavaScript 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.TS

--

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>
<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.TS

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>
  </script>
  <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.TS

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>
    </div>
  </div>

```

```

        <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 EJ2 JavaScript 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.TS

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.TS

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>
</body>
</html>

```

```

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.TS

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 class="e-card-content">
                <div>SVG</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 EJ2 JavaScript 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.TS

```
import { ListView } from '@syncfusion/ej2-lists';
// Listview datasource with avatar and image source fields
let dataSource: { [key: string]: Object; }[] = [
    { 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: '' }
];
let letterAvatarList: ListView = new 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>

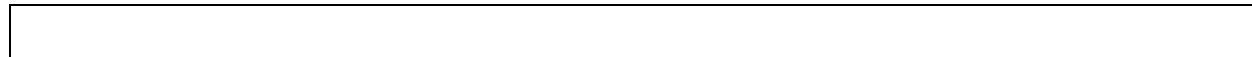
```

Integrate avatar into badge in EJ2 JavaScript 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.TS



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 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>
        </div>
    </div>
</div>

```

[illegible]

Badge

Types in EJ2 JavaScript 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.TS

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 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>
</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-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>
</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.TS

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>
</body>
</html>
```

```
        </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.TS

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">

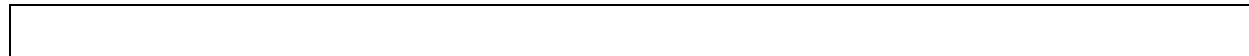
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

    <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.TS



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.TS

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.TS

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.TS

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.TS

```


```

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>
<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 EJ2 JavaScript Badge control

Colour customization

Even though badges come with **8 predefined colors**, you can also customize the colour of the badge as desired.

INDEX.TS



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>
</body>
<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.TS

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>

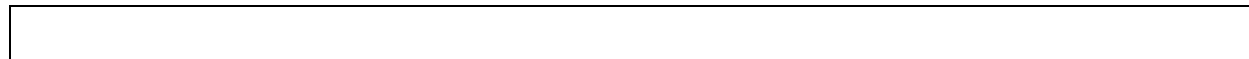
  <div id="container">
    <div id="element">
      <h1>Badge Component <span class="e-badge e-badge-primary
size_1">New</span></h1>
      <h1>Badge Component <span class="e-badge e-badge-primary
size_2">New</span></h1>
      <h1>Badge Component <span class="e-badge e-badge-primary
size_3">New</span></h1>
    </div>
  </div>
  <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

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.TS



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>
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 EJ2 JavaScript 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.TS

```

import { ListView } from '@syncfusion/ej2-lists';
// Datasource for listview, badge field is the class data for Badges
let dataSource: { [key: string]: Object }[] = [
    { 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' },
  ];
let list: ListView = new 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: () => {
    let list: HTMLElement =
<HTMLElement>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 EJ2 JavaScript 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.TS

```

import { ListView } from '@syncfusion/ej2-lists';
let badgeElements: HTMLElement[];
// Datasource for listview, badge field is the class data for Badges
let dataSource: { [key: string]: Object }[] = [
    { 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' },
];
let list: ListView = new 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>

```

```
</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 EJ2 JavaScript 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.

```
`javascript
|-- @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.

```
`javascript
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';
`

```

To know about individual component CSS, please refer to [Individual Component theme files](#) section.

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.TS

```
import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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%;
      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-->
<script src="node_modules/systemjs/dist/system.src.js" type="text/javascript"></script>
<script src="system.config.js" type="text/javascript"></script>
</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.TS

```
import { QRCodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new QRCodeGenerator({
    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>
```

Adding Datamatrix Generator control

You can add the datamatrix code in our barcode generator component.

INDEX.TS

```
import { DataMatrixGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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 EJ2 JavaScript 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.TS

```
import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';

```

```
let barcode = new 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>
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.TS

```
import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```
import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```
import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { BarcodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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;
    }
</style>

```

```

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 EJ2 JavaScript 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.TS

```

import { QRCodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { QRCodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { QRCodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { QRCodeGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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 EJ2 JavaScript 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.TS

```

import { DataMatrixGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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.TS

```

import { DataMatrixGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new DataMatrixGenerator({
    width: '200px',
    height: '150px',
    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.TS

```

import { DataMatrixGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new DataMatrixGenerator({
  width: '100px',
  height: '100px',
  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 text

In barcode generators you can customize the barcode text by using the display text property .

INDEX.TS

```

import { DataMatrixGenerator, ValidateEvent } from '@syncfusion/ej2-barcode-generator';
let barcode = new 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">

<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>

```

Export in EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
new 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
new 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>
<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.TS

```
import { BreadcrumbItemModel, Breadcrumb,
BreadcrumbBeforeItemRenderEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
new Breadcrumb({
  enableNavigation: false,
  url: "https://ej2.syncfusion.com/demos/breadcrumb/bind-to-location",
  beforeItemRender: (args: BreadcrumbBeforeItemRenderEventArgs) => {
    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 EJ2 JavaScript 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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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

By default, the icon is positioned to the left side of the item in the Breadcrumb component. If you need to add the icon to right of the breadcrumb item, add the `e-icon-right` class to the required item. In the following example, the `e-icon-right` class was added to the breadcrumb items using the [beforeItemRender](#) event. You can also add the `e-icon-right` class to the [cssClass](#) property of the breadcrumb component to position the icons to the right of all breadcrumb items.

INDEX.TS

```

import { BreadcrumbItemModel, Breadcrumb,
BreadcrumbBeforeItemRenderEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        text: "Program Files",
        iconCss: "e-icons e-folder"
    },
    {
        text: "Services",
        iconCss: "e-icons e-folder"
    },
    {
        text: "Config.json",
        iconCss: "e-icons e-file"
    }
];
new Breadcrumb({
    items: items,
    enableNavigation: false
}, '#breadcrumb');
new Breadcrumb({
    items: items,

```

```

enableNavigation: false,
beforeItemRender: (args: BreadcrumbBeforeItemRenderEventArgs) => {
    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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        iconCss: 'e-icons e-home'
    },
    {
        iconCss: "e-icons e-folder"
    },
    {
        iconCss: "e-icons e-file"
    }
];
new 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>
</body>
<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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
  {
    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 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 EJ2 JavaScript 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

The [url](#) property of the items contains a portion of the full path which is based on its relation to the current path where it is linked. In the following example, the items represent only the relative URL path.

INDEX.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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

The [url](#) property of the items contains the full path or entire address of the page. In the following example, the items represent absolute URL.

INDEX.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb,
BreadcrumbBeforeItemRenderEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 Breadcrumb({
    items: items,
    beforeItemRender: (args: BreadcrumbBeforeItemRenderEventArgs) => {
        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 EJ2 JavaScript 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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
const items: BreadcrumbItemModel[] = [
    {
        text: 'Cart'
    },
    {
        text: 'Billing'
    },
    {
        text: 'Shipping'
    },
    {
        text: 'Payment'
    }
]

```



```

];
new 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
const items: BreadcrumbItemModel[] = [
    {
        text: 'Cart'
    },
    {
        text: 'Billing'
    },
    {
        text: 'Shipping'
    },
    {
        text: 'Payment'
    }
];
new 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>
<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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-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 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 EJ2 JavaScript 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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition

```

```
let items: BreadcrumbItemModel[] = [
    {
        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 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>
<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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
    {
        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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-navigations';  
import { enableRipple } from '@syncfusion/ej2-base';  
enableRipple(true);  
//Breadcrumb items definition  
let items: BreadcrumbItemModel[] = [  
    {  
        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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
  {
    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 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.TS

```
import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition
let items: BreadcrumbItemModel[] = [
  {
    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 Breadcrumb({
  items: items,
```

```

enableNavigation: false,
maxItems: 3,
overflowMode: 'Scroll'
}, '#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.TS

```

import { BreadcrumbItemModel, Breadcrumb } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Breadcrumb items definition

```

```
let items: BreadcrumbItemModel[] = [
    {
        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 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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

Bullet Chart

Bullet chart dimensions in EJ2 JavaScript 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.

```
`javascript
<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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
    //width and height for chart in pixel
    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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
    // Width and height for chart in percentage
    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 EJ2 JavaScript 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
    //color and width customization of the major and minor ticks
    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>
</body>
</html>
```

```

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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
    // To place the ticks inside of the bullet-chart
    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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
  //Label format as currency
  labelFormat: 'c',
  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>

```

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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
  // To place the axis label inside of the bullet-chart
  labelPosition: 'Inside',
  dataSource: [{ value: 23, target: 22 }],
  animation: { enable: false },
  valueField: 'value',
  targetField: 'target',
  ranges: [{ end: 20 },
    { end: 25 },
    { end: 30 }
  ],
  minimum: 0, 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
  opposedPosition: true,
  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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let localData: any[] = [
    {
        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'
    }
];
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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>

```

Subtitle

To show additional information about the data plotted, the Bullet Chart can also be given a subtitle using the `subtitle` property.

INDEX.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
  title: 'Sales Rate',

```

```

        dataSource: [
            { value: 55, target: 45, category: 'Year 1' },
        ],
        animation: { enable: false },
        targetField: 'target',
        valueField: 'value',
        labelFormat: '${value}',
        titleStyle: { size: '22px', color: 'red', fontFamily: 'cursive',
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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({

```

```

        title: 'Sales Rate',
        dataSource: [
            { value: 55, target: 45, category: 'Year 1' },
        ],
        animation: { enable: false },
        targetField: 'target',
        valueField: 'value',
        labelFormat: '${value}',
        subtitleStyle: { size: '22px', color: 'red', fontFamily: 'cursive',
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 EJ2 JavaScript 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS


```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let localData: any[] = [

```

```

    {
        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'
    }
];
let chart: BulletChart = new 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.TS

```
import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let localData: any[] = [
  {
    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'
  }
];
let chart: BulletChart = new 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 EJ2 JavaScript 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);

```

```
let chart: BulletChart = new 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,
  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="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>
```

```

        </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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new 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.TS

```

import { BulletChart, BulletTooltip } from '@syncfusion/ej2-charts';
BulletChart.Inject(BulletTooltip);
let chart: BulletChart = new BulletChart({
  height: '110px',
  tooltip: { enable: true, fill: 'red', border:{ color: 'green',
width: 10 } },
  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="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 EJ2 JavaScript 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 EJ2 JavaScript controls](#)

ButtonGroup

Getting started in EJ2 JavaScript 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](https://www.npmjs.com) public registry. You can install all Syncfusion JavaScript (Essential JS 2) controls in a single [@syncfusion/ej2](https://www.npmjs.com/package/@syncfusion/ej2) package or individual packages for each control.

The quickstart application is preconfigured with the dependent [@syncfusion/ej2](https://www.npmjs.com/package/@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>
</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

```
#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>
</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;
    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 EJ2 JavaScript 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>
```

```

    </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 EJ2 JavaScript 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>
<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;
}

```

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.TS

```
import { DropDownButton, ItemModel, DropDownButtonModel } from
'@syncfusion/ej2-splitbuttons';
let items: ItemModel[] = [
{
    text: 'Learn SQL'
},
{
    text: 'Learn PHP'
},
{
    text: 'Learn Bootstrap'
}];
let menuOptions: DropDownButtonModel = {
    items: items
};
// To render DropDownButton.
let btnObj: DropDownButton = new 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.TS

```
import { SplitButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
let items: ItemModel[] = [
  {
    text: 'Paste'
  },
  {
    text: 'Paste Text'
  },
  {
    text: 'Paste Special'
  }
];
// To render SplitButton.
let btnObj: SplitButton = new 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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

How To

Create buttongroup with icons in EJ2 JavaScript 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>
<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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```

import { rippleEffect, enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// To enable ripple in ButtonGroup.
let button: HTMLElement = document.querySelector('.e-btn-group');
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>
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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { createButtonGroup } from '@syncfusion/ej2-splitbuttons';
// To create basic ButtonGroup.
createButtonGroup('#basic', {
    buttons: [
        { content: 'HTML' },
        { content: 'CSS' },
    ],
});
```

```

        { content: 'Javascript' }
    ]
});
// To create checkbox type ButtonGroup.
createButtonGroup('#checkbox', {
    buttons: [
        { content: 'Bold' },
        { content: 'Italic' },
        { content: 'Undeline' }
    ]
});
// To create radio type ButtonGroup.
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="checkbox">
      <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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Primary Button - Used to represent a primary action.
let button: Button = new Button({ cssClass: `e-primary`, '#primarybtn' });
//Success Button - Used to represent a positive action.
button = new Button({ cssClass: `e-success`, '#successbtn' });
//Info Button - Used to represent an informative action.
button = new Button({ cssClass: `e-info`, '#infobtn' });
//Warning Button - Used to represent an action with caution.
button = new Button({ cssClass: `e-warning`, '#warningbtn' });
//Danger Button - Used to represent a negative action.
button = new Button({ cssClass: `e-danger`, '#dangerbtn' });
//Link Button - Changes the appearance of the Button like a hyperlink.
button = new 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Submit.
let button: Button = new Button();
button.appendTo('#submitbtn');
//Reset.
button = new 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Flat Button.
let button: Button = new Button({ cssClass: `e-flat`, '#flatbtn' });
//Outline Button.
button = new 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 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.TS

```
//Button is active in toggled state.
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let togglebtn: Button = new Button({cssClass: `e-flat`, iconCss: 'e-btn-sb-icon e-play-icon', isToggle: true, content: 'Play'}, '#togglebtn');
//Click Event.
document.getElementById('togglebtn').onclick = (): void => {
    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="toggelbtn"></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,AAEAAAKAIAAAwAgT1MvMj1tSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYaGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQYN6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaawAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE

```

```

fDyE/DxEvDyEPDgwBAGMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCacFBQmCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBAArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAgIEBAoBAQEECgChBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICacLBaIFawIEBAQFBQcGBwgHBgY
GBgoJCAYCagEBAQFGMRkaGw0NDA0LIh4xBAQCBAEBagADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmZ8KNScxBxEfDjsSBHQefDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFeBAMDawIBAQGL7Y0EAwQCAgIBAYYKChEQDQsJCACBAU
CYt8BAQIDBAUFBgCHBwgICQgKjQECagMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQBAGI
DBAQFBQYGBgcHBwgMAQMDawUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDawTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQAeAwI
CUAGHBwcGBgYFBQQAeAwMBAgIBAwMEBAUFBgGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAIAAAAAA6cD9AADAawAADchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EWMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAAAAABAAAAAABAAgAAQABAAAAA
CAACACQABAAAAAADAAGAEABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIACQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAKQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAyGBOAG4ALQBpAGM
ABwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBAGMAZgB1AHM
AaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAA
GAQcBCAEJAaptZWRpYS1wbGF5C21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmLQAA) format('trueType');

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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//To position the icon to the left of the text on a Button.
let button: Button = new 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 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,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXDNH+dzAAABoAAAAEJnbHlmlv4
8pAAAAfgAAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAYAAAAA
gbG9jYQYN6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l071FxAABhAAAAIxcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAAABAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBagMFBQcICQkLCwMDQ4NatoNDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAgMFBQcICQkLCwsNDQ0
OAtOoDQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwg3NT8INS8FNT8NNS8JByU/BDUvCyMPAQytrQH5AgoEAQEBArg
hERESEyIJSgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAge
BAxYLBQQEAwMCAGIEBAoBAQEECgCHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwkEawIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmz8KNScxBxEfdjsBHQEfdTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EawQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgcHBwgMAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAgMDAwTV+94BAQECAwMDBAGyAQECBAYHCAgJCgkQCaQC6/47CQkICQcIBwYGBQQEAwI
CUAgHBwcGBgYFBQQAeAwMBAgIBAwMEBAUFBQcGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBAQDAgEBAAAAAIAAAAAA6cD9AADAaAAdchNSElAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAAEAAGAGAABAAAAAFAAASAIABAAAAAAGAAGAKwABAAA
AAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIACQADAAEECQABABAAcwADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMabwBuAFIAZQBnAHUAbABhAHIAyAgB0AG4ALQBpAGM
AbwBuAGIAAdABUAC0AAQBJAG8ABgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMabwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAyAgB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHM
AaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQB
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;
}
/*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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let button: Button = new 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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Small Button.
let button: Button = new Button({ cssClass: `e-small`, '#smallbtn' });
//Normal Button.
button = new 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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

How To

Add link to a button in EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize the Button component.
let button: Button = new Button({cssClass: 'e-link'});
// Render initialized button.
button.appendTo('#element');
button.element.onclick = (): void => {
    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 EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Block Button.
let button: Button = new Button({cssClass: `e-block`});
button.appendTo('#blockbtn');
//Primary Block Button.
button = new Button({ isPrimary: true, cssClass: `e-block` });
button.appendTo('#primarybtn');
//Success Block Button.
button = new 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 EJ2 JavaScript 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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//To customize Button appearance.
//Refer the "e-custom" class details in "styles.css".
let button: Button = new 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,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXDnEOdVAAABiAAAADZnbHlm7/n

```



```

pHAAAAcgAAAEwaGVhZBKtDIMAAADQAAAAANmhoZWEHmQNrAAAArAAAACRobXR4B+gAAAAAYAAAA
IbG9jYQCYAAAAAHAAAAABmlheHABDgB0AAABCAAAACBuYW1lcmFdywAAAvGAAAIxcG9zdDwSCic
AAAUAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAMLNyX18
PPPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAACAGgAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAQP0AZAABQAAAanoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAmAAAAAIAAAAA+oD6gALAGcAAAEAOAQcuASc+ATceAQEVBgcJiIPAQYUHWEOAQcJgYdAR4
BOwEWFwcGFB8BFjI/AR4BFxUeATsBPgE9ATY3FxYyPwE2NC8BPgE3MzI2PQE0JisBJic3NjQvASY
iDwEuASc1LgEnIw4Bar4CcVVUcQMDcVRVcf7pOTRbBRQGUQYgWhAYB30LDgEPCX00IVoHB1EFFQV
bGTyEAQ4KcQsOOTRbBRQGUQYgWhAXB34LDQ8Jfg4gWgcHUQUVBVszNh4BDgpxCQ8B9lRxAwNxVfV
xAgJxAyd+DiBaBgZRRUFWBk2Hg8KcQsOOTRbBxQHUQYgWhAYB30LDgEPCX00IVoGBk4FFQVbGTy
eDwpxCw06NFoIEwhRBgZaEBgHfQsNAQENAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAAgAAQA
BAAAAAAAACAAcACQABAAAAAADAAgAEAAABAAAAAAAEAAgAGAABAAAAAAAFAAAsIAABAAAAAAAGAAG
AKwABAAAAAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIACQADAAEECQABABAAcwADAAEECQA
CAA4AgwADAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAE
ECQAKAFgAlwADAAEECQALACQBLyBzZXR0aW5nc1JlZ3VsYXJzZXR0aW5nc3NldHRpbmdzVmVyc2l
vbiAxLjBzZXR0aW5nc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1
vd3d3LnN5bmNmdXNpb24uY29tACAACwB1AHQAdABpAG4AZwBzAFIAZQBnAHUAbABhAHIAcwB1AHQ
AdABpAG4AZwBzAHMAZQB0AHQAaQBuaGcAcwBWAUAcgBzAGkAbwBuACAAMQAuADAACwB1AHQAdAB
pAG4AZwBzAEYAbwBuAHQAIAbnAGUAbgB1AHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAEQBuAGM
AZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgB
1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAaAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIA
DAA1zZXR0aW5ncy0tLTExAAAA) 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { EventHandler, enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let timeout: any = null;
let button: Button = new Button({ cssClass: 'e-small' });
button.appendTo('#button');
let clear: Button = new Button({ cssClass: 'e-small' });
clear.appendTo('#clear');
clear.element.onclick = () => {
  document.getElementById('eventlog').innerHTML = '';
}

```

```

button.element.addEventListener("mousedown", mouseDownHandler);
button.element.addEventListener("touchstart", mouseDownHandler);
button.element.addEventListener("mouseup", mouseUpHandler);
button.element.addEventListener("touchend", mouseUpHandler);
button.element.addEventListener("click", clickEventHandler);
function mouseUpHandler(event: any): void {
    clearInterval(timeout);
}
function mouseDownHandler(event: any): void {
    event.preventDefault();
    timeout = setInterval(clickEventHandler, 200);
}
function clickEventHandler() {
    EventHandler.trigger(button.element, "click");
    appendSpanElement('Button click event triggered.<hr>');
}
function appendSpanElement(text: string): void {
    let span: HTMLElement = document.createElement('span');
    span.innerHTML = text;
    let log: HTMLElement = document.getElementById('eventlog');
    log.insertBefore(span, log.firstChild);
}

```

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 EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let button: Button = new 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,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXDNdEOdVAAABiAAAADZnbHlm7/n
pHAAAAcGAAAEwaGVhZBKTdIMAADQAAAAANmhoZWEHmQNrAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQCYAAAAAHAAAAABmlheHABDgB0AAABCAAAACBuYW1lcmFdywAAAvGAAAIxcG9zdDwSCic
AAAUAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAgABAAAAQAAMLNyX18
PPPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAACAGgAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAwGpQaJYAAAABAAAAAAAAABAAAAAPoAAA
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAmAAAAAIAAAAAA+oD6gALAGcAAAEAOAQcuASc+ATceAQEVBgcnJiIPAQYUHWEOAQcjIgYdAR4
BOWEWFwcGFB8BFjI/AR4BFxUeATsBPgE9ATY3FxYyPwE2NC8BPgE3MzI2PQE0JisBJic3NjQvASY
iDwEuASclLgEnIw4BAr4CcVVUCQMDcVRVcf7pOTRbBRQGUQYGWWhAYB30LDgEPCX00IvOHB1EFFQV
bGTyEAQ4KcQsOOTRbBRQGUQYGWWhAXB34LDQ8JfG4gWgCHUQUVBVsZnH4BDGpxCQ8B9lRxAwNxFV
xAgJxAYd+DiBaBgZRBRUFWBk2Hg8KcQsOOTRbBxQHUQYGWWhAYB30LDgEPCX00IvOGBk4FFQVbGTy
eDwpxCw06NFoIEwhRBgZaEBgHfQsNAQENAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAAGAAQA
BAAAAAAACAACACQABAAAAAADAAgAEABAAAAAAAEAAgAGAABAAAAAAAFaAsAIAABAAAAAAGAAG
AKwABAAAAAAAKACwAMwABAAAAAALABIAxwADAAEECCQAAAAIACQADAAEECCQABABAAcWADAAEECCQ
CAA4AgwADAAEECCQADABAAkQADAAEECCQAEABAAoQADAAEECCQAFABYAsQADAAEECCQAGABAAxwADAAE
ECQAKAFgAlwADAAEECCQALACQBLyBzZXR0aW5nc1JlZ3VsYXJzZXR0aW5nc3NldHRpbmdzVmVyc2l
vbiAxLjBzZXR0aW5nc0Zvb3N0Z2VzZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZG1
vd3d3LnN5bmNmdXNpb24uY29tACAACwBlAHQAdABpAG4AZwBzAFIAZQBnAHUAbABhAHIAcWBlAHQ
AdABpAG4AZwBzAHMAZQB0AHQAaQBuAGcAcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAcWBlAHQAdAB
pAG4AZwBzAEYAbwBuAHQAIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBAGM
AZgBlAHMAaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBAGMAZgB
lAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIBAgE
DAA1zZXR0aW5ncy0tLTExAAAA) 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";
}
```

Set the disabled state in EJ2 JavaScript 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.TS

```
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let button: Button = new 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,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXdDnEOdVAAABiAAAADZnbHlm7/n
pHAAAACgAAAEwaGVhZBktDIMAAADQAAAAANmhoZWEHmQNrAAAArAAAACRobXR4B+gAAAAAYAAAAA
IbG9jYQCYAAAAAHAAAAABmlheHABDgB0AAABCAAAACBuYW11xmFdywAAAvGAAAIxcG9zdDwSCic
AAAUsAAAAANwABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAgABAAAAQAAMLNyX18
PPPUACwPoAAAAANfSY9oAAAAA19Jj2gAAAAAD6gPqAAAAACAACAAAAAAAAAAEAAAAACAGgAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
AAAACAAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAmAAAAAIAAAAAA+oD6gALAGcAAAEOAQcuASc+ATceAQEVBgcnJiIPAQYUHWEOAQcjIgYdAR4

```



```

BOWEFwCgFB8BFjI/AR4BFxUeATsBPgE9ATY3FxYyPwE2NC8BPgE3MzI2PQE0JisBJic3NjQvASY
iDwEuASclLgEnIw4Bar4CcVVUCQMDcVRVcf7pOTRbBRQGUQYGWhAYB30LDgEPCX0OIVoHB1EFFQV
bGTyEAQ4KcQsOOTRbBRQGUQYGWhAXB34LDQ8Jfg4gWgcHUQUVBVsZNh4BDgpxCQ8B9lRxAwNxVfV
xAgJxAYd+DiBaBgZRBRUFWBk2Hg8KcQsOOTRbBxQHUQYGWhAYB30LDgEPCX0OIVoGBk4FFQVbGTy
eDwpxCw06NFoIEwhRBgZaEBgHfQsNAQENAAAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAgAAQA
BAAAAAACAACACQABAAAAAADAAGAEAAABAAAAAEEAAGAGAABAAAAAFAAsAIAABAAAAAAGAAG
AKwABAAAAAAKACwAMwABAAAAAALABIAXwADAAEECQAAAAIAcQADAAEECQABABAAcwADAAEECQA
CAA4AgwADAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAE
ECQAKAFgAlwADAAEECQALACQBLyBzZXR0aW5nc1JlZ3VsYXJzZXR0aW5nc3NldHRpbmdzVmVyc2l
vbiAxLjBzZXR0aW5nc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZG1
vd3d3LnN5bmNmdXNpb24uY29tACAACwBLAHQAdABpAG4AZwBzAFIAZQBnAHUAbABhAHIAcWBLAHQ
AdABpAG4AZwBzAHMAZQB0AHQAaQBwAGcAcwBWAGUAcgBzAGkAbwBuACAAMQAuADAACwBLAHQAdAB
pAG4AZwBzAEYAbwBuAHQAIAbnAGUAbgBLAHIAyQB0AGUAZAAGAHUAACwBpAG4AZwAgAFMAeQBwAGM
AZgB1AHMAaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB
1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIBAgE
DAA1zZXR0aW5ncy0tLTExAAAA) 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 EJ2 JavaScript 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.TS

```

import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize the Button component.
let button: Button = new Button({isPrimary: true, content: 'Button'});
// Render initialized 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%;
}

```

Ej1 api migration in EJ2 JavaScript Button control

This article describes the API migration process of Button component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Specifies the text of the button | **Property:** *text*

 \$("#button").ejButton({
 text: "Button"
}); | **Property:** *content*

 var button = new ej.buttons.Button({
content: "Button"
});
button.appendTo("#button"); |

| Specifies the content type of the button | **Property:** *ContentType*
`$("#button").ejButton({contentType: ej.ContentType.TextAndImage, prefixIcon: "e-icon e-save", text: "Save" });` | Not applicable |

| Specifies icon for the button | **Property:** *prefixIcon*
`$("#button").ejButton({contentType: ej.ContentType.ImageOnly, prefixIcon: "e-icon e-save" });` | **Property:** *iconCss*
`var button = new ej.buttons.Button({iconCss: "e-icons e-save" });`
`button.appendTo("#button");` |

| Positioning icon in the button | **Property:** *imagePosition*
`$("#button").ejButton({imagePosition: ej.ImagePosition.ImageRight, contentType: ej.ContentType.TextAndImage, prefixIcon: "e-icon e-save", text: "Save" });` | **Property:** *iconPosition*
`var button = new ej.buttons.Button({iconCss: "e-icons e-save", iconPosition: "Right", content: "Save" });`
`button.appendTo("#button");` |

| Specifies secondary icon for the button | **Property:** *suffixIcon*
`$("#button").ejButton({contentType: ej.ContentType.ImageTextImage, suffixIcon: "e-icon e-file-html", prefixIcon: "e-icon e-search", text: "FileSearch" });` | Not applicable |

| Specifies the size of the button | **Property:** *size*
`$("#button").ejButton({size: ej.ButtonSize.Small, text: "Button" });` | **Property:** *cssClass*
`var button = new ej.buttons.Button({cssClass: "e-small", content: "Button" });`
`button.appendTo("#button");` |

| Adding custom css class | **Property:** *cssClass*
`$("#button").ejButton({cssClass: "custom-class", text: "Button" });` | **Property:** *cssClass*
`var button = new ej.buttons.Button({cssClass: "custom-class", content: "Button" });`
`button.appendTo("#button");` |

| Triggers click event repeatedly while pressing the button | **Property:** *repeatButton*
`$("#button").ejButton({repeatButton: true, text: "Button" });` | Not applicable |

| Sets time interval between two consecutive click event on the repeat button | **Property:** *timeInterval*
`$("#button").ejButton({repeatButton: true, timeInterval: "100", text: "Button" });` | Not applicable |

| Specifies the type of the button | **Property:** *type*
`$("#button").ejButton({type: ej.ButtonType.Submit, text: "Submit" });` | Not applicable |

| Changes normal button into rounded corner | **Property:** *showRoundedCorner*
`$("#button").ejButton({showRoundedCorner: true, text: "Button" });` | Not applicable |

| Specifies the width of the button | **Property:** *width*
`$("#button").ejButton({width: "150px", text: "Button" });` | Not applicable |

| Specifies the height of the button | **Property:** *height*
`$("#button").ejButton({height: "30px", text: "Button" });` | Not applicable |

| Adds HTML attributes to the button | **Property:** *htmlAttributes*
`$("#button").ejButton({htmlAttributes: { disabled: "disabled" }, text: "Button" });` | Not applicable |

| Allows the appearance of the Button to be enhanced and visually appealing | Not applicable | **Property:** *isPrimary*
`var button = new ej.buttons.Button({isPrimary: true, content: "Button" });`
`button.appendTo("#button");` |

| Makes the button toggle from normal to active state | Not applicable | **Property:** *isToggle*

 var button = new ej.buttons.Button({
isToggle: true,
content: "Button"
});

button.appendTo("#button"); |

| Specifies the disabled state of the button | **Property:** *enabled*

 \$("#button").ejButton({

enabled: false,
text: "Button"
}); | **Property:** *disabled*

 var button = new
 ej.buttons.Button({
 disabled: true,
 content: "Button"
});

 button.appendTo("#button"); |

| Enable or disable rendering component in right to left direction | **Property:** *enableRTL*

 \$("#button").ejButton({
 enableRTL: true,
 text: "Button"
}); | **Property:** *enableRtl*

 var button = new ej.buttons.Button({
enableRtl: true,
content:
 "Button"
});
 button.appendTo("#button"); |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Destroys the control | **Methods:** *destroy*

 \$("#button").ejButton({
text: "Button"

});
 var button = \$("#button").data("ejButton");
button.destroy(); | **Methods:** *destroy*

 var button = new ej.buttons.Button({
content: "Button"
});

 button.appendTo("#button");
button.destroy(); |

| Disables the button | **Methods:** *disable*

 \$("#button").ejButton({
 text: "Button"

 });
 var button = \$("#button").data("ejButton");
 button.disable(); | Not applicable |

| Enables the button | **Methods:** *enable*

 \$("#button").ejButton({
enabled: false,

text: "Button"
});
 var button = \$("#button").data("ejButton");
button.enable(); |
 Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Triggers while clicking the button | **Events:** *click*

 \$("#button").ejButton({
text:
 "Button",
click: function(args) { / code block */ }
}); | Not applicable |

| Triggers once the component rendering is completed | **Events:** *create*

 \$("#button").ejButton({
text: "Button",
create: function(args) { / code block / }
}); |
Events: *created*

 var button = new ej.buttons.Button({
content:
 "Button",
created: function created() { / code block / }
});
 button.appendTo("#button");
 |

| Triggers once the component is destroyed | **Events:** *destroy*

 \$("#button").ejButton({

text: "Button",
destroy: function(args) { / code block */ }
}); | Not applicable |

Calendar

Date range in EJ2 JavaScript 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.TS

```
import { Calendar, ChangedEventArgs } from '@syncfusion/ej2-calendars';
//Creates a Calendar with min and max properties.
let today: Date = new Date();
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
let calendarObject: Calendar = new 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))
});
calendarObject.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 EJ2 JavaScript 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.TS

```

import { Calendar } from '@syncfusion/ej2-calendars';
/*Initialize the calendar component*/
let calendar: Calendar = new 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>
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 EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';  
//import the loadCldr from ej2-base  
import { loadCldr, L10n } from '@syncfusion/ej2-base';  
//load the CLDR JSON data files.  
import * as numberingSystems from './numberingSystems.json';  
import * as gregorian from './ca-gregorian.json';  
import * as numbers from './numbers.json';  
import * as timeZoneNames from './timeZoneNames.json';  
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
```

```

L10n.load({
  'de': {
    'calendar': {
      today: 'heute'
    }
  }
});
//creates a calendar with German culture.
let calendarObject: Calendar = new Calendar({ locale: 'de' });
calendarObject.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>

```

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.TS

```
//imports the loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { Calendar } from '@syncfusion/ej2-calendars';
//loads the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
```

```
'ar': {
    'calendar': { today: 'اليوم' }
};
});
//creates the calendar with Arabic culture.
let calendarObject: Calendar = new Calendar({
    //sets the locale
    locale: 'ar',
    //sets the enableRtl
    enableRtl: true
});
calendarObject.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">
    <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 EJ2 JavaScript 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.TS

```
import { Calendar, RenderDayCellEventArgs } from '@syncfusion/ej2-
calendars';
//Creates a calendar with weekends disabled.
let calendarObject: Calendar = new Calendar({
    renderDayCell: disabledDate,
    value: new Date()
});
calendarObject.appendTo('#element');
function disabledDate(args: RenderDayCellEventArgs): void {
    if (args.date.getDay() === 0 || args.date.getDay() === 6) {
        //Set 'true' to disable the weekends.
        args.isDisabled = true;
    }
}
```

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.TS

```

import { Calendar, RenderDayCellEventArgs } from '@syncfusion/ej2-
calendars';

```

```
//Creates a Calendar with World Health Day and World Forest Day highlighted.
let calendarObject: Calendar = new Calendar({
    renderDayCell: customDates,
    value: new Date('03/10/2017')
});
calendarObject.appendTo('#element');
function customDates(args: RenderDayCellEventArgs): void {
    let span: HTMLElement;
    //Defines the custom HTML element to be added.
    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!');
    }
}
}
```

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>

```

Highlight weekends

You can highlight the weekends of every month in a Calendar by using the [renderDayCell](#) event. The following example demonstrates how to highlight the weekends of every month.

INDEX.TS

```

import { Calendar, RenderDayCellEventArgs } from '@syncfusion/ej2-
calendars';
//Creates a calendar with weekends highlighted.
let calendarObject: Calendar = new Calendar({
    renderDayCell: highlightDate,
    value: new Date()
});
calendarObject.appendTo('#element');
function highlightDate(args: RenderDayCellEventArgs): void {
    if (args.date.getDay() === 0 || args.date.getDay() === 6) {
        // To highlight the week end of every month
        args.element.classList.add('e-highlightweekend');
    }
}

```

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 EJ2 JavaScript 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.TS

```
import { Calendar, ChangedEventArgs } from '@syncfusion/ej2-calendars';
//creates a calendar with year view.
let calendarObject: Calendar = new Calendar({
    //sets the start view.
    start: "Year",
    //sets the value.
    value: new Date()
});
calendarObject.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.TS

```
import { Calendar, ChangedEventArgs } from '@syncfusion/ej2-calendars';
```

```
//creates a calendar with decade view and navigates up to year view.
let calendarObject: Calendar = new Calendar({
    //sets the start view.
    start: "Decade",
    //sets the depth view.
    depth: "Year"
});
calendarObject.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>
```

Accessibility in EJ2 JavaScript 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-activedescendent:** 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
```

```
// creates a Calendar component.
let calendarObj: Calendar = new Calendar();
calendarObj.appendTo('#element');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84) {
        // press alt+t to focus the control.
        (<HTMLElement>calendarObj.element.querySelectorAll('.e-content
table')[0]).focus();
    }
};
```

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 EJ2 JavaScript 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';\n\nCalendar.Inject(Islamic);
```

The following example demonstrates how to display the Islamic Calendar (Hijri Calendar).

INDEX.TS

```
import { Calendar, Islamic, RenderDayCellEventArgs } from '@syncfusion/ej2-calendars';
import { addClass } from '@syncfusion/ej2-base';
// 'Islamic' module injection
Calendar.Inject(Islamic);
//creates a calendar with islamic mode.
let calendarObject: Calendar = new Calendar({
    // To display the Islamic calendar
    calendarMode: 'Islamic',
    renderDayCell: customDates
});
calendarObject.appendTo('#element');
function customDates(args: RenderDayCellEventArgs): void {
    /*Date need to be disabled*/
    if ( args.date.getDate() === 12 || args.date.getDate() === 17 ||
args.date.getDate() === 28) {
        args.isDisabled = true;
    }
    /*Dates need to be customized*/
    if (args.date.getDate() === 13) {
        let span: HTMLElement;
        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) {
```



```

let span: HTMLElement;
span = document.createElement('span');
args.element.children[0].className += 'e-day sf-icon-start highlight';
span.setAttribute('class', 'sx !');
//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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
// creates a Calendar with today and clear buttons.
let calendarObject: Calendar = new Calendar({
    created: onCreate
});
calendarObject.appendTo('#element');
function onCreate() {
    //creates the custom element for clear button.
    let clearBtn: HTMLElement = document.createElement('button');
    let footerElement: HTMLElement = 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 current date
object.
document.querySelector('.e-footer-container .e-
clear').addEventListener('click', function () {
    calendarObject.value = new Date();
    calendarObject.dataBind();
});
```

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 EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';  
//creates a calendar with dates of other months shown.  
let calendarObject: Calendar = new Calendar({  
});  
calendarObject.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 EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
import moment from "moment";
/*Initialize the calendar component*/
let calendar: Calendar = new 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: ChangedEventArgs) {
    let startOfWeek: any = moment(calendar.value).startOf('week');
    let endOfWeek: any = 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: any, endOfWeek: any): void {
    let days: Date[] = [];
    let day: any = 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 EJ2 JavaScript 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.TS

```

import { Calendar, NavigatedEventArgs } from '@syncfusion/ej2-calendars';
//Creates a calendar component.
let calendarObject: Calendar = new Calendar({ navigated: onNavigate });
calendarObject.appendTo('#element');
//Skips a month while clicking previous and next icons in month view.
function onNavigate(args: NavigatedEventArgs) {

```



```

    let date: Date;
    if ((<HTMLElement>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 ((<HTMLElement>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);
    }
}

```

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>

```

Render the calendar with week numbers in EJ2 JavaScript Calendar control

You can enable `weekNumbers` in the Calendar by using the [weekNumber](#) property.

INDEX.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
//creates a calendar with weekNumber enabled
let calendarObject: Calendar = new Calendar({
    //sets the weekNumber
    weekNumber: true
});
calendarObject.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>
```

Change the first day of week in EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
//creates a calendar with Tuesday as the first day of the week.
let calendarObject: Calendar = new Calendar({
    //sets the first day of the week.
    firstDayOfWeek: 2
});
calendarObject.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>
```

Customize the calendar day header in EJ2 JavaScript 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.TS

```
import { Calendar } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let calendarObject: Calendar = new Calendar({
    dayHeaderFormat: "Short"
});
calendarObject.appendTo('#element');
let formatLabel: DropDownList = new DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: (args: any) => {
        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>
</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

Getting started in EJ2 JavaScript Card control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework that can be directly used in latest web browsers.

Component initialization

The Essential JS 2 JavaScript components can be initialized by using any of the following two ways:

- Using local script and style references in an HTML page.
- Using CDN link for script and style reference.

Using local script and style references in an HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript components.

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

Syntax:

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

Styles: **(installed location)/Syncfusion/Essential Studio/{RELEASEVERSION}/Essential JS 2/{PACKAGENAME}/styles/material.css**

Example:

Script: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.17/Essential JS 2/ej2-lists/dist/global/ej2-lists.min.js

Styles: C:/Program Files (x86)/Syncfusion/Essential Studio/15.4.17/Essential JS 2/ej2-layouts/styles/material.css

Step 3: Create a folder **myapp/resources**, and copy and paste the global scripts and styles from the above installed location to **myapp/resources** location.

Step 4: Create an HTML page (index.html) in **myapp** location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, create a **span** element to apply the **Essential JS 2 Card** component in the **index.html** by using following code.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
```

```

<!-- Essential JS 2 material theme -->
<link href="resources/material.css" rel="stylesheet" type="text/css"/>
</head>
<body>
<!-- Add the HTML <span> element -->
<span class='e-card'></span>
</body>
</html>
`

```

Step 6: Now, open the `index.html` in web browser, it will render the **Essential JS 2 Card** component.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the following CDN link formats.

Syntax:

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

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

Example:

Styles: <https://cdn.syncfusion.com/ej2/ej2-layouts/styles/material.css>

Step 3: Create an HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `span` element and initiate the `Essential JS 2 Card` component in the `index.html` by using the following code.

```

`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="https://cdn.syncfusion.com/ej2/ej2-base/styles/material.css" rel="stylesheet"
type="text/css"/>
<link href="https://cdn.syncfusion.com/ej2/ej2-layouts/styles/material.css" rel="stylesheet"
type="text/css"/>
</head>

```

```

<body>
<!-- Add the HTML <span> element -->
<span class='e-card'>This is simple Card</span>
</body>
</html>
`

```

Step 4: Now, open the `index.html` in the web browser, it will render the **Essential JS 2 Card** component.

Need to refer dependency component styles and scripts as like above example. We can also use [CRG](#) to generate combined component styles.

Adding a header and content

You can create Card with a header in a specific structure. For adding header you need to create a `div` element with `e-card-header` class added.

- You can include heading inside the Card header by adding a `div` element with `e-card-header-caption` class, and also content will be added by adding element with `e-card-content`. For detailed information, refer to the [Header and Content](#).

```

<div class = "e-card">           --> Root Element
<div class="e-card-header">      --> Root Header Element
<div class="e-card-header-caption"> --> Root Heading Element
<div class="e-card-header-title"></div> --> Heading Title Element
</div>
<div class="e-card-content"></div> --> Card content Element
</div>
</div>
`

```

- Now, open the html file in the web browser, it will render the **Card** component with header and content.

Output will be as follows:

INDEX.HTML

```

<html><head><script
src="https://cdn.syncfusion.com/ej2/24.1.41/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>
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 a header and content](#)

Header content in EJ2 JavaScript 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 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 EJ2 JavaScript 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>

```

```

</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 EJ2 JavaScript 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 EJ2 JavaScript 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 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 EJ2 JavaScript 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 EJ2 JavaScript 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 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
  placeholder:"Select Position",
  change: changed,

});
// render initialized DropDownList
dropDownListObject.appendTo('#title_position');

```

```
function changed(e) {
    let cardEle = document.querySelector('.e-card');
    let 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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>
<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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
(window as TemplateFunction).getImage = (bird: string) => {
  return `https://ej2.syncfusion.com/products/images/carousel/${bird}.png`;
};
interface TemplateFunction extends Window {
  getImage?: Function;
}
const productItems: Record<string, string | number>[] = [
  { 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' }
  ];
  const carouselObj: Carousel = new Carousel({
    dataSource: productItems,
    itemTemplate: '<figure class="img-container"><figcaption class="img-
caption">${Name}</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>
  <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.TS

```
import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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>
</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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const carouselObj: Carousel = new 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");
const prevButton: Button = new Button({ cssClass: "e-info" });
prevButton.appendTo("#prev");
prevButton.element.onclick = (): void => {
  carouselObj.prev();
};
const nextButton: Button = new Button({ cssClass: "e-info" });
nextButton.appendTo("#next");
nextButton.element.onclick = (): void => {
  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>
  </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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new Carousel({
    partialVisible: true,
    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>
<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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new Carousel({
    partialVisible: true,
    loop: false,
    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>
<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 EJ2 JavaScript 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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"><img
src="https://ej2.syncfusion.com/products/images/carousel/kaohsiung.png"

```



```
alt="yellow-warbler" style="height:100%;width:100%;" /><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>
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.TS

```
import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const carouselObj: Carousel = new 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");
const prevBtn: Button = new Button({
  cssClass: "e-flat e-round",
  iconCss: "e-icons e-chevron-left-double",
});
prevBtn.appendTo("#previous");
const nextBtn: Button = new Button({
  cssClass: "e-flat e-round",
  iconCss: "e-icons e-chevron-right-double",
});
nextBtn.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.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

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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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",
});
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>
    </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>

```

```

    </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>

```

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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
(window as TemplateFunction).getContent = (index: number) => {
    const slides: string[] = ["Slide 1", "Slide 2", "Slide 3", "Slide 4",
    "Slide 5"];
    return slides[index];
};
interface TemplateFunction extends Window {
    getContent?: Function;
}
const carouselObj: Carousel = new 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.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 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.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>

```

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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```
import { Carousel } from "@syncfusion/ej2-navigations";
import { Button } from "@syncfusion/ej2-buttons";
const carouselObj: Carousel = new 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");
const button: Button = new Button({ cssClass: "e-info", content: "Pause" });
button.appendTo("#play");
button.element.onclick = (): void => {
  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 EJ2 JavaScript 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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>
</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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new Carousel({
    items: [
        { template: '<figure class="img-container"><img
src="https://ej2.syncfusion.com/products/images/carousel/cardinal.png"

```



```

alt="cardinal" style="height:100%;width:100%;" /><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.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>

```

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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```

import { Carousel, SlideChangedEventArgs, SlideChangingEventArgs } from
"@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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: SlideChangingEventArgs) {
    console.log(args.currentSlide); // You can customize the slide before
changing.
  },
  slideChanged: function (args: SlideChangedEventArgs) {
    console.log(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>
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.TS

```

import { Carousel } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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.TS

```
import { Carousel, CarouselSwipeMode } from "@syncfusion/ej2-navigations";
const carouselObj: Carousel = new 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: CarouselSwipeMode.Touch & CarouselSwipeMode.Mouse
});
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 EJ2 JavaScript Carousel control

The [JavaScript Carousel](#) control has been designed, keeping in mind the [WAI-ARIA](#) specifications, and applying the WAI-ARIA roles, states and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like `aria-roledescription`, `aria-label`, `aria-current`, `aria-live`, `aria-role` and `aria-hidden`. It provides information about elements in a document for assistive technology. The control implements keyboard navigation support by following the [WAI-ARIA practices](#) and has been tested in major screen readers.

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, **aria-current** is set to **true**.

|

| aria-hidden | For all carousel elements except the currently visible item, **aria-hidden** is set to **true**. |

| aria-live | For carousel items element, when **autoPlay** is **true**, **aria-live** is set to **off**; when **autoPlay** is **false**, **aria-live** is set to **polite**. |

| aria-role | For carousel slide item, **aria-role** 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 EJ2 JavaScript 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

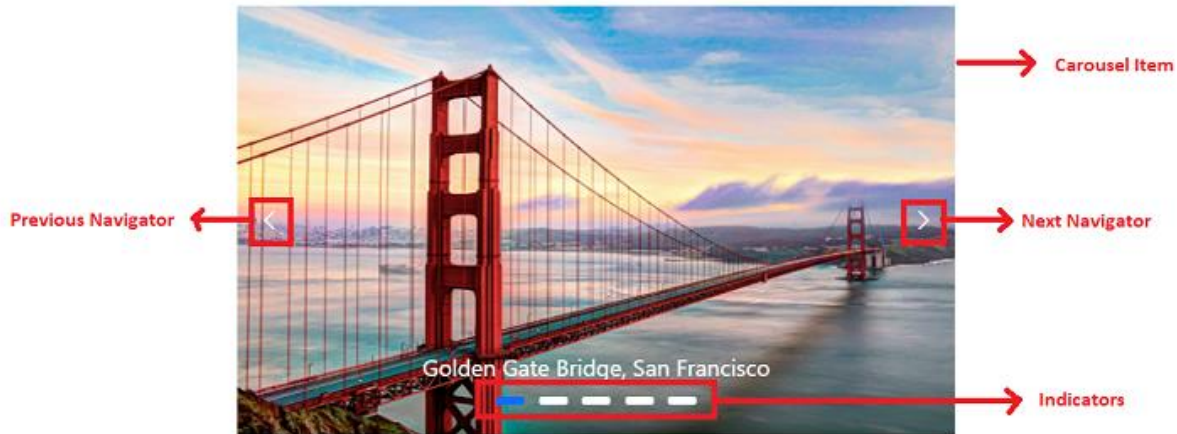
|.e-carousel .e-carousel-navigators|To customize the navigators

|.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
```

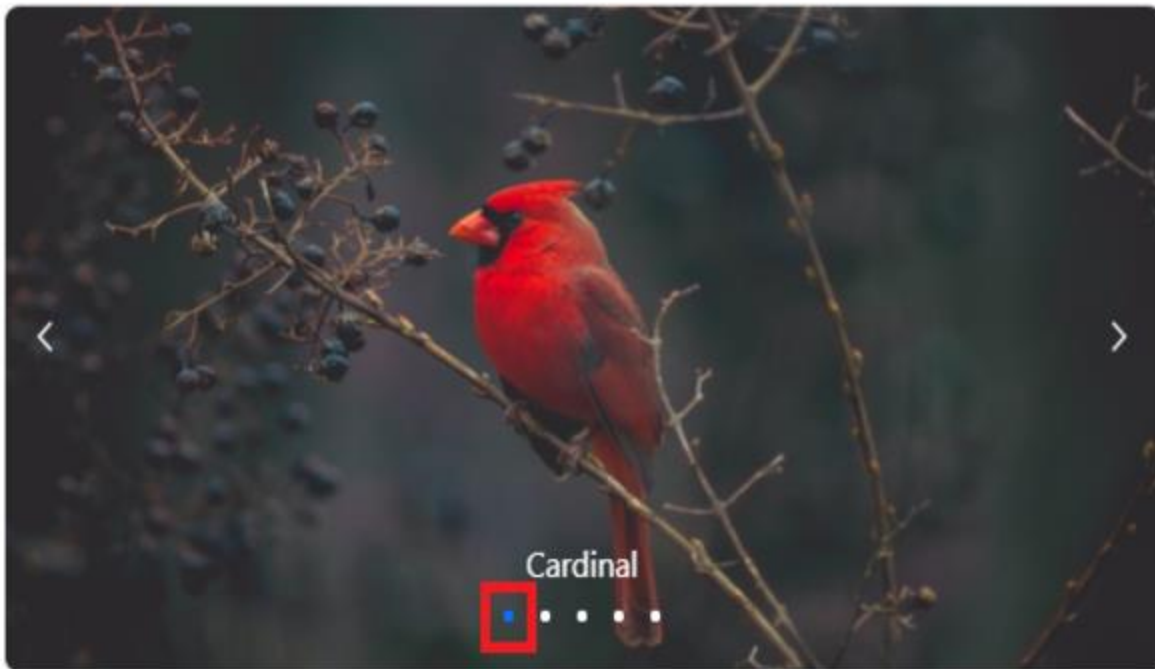
```
.e-carousel .e-carousel-indicators .e-indicator-bars .e-indicator-bar .e-indicator {
```

```
width: 20px;
```

```
border-radius: 100%;
```

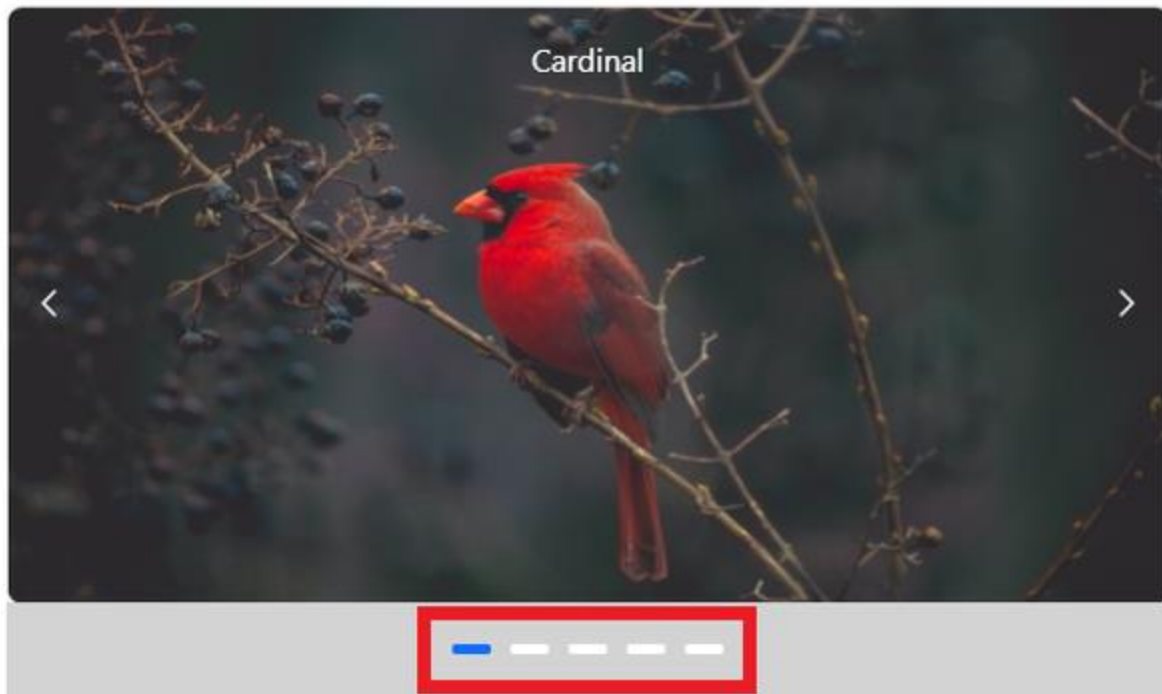
```
}
```

```
,
```



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;
}
```



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.

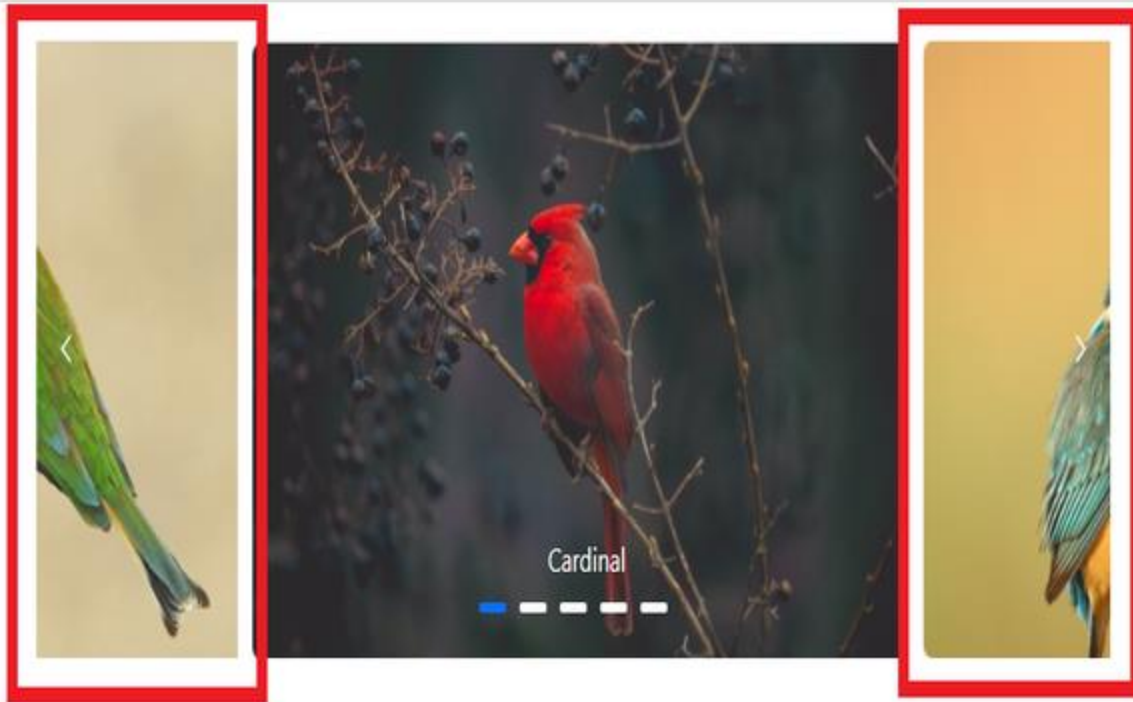
```
`css
```

```
.e-carousel.e-partial .e-carousel-slide-container{
```

```
padding: 0 150px;
```

```
}
```

```
`
```



Chart

<!-- markdownlint-disable MD036 -->

Working with data in EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
  { 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: Chart = new 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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { DataManager, Query } from '@syncfusion/ej2-data';
Chart.Inject(ColumnSeries, Category);
let dataManager: DataManager = new DataManager({
  url: 'https://services.syncfusion.com/js/production/api/orders'
});
let query: Query = new Query().take(5).where('Estimate', 'lessThan', 3,
false);
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { DataManager, Query, ODataAdaptor } from '@syncfusion/ej2-data';
Chart.Inject(ColumnSeries, Category);
let dataManager: DataManager = new DataManager({

```

```
url: 'https://services.syncfusion.com/js/production/api/orders',
adaptor: new ODataAdaptor()
});
let query: Query = new Query();
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  //Initializing Chart Sample
  series: [
    {
      type: 'Column',
      dataSource: dataManager,
      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.TS

```

import { Chart, ScrollBar, Zoom, IScrollEventArgs, LineSeries, Tooltip,
DateTime } from '@syncfusion/ej2-charts';
import { Internationalization, DateFormatOptions } from '@syncfusion/ej2-
base';
Chart.Inject(DateTime, ScrollBar, Zoom, LineSeries, Tooltip);
let intl: Internationalization = new Internationalization();
let chart: Chart = new 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: (args: IScrollEventArgs) => {
    if (lazymode.value === 'Range') {
      chart.series[0].dataSource =
GetDateTimeData(args.currentRange.minimum as Date, args.currentRange.maximum
as Date);
    }
    chart.dataBind();
  },
}, '#element');
function GetDateTimeData(start: Date, end: Date): { x: Date, y: number }[] {
  let series1: { x: Date, y: number }[] = [];
  let date: number;
  let value: number = 30;
  let option: DateFormatOptions = {
    skeleton: 'full',
    type: 'dateTime'
  };
  let dateParser: Function = intl.getDateParser(option);
  let dateFormatter: Function = intl.getDateFormat(option);
  for (let i: number = 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);
    }
    let point1: { x: Date, y: number } = { x: new Date(date), y:
Math.round(value) };
    new Date(start.setDate(start.getDate() + 1));
    series1.push(point1);
}
return series1;
} function getRandomInt(min: number, max: number): number {
    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.TS

```
import { Chart, ColumnSeries, LineSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, LineSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, LineSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, LineSeries, Category);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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>
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 EJ2 JavaScript Chart control

Size for Container

Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

```
`javascript
<div id='container'>
  <div id='element' style="width:650px; height:350px;"></div>
</div>
`
```

INDEX.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category);
let chartData: any[] = [
  { 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: Chart = new Chart({
        primaryXAxis: {
            valueType: 'Category'
        },
        series:[{
            dataSource: chartData,
            xName: 'month',
            yName: 'sales',
            type: 'Line'
        }],
        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>
  <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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category);
let chartData: any[] = [
    { 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: Chart = new 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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category);
let chartData: any[] = [
    { 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: Chart = new 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 EJ2 JavaScript Chart control

<!-- markdownlint-disable MD036 -->

Category axis are used to represent, the string values instead of numbers.

INDEX.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        //Category in primary X Axis
        valueType: 'Category',
    },
    series:[{
        dataSource: chartData,
        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>
</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 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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label placement as on ticks
        labelPlacement: 'OnTicks',
    },
    series: [{
        dataSource: categoryData,
        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>
</script>
<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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        interval: 2, minimum: 1, maximum: 5
    },
    series:[{
        dataSource: categoryData,
        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>
</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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        isIndexed: true
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: [{ x: 'Myanmar', y: 7.3 }, { x: 'India', y: 7.9 }, { x: 'Bangladesh', y: 6.8 }, { x: 'Cambodia', y: 7.0 }, { x: 'China', y: 6.9 }],
            xName: 'x', yName: 'y',
        },
        {
            type: 'Column',
            dataSource: [{ x: 'Poland', y: 2.7 }, { x: 'Australia', y: 2.5 }, { x: 'Singapore', y: 2.0 }, { x: 'Canada', y: 1.4 }, { x: 'Germany', y: 1.8 }],
            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>
```

<!-- markdownlint-disable MD036 -->

Numeric axis in EJ2 JavaScript Chart control

You can use [numeric axis](#) to represent numeric values of data in chart. By default, the `valueType` of an axis is `Double`.

INDEX.TS

```
import { Chart, LineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries);
let chartData: any[] = [{ 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 }];
let chart: Chart = new Chart({
  primaryXAxis: {
    // Numerical scale in primary X Axis
    valueType: 'Double',
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    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>

```

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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Double',
        // Numeric axis range
        minimum: 1,
        maximum: 20,
        interval: 5
    },
    series: [{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Double',
    },
    primaryYAxis: {
        //RangePadding as none in Y Axis
        rangePadding: 'None'
    },
    series:[{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Double',
  },
  primaryYAxis: {
    rangePadding: 'Round'
  },
  series:[{
    dataSource: numericData,
    xName: 'x', yName: 'y',
    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>

```

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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Double',
    },
    primaryYAxis: {
        rangePadding: 'Additional'
    },
    series:[{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

Numeric - Normal

When the [rangePadding](#) is set to **Normal**, padding is applied to the axis based on default range calculation.

INDEX.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Double',
    },
    primaryYAxis: {
        rangePadding: 'Normal'
    },
    series:[{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Double',
        // Set the rangePadding as auto in X Axis
        rangePadding: 'Auto'
    },
    primaryYAxis: {
        valueType: 'Double',
        // Set the rangePadding as auto in Y Axis
        rangePadding: 'Auto'
    },
    series:[{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

Label Format

Numeric Label Format

Numeric labels can be formatted by using the [labelFormat](#) property. Numeric labels supports all globalize format.

INDEX.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
    primaryYAxis: {
        //Label format as currency
        labelFormat: 'c'
    },
    series:[{
        dataSource: numericData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```
import { Chart, ColumnSeries } from '@syncfusion/ej2-charts';
import { groupingData } from './datasource.ts';
Chart.Inject(ColumnSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: groupingData,
    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.TS

```
import { Chart, ColumnSeries } from '@syncfusion/ej2-charts';
import { numericData } from './datasource.ts';
Chart.Inject(ColumnSeries);
let chart: Chart = new Chart({
  primaryYAxis: {
    // Custom label format
    labelFormat: '${value}K'
  }
});
```

```

    },
    series:[{
      dataSource: numericData,
      xName: 'x', yName: 'y',
      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>

```

<!-- markdownlint-disable MD036 -->

Date time axis in EJ2 JavaScript 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.TS

```

import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime);
let chartData: any[] = [{ 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 }];
let chart: Chart = new Chart({
  primaryXAxis: {

```

```
// Date time scale in primary X Axis
valueType: 'DateTime',
},
series:[{
  dataSource: chartData,
  xName: 'x', yName: 'y',
  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>
```

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.TS

```
import { Chart, DateTimeCategory, ColumnSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, DateTimeCategory);
let chartData: any[] = [{ 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 }
    ];
    let chart: Chart = new Chart({
        //Initializing Primary X Axis
        primaryXAxis: {
            valueType: 'DateTimeCategory',
            skeleton: 'Ed',
        },
        series: [
            {
                type: 'Column',
                dataSource: chartData,
                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 `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.TS

```
import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
  primaryXAxis: {
    // Date time range in primary X Axis
    valueType: 'DateTime',
    minimum: new Date(2000, 6, 1),
    maximum: new Date(2010, 6, 1), interval: 1
  },
  series:[{
    dataSource: data,
    xName: 'x', yName: 'y',
    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>
```

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.TS

```
import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    //interval type as years in primary x axis
    intervalType: 'Years'
  },
  series:[{
    dataSource: data,
    xName: 'x', yName: 'y',
    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>
<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.TS

```

import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        //Range padding as none
        rangePadding: 'None'
    },
    series:[{
        dataSource: data,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        //Range padding as round
        rangePadding: 'Round'
    },
    series:[{
        dataSource: data,
        xName: 'x', yName: 'y',
        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>
</body>
<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.TS

```

import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        //Range padding as additional
        rangePadding: 'Additional'
    },
    series:[{
        dataSource: data,
        xName: 'x', yName: 'y',
        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>
```

Label Format

You can format and parse the date to all globalize format using [labelFormat](#) property in an axis.

INDEX.TS

```
import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(LineSeries, DateTime);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    //Label format as yMd
    labelFormat: 'yMd'
  },
  series:[{
    dataSource: data,
    xName: 'x', yName: 'y',
    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>

```

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.TS

```

import { Chart, DateTime, LineSeries } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(DateTime, LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    primaryYAxis: {
        // Custom label format
        labelFormat: '${value}K'
    },
    series: [{
        dataSource: data,
        xName: 'x', yName: 'y',
        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>

```

Logarithmic axis in EJ2 JavaScript 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: 10^6).

INDEX.TS

```

import { Chart, DateTime, LineSeries, Logarithmic } from '@syncfusion/ej2-
charts';
Chart.Inject(LineSeries, DateTime, Logarithmic);
let chartData: any[] = [{ 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: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',

```

```

    },
    primaryYAxis: {
        // Logarithmic scale in primary X Axis
        valueType: 'Logarithmic',
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, DateTime, LineSeries, Logarithmic } from '@syncfusion/ej2-
charts';
import { logData } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Logarithmic);

```

```
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  primaryYAxis: {
    //Logarithmic scale range in primary X Axis
    valueType: 'Logarithmic',
    minimum: 100,
    maximum: 10000
  },
  series:[{
    dataSource: logData,
    xName: 'x', yName: 'y',
    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>
```

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.TS


```
import { Chart, DateTime, LineSeries, Logarithmic } from '@syncfusion/ej2-charts';
import { logData } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Logarithmic);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  primaryYAxis: {
    valueType: 'Logarithmic',
    // logBase for logarithmic scale
    logBase: 2
  },
  series:[{
    dataSource: logData,
    xName: 'x', yName: 'y',
    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>
```

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.TS

```
import { Chart, DateTime, LineSeries, Logarithmic } from '@syncfusion/ej2-charts';
import { logData } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Logarithmic);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    primaryYAxis: {
        //Logarithmic interval in primary X Axis
        valueType: 'Logarithmic',
        interval: 2
    },
    series:[{
        dataSource: logData,
        xName: 'x', yName: 'y',
        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>
```

Axis labels in EJ2 JavaScript 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.TS

```
import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chartData: any[] = [{ 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 }];
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    //label intersect as hide
    labelIntersectAction: 'Hide'
  },
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    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>

```

When setting `labelIntersectAction` as `Rotate45`

INDEX.TS

```

import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-charts';
import { smartAxisData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        // label intersect as 45
        labelIntersectAction: 'Rotate45'
    },
    series:[{
        dataSource: smartAxisData,
        xName: 'x', yName: 'y',
        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>
```

When setting `labelIntersectAction` as `Rotate90`

INDEX.TS

```
import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-charts';
import { smartAxisData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label intersect as 90
    labelIntersectAction: 'Rotate90'
  },
  series:[{
    dataSource: smartAxisData,
    xName: 'x', yName: 'y',
    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>
```

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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    // label placement as on ticks
    labelPosition: 'Inside',
  },
  series:[{
    dataSource: categoryData,
    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>
  <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.TS

```
import { Chart, ColumnSeries, Category, MultiLevelLabel } from
 '@syncfusion/ej2-charts';
import { categoryData, MultiLevelLabel } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, MultiLevelLabel);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    multiLevelLabels:[{ categories: [
      {
        //Start and end values of the multi-level labels
        //accepts number, date and string values
        start: -0.5,
        end: 3.5,
        //Multi-level label's text.
        text: 'Half Yearly 1',
        //Maximum width of the text for multi level
        //labels
        maximumTextWidth:50
      },
      { start: 3.5, end: 7.5, text: 'Half Yearly
2',maximumTextWidth:50 },
    ]}]
  },
  series:[{
    dataSource: categoryData,
    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>
<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.TS

```

import { Chart, ColumnSeries, Category, MultiLevelLabel } from
'@syncfusion/ej2-charts';
import { categoryData, MultiLevelLabel } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, MultiLevelLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1',
maximumTextWidth:50 },
                { start: 3.5, end: 7.5, text: 'Half Yearly
2',maximumTextWidth:50 }},
            overflow:'Trim'
        ]
    },
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, ColumnSeries, Category, MultiLevelLabel } from
'@syncfusion/ej2-charts';
import { categoryData, MultiLevelLabel } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, MultiLevelLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1' },
                { start: 3.5, end: 7.5, text: 'Half Yearly 2' }],
            alignment : 'Far'
        }]
    },
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, ColumnSeries, Category, MultiLevelLabel } from
'@syncfusion/ej2-charts';
import { categoryData, MultiLevelLabel } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, MultiLevelLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1' },
                { start: 3.5, end: 7.5, text: 'Half Yearly 2' }]],
            textStyle:{size:'18px', color:'Red'}
        }
    ],
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, ColumnSeries, Category, MultiLevelLabel } from
'@syncfusion/ej2-charts';
import { categoryData, MultiLevelLabel } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, MultiLevelLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        multiLevelLabels:[{
            categories: [{start: -0.5, end: 3.5, text: 'Half Yearly 1' },
                { start: 3.5, end: 7.5, text: 'Half Yearly 2' }]],
            border:{type:'Brace', color:'Blue', width: 2},
        }]
    },
    series:[{
        dataSource: categoryData,
        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>
  <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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    //Edgelabelplacement for primary x axis
    edgeLabelPlacement: 'Shift',
  },
  series:[{
    dataSource: categoryData,
    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>
<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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        //EdgeLabelplacement for primary x axis
        edgeLabelPlacement: 'Shift',
    },
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        border: { width: 1, type: 'Rectangle', color: 'red' }
    },
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, ColumnSeries, Category, IAxisLabelRenderEventArgs } from
'@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        border: { width: 1, type: 'Rectangle', color: 'red' }
    },
    series:[{
        dataSource: categoryData,
        xName: 'country', yName: 'gold',
        type: 'Column'
    }],
    axisLabelRender : (args : IAxisLabelRenderEventArgs ) => {
        if(args.text === 'France') {
            args.labelStyle.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>

```

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.TS

```

import {
    Chart, BarSeries, DataLabel, Category,
    Tooltip
} from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(BarSeries, Category, Tooltip, DataLabel);
let chart: Chart = new Chart({
    //Initializing Primary X and YAxis
    primaryXAxis: {
        title: 'Country',
        valueType: 'Category',
        majorGridLines: { width: 0 },
        enableTrim: false,
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        {
            minimum: 0,
            maximum: 800,
            labelFormat: Browser.isDevice ? '{value}' : '{value}M',
            labelStyle: {
                color: 'transparent'
            }
        },
    },
    chartArea: {
        border: {
            width: 0
        }
    },
},

```



```
//Initializing Chart Series
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
},
'#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { DateTime, ILoadedEventArgs, ChartTheme, ScrollBar } from
 '@syncfusion/ej2-charts';
import { Chart, AreaSeries, Legend, Zoom } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom, ScrollBar);
let series1: Object[] = [];
let point1: Object;
let value: number = 80;
let i: number;
for (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);
}
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        title: 'Years',
        edgeLabelPlacement: 'Shift',
        majorGridLines: { width: 0 },
        maximumLabels: 1,
    },
    //Initializing Primary Y Axis
    primaryYAxis: {
        title: 'Profit ($)',
        rangePadding: 'None',
        lineStyle: { width: 0 },
        majorTickLines: { width: 0 }
    },
    chartArea: { border: { width: 0 } },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',

```

```

        xName: 'x',
        yName: 'y',
        fill: 'url(#gradient-chart)',
        border: { width: 0.5, color: '#00bdae' },
        animation: { enable: false }
    },
],
zoomSettings:
{
    enableMouseWheelZooming: true,
    enablePinchZooming: true,
    enableSelectionZooming: true,
    mode: 'X',
    enableScrollbar: 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>
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 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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { stripData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    crossesAt : 15
  },
  primaryYAxis: {
    crossesAt : 5
  },
  series:[{
    dataSource: stripData,
    xName: 'x', yName: 'y',
    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>
```

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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries',
    //Axis title text style
    titleStyle: {
      size: '16px', color: 'grey',
      fontFamily : 'Segoe UI', fontWeight : 'bold'
    }
  },
  series:[{
    dataSource: categoryData,
    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>
  <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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    title: 'Countries', titleRotation: 90,
    //Axis title text style
    titleStyle: {
      size: '16px', color: 'grey',
      fontFamily : 'Segoe UI', fontWeight : 'bold'
    }
  },
  series:[{
    dataSource: categoryData,
    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>
  <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.TS

```
import { Chart, Category, ColumnSeries } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        //Tick lines customization
        majorTickLines: {
            color: 'blue',
            width: 5
        },
        minorTickLines: {
            color: 'red',
            width: 0
        }
    },
    primaryYAxis: {
        //Grid lines customization
        majorTickLines: {
            color: 'blue',
            width: 5
        },
        minorTickLines: {
            color: 'red',
            width: 0
        }
    },
    series: [{
        dataSource: categoryData,
        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>
</script>
<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.TS

```

import { Chart, Category, ColumnSeries } from '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
        //Grid lines customization
        majorGridLines : {
            color : 'blue',
            width : 1
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    },
    primaryYAxis: {
        //Grid lines customization
        majorGridLines : {
            color : 'blue',
            width : 1
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    },
    series:[{
        dataSource: categoryData,
        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>
<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.TS

```

import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-
charts';
import { categoryData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    // Initializing multiple axis
    axes:[
        {
            rowIndex: 0,
            name: 'yAxis',
        }
    ],
    series:[{
        dataSource: categoryData,
        xName: 'country', yName: 'gold',
        type: 'Column'
    },{
        dataSource: categoryData,
        xName: 'country', yName: 'silver',
        yAxisName: 'yAxis',
    },

```

```

        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>

```

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.TS

```

import { Chart, LineSeries, Category, Legend, DataLabel } from
 '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(LineSeries, Category, Legend, DataLabel);
/**
 * inversed axis sample
 */
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
    isInversed: true

```

```

    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        isInversed: true
    },
    series: [
        {
            type: 'Line',
            dataSource: categoryData,
            xName: 'country',
            yName: 'gold',
        },
    ],
    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.TS

```

import { Chart, LineSeries, Category, Legend, DataLabel } from
 '@syncfusion/ej2-charts';
import { categoryData } from './datasource.ts';
Chart.Inject(LineSeries, Category, Legend, DataLabel);

```

```
/**
 * inversed axis sample
 */
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    //Axis position as opposite
    opposedPosition: true
  },
  series: [
    {
      type: 'Line',
      dataSource: categoryData,
      xName: 'country',
      yName: 'gold',
    },
  ],
  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 EJ2 JavaScript 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.TS

```
import { Chart, Category, ColumnSeries, LineSeries, DataLabel, StripLine }
from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, LineSeries, DataLabel, StripLine);
let chartData: any[] = [{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}];
let chart: Chart = new Chart({
    primaryYAxis: {
        stripLines:[
            { start: 15, end: 22 },
            { start: 8, end: 15 },
            { start: 0, end: 8 }
        ],
    },
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y', 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>
</body>
```

```

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.TS

```

import { Chart, Category, ColumnSeries, LineSeries, DataLabel, StripLine }
from '@syncfusion/ej2-charts';
import { stripData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, LineSeries, DataLabel, StripLine);
let chart: Chart = new Chart({
    primaryXAxis: {
        stripLines:[
            {start: 0, end: 5 },
            {start: 5, end: 10 },
        ]
    },
    series:[{
        dataSource: stripData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, ColumnSeries, LineSeries, DataLabel, StripLine } from
'@syncfusion/ej2-charts';
import { stripData } from './datasource.ts';
Chart.Inject(ColumnSeries, LineSeries, DataLabel, StripLine);
let chart: Chart = new Chart({
    primaryXAxis: {
        stripLines:[
            { startFromOrigin: true, size: 4, zIndex: 'Behind', opacity:
0.5}
        ]
    },
    series:[{
        dataSource: stripData,
        xName: 'x', yName: 'y',
        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>

```

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.TS

```

import { Chart, ColumnSeries, LineSeries, StripLine } from '@syncfusion/ej2-charts';
import { stripData } from './datasource.ts';
Chart.Inject(ColumnSeries, LineSeries, StripLine);
let chart: Chart = new Chart({
    primaryXAxis: {
        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' }
        ]
    },
    series: [{
        dataSource: stripData,
        xName: 'x', yName: 'y',
        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>

```

See Also

- [Mark the threshold in chart](#)

Multiple panes in EJ2 JavaScript 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.TS

```

import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```
import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, Category, ColumnSeries, LineSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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>
</body>
<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 EJ2 JavaScript control

Line

To render a line series, use series [type](#) as `Line` and inject `LineSeries` module using `Chart.Inject(LineSeries)` method.

INDEX.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new Chart({
  series:[{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    //Series type as line
    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>
</body>
<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.TS

```
import { Chart, MultiColoredLineSeries } from '@syncfusion/ej2-charts';
Chart.Inject(MultiColoredLineSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: [{ x: 2005, y: 28, color: 'red'}, { x: 2006, y: 25,
color:'green'}, { x: 2007, y: 26, color: '#ff0097' }, { x: 2008, y: 27,
color: 'crimson' }, { x: 2009, y: 32, color: 'blue' }, { x: 2010, y: 35
,color: 'darkorange'}],
    xName: 'x', yName: 'y', pointColorMapping: 'color',
    type: 'MultiColoredLine'
  }],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData,
    //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: '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

- [Data label](#)
- [Tooltip](#)

Step line Chart in EJ2 JavaScript control

Step line

To render a step line series, use series [type](#) as `StepLine` and inject `StepLineSeries` module using `Chart.Inject(StepLineSeries)` method.

INDEX.TS

```
import { Chart, StepLineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(StepLineSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData,
    xName: 'x', yName: 'y',
    // Series type as StepLine
    type: 'StepLine'
  }],
}, '#element');
```

INDEX.HTML

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

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

```

<div id="container">
  <div id="element"></div>
</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.TS

```

import { Chart, StepLineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(StepLineSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData,
    //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'
  }],
}, '#element');

```

INDEX.HTML

```

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript control

Stacked Line

To render a stacked line series, use series `type` as `StackingLine` and inject `StackingLineSeries` module using `Chart.Inject(StackingLineSeries)` method.

INDEX.TS

```

import { Chart, Category, Legend, Tooltip, StackingLineSeries } from
'@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(StackingLineSeries, Category, Legend, Tooltip);
/**
 * Sample for StackedLine Series
 */
let chartData: Object[] = [
    { 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 }
];

let chart: Chart = new Chart({
    //Initializing Primary X Axis

```

```

    primaryXAxis: {
        interval: 1, valueType: 'Category'
    },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        title: 'Expense',
        interval: 100,
        labelFormat: '${value}',
    },
    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
    },
}, '#element');

```

INDEX.HTML

```

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, StackingLineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(StackingLineSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        //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}
        type: 'StackingLine'
    }],
}, '#element');

```

INDEX.HTML

```

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

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript 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.TS

```

import { Chart, Category, Legend, Tooltip, StackingLineSeries } from
'@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(StackingLineSeries, Category, Legend, Tooltip);
/**
 * Sample for StackedLine Series
 */
let chartData: Object[] = [
    { 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 }
  ];
  let chart: Chart = new Chart({
    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>'
    }
  }, '#element');

```

INDEX.HTML

```

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

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, StackingLineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(StackingLineSeries);
let chartData: Object[] = [
    { 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 }];
let chart: Chart = new Chart({
    series: [ {
        dataSource: chartData,
        fill: 'green', width: 2,
        dashArray: '2', xName: 'x', marker: { visible: true },
        yName: 'y', type: 'StackingLine100',
    },
    {
        dataSource: chartData,
        fill: 'pink', width: 2,
        dashArray: '2', xName: 'x', marker: { visible: true },
        yName: 'y1', type: 'StackingLine100',
        type: 'StackingLine100',
    },
    {
        dataSource: chartData,

```

```

        fill: 'yellow', width: 2,
        dashArray: '2', xName: 'x', marker: { visible: true },
        yName: 'y2', type: 'StackingLine100',
    },
    {
        dataSource: chartData,
        fill: 'red', width: 2,
        dashArray: '2', xName: 'x', marker: { visible: true },
        yName: 'y3', type: 'StackingLine100',
    }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript control

Spline

To render a spline series, use series [type](#) as `Spline` and inject `SplineSeries` module using `Chart.Inject(SplineSeries)` method.

INDEX.TS

```
import { Chart, SplineSeries, Category } from '@syncfusion/ej2-charts';
import { polarCategory } from './datasource.ts';
Chart.Inject(SplineSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: polarCategory,
        xName: 'x', yName: 'y',
        // Series type as spline series
        type: 'Spline'
    }],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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** 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.TS

```
import { Chart, SplineSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(SplineSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        //fill for chart series
        fill: 'blue',
        //line width as 4 for chart series
        width:4,
        //dash array value as 5,5
        dashArray: '2',
        xName: 'x', yName: 'y',
        type: 'Spline'
    }],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript control

Area

To render a [area series](#), use series [type](#) as `Area` and inject `AreaSeries` module using `Chart.Inject(AreaSeries)` method.

INDEX.TS

```
import { Chart, AreaSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(AreaSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        xName: 'x', yName: 'y',
        // Series type as area series
        type: '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>
<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.TS

```
import { Chart, MultiColoredAreaSeries } from '@syncfusion/ej2-charts';
Chart.Inject(MultiColoredAreaSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: [{ 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: 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.TS

```
import { Chart, AreaSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(AreaSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        xName: 'x', yName: 'y',
        fill:'#69D2E7',
        border:{width:2, color:'Red'},
        dashArray: '5,5',
        name: 'Product A',
        type: '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>
  <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.TS

```
import { Chart, AreaSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(AreaSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        xName: 'x', yName: 'y',opacity:0.4,
        // Series type as area series
        type: 'Area', border:{ width: 1.5 }
    }],
}, '#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>
```

See Also

- [Data label](#)

- [Tooltip](#)

Range area Chart in EJ2 JavaScript 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.TS

```
import { Chart, RangeAreaSeries, DateTime } from '@syncfusion/ej2-charts';
Chart.Inject(RangeAreaSeries, DateTime);
let series: Object[] = [];
let value: number = 70;
let point: Object;
for (let i: number = 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 - 25 };
    series.push(point);
}
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'RangeArea',
            dataSource: series,
            xName: 'x', high: 'high', low: 'low',
        },
    ],
}, '#element');
```

INDEX.HTML

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

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

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.TS

```

import { Chart, RangeAreaSeries, DateTime } from '@syncfusion/ej2-charts';
Chart.Inject(RangeAreaSeries, DateTime);
let series: Object[] = [];
let value: number = 70;
let point: Object;
for (let i: number = 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 - 25 };
  series.push(point);
}
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  series: [
    {
      type: 'RangeArea', opacity: 0.7,
      dataSource: series, fill: 'blue',
      xName: 'x', high: 'high', low: 'low',
    },
  ],
}, '#element');

```

INDEX.HTML

```

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

```

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript 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.TS

```

import { Chart, RangeStepAreaSeries, Category } from '@syncfusion/ej2-
charts';
import { splinedata } from './datasource.ts';
Chart.Inject(RangeStepAreaSeries, Category);
let chart: Chart = new 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,
    },
  ],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, RangeStepAreaSeries, Category } from '@syncfusion/ej2-
charts';
import { splinedata } from './datasource.ts';
Chart.Inject(RangeStepAreaSeries, Category);
let chart: Chart = new 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'
    }
],
'#element');

```

INDEX.HTML

```

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

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript 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.TS

```
import { Chart, SplineRangeAreaSeries, Category } from '@syncfusion/ej2-charts';
import { splinedata } from './datasource.ts';
Chart.Inject(SplineRangeAreaSeries, Category);
let chart: Chart = new 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,
        }
    ],
}, '#element');
```

INDEX.HTML

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

```

<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, SplineRangeAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
Chart.Inject(SplineRangeAreaSeries, DateTime);
let series: Object[] = [];
let value: number = 70;
let point: Object;
for (let i: number = 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 - 25 };
    series.push(point);
}
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'SplineRangeArea', opacity: 0.7,

```



```

        dataSource: series, fill: 'blue',
        xName: 'x', high: 'high', low: 'low',
    }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript control

Stacked Area

To render a stacked area series, use series [type](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.TS

```

import { Chart, StackingAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingAreaSeries, DateTime);
let chart: Chart = new Chart({
  series: [
    {

```

```

        dataSource: stackedData, xName: 'x', yName: 'y',
        //Series type as stacked area series
        type: 'StackingArea',
    }, {
        dataSource: stackedData, xName: 'x', yName: 'y1',
        type: 'StackingArea',
    }, {
        dataSource: stackedData, xName: 'x', yName: 'y2',
        type: 'StackingArea',
    }
],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, StackingAreaSeries, DateTime } from '@syncfusion/ej2-
charts';

```

```

Chart.Inject(StackingAreaSeries, DateTime);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  series: [
    {
      type: 'StackingArea', opacity: 0.7,
      dataSource: [
        { x: new Date(2000, 0, 1), y: 0.03 },
        { x: new Date(2001, 0, 1), y: 0.05 }, { x: new
Date(2002, 0, 1), y: 0.06 },
        { x: new Date(2003, 0, 1), y: 0.09 }, { x: new
Date(2004, 0, 1), y: 0.14 },
        { x: new Date(2005, 0, 1), y: 0.20 }, { x: new
Date(2006, 0, 1), y: 0.29 },
        { x: new Date(2007, 0, 1), y: 0.46 }, { x: new
Date(2008, 0, 1), y: 0.64 },
        { x: new Date(2009, 0, 1), y: 0.75 }, { x: new
Date(2010, 0, 1), y: 1.06 },
        { x: new Date(2011, 0, 1), y: 1.25 }, { x: new
Date(2012, 0, 1), y: 1.55 },
        { x: new Date(2013, 0, 1), y: 1.55 }, { x: new
Date(2014, 0, 1), y: 1.65 }
      ], fill: 'blue',
      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](#)

Stacked area Chart in EJ2 JavaScript control

100% Stacked Area

To render a [100% stacked area](#) series, use series [type](#) as `StackingArea100` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.TS

```
import { Chart, StackingAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
Chart.Inject(StackingAreaSeries, DateTime);
let chartData: any[] = [
  { 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: 1.0, y1: 0.09, y2: 0.61, y3: 0.2 },
  { 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: 0.2, 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: 0.51 },
  { 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: 1.7, 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 },
];
let chart: Chart = new Chart(
  {
    primaryXAxis: {
      valueType: 'DateTime',
    },
    series: [
      {
        type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
        fill: 'blue', xName: 'x', yName: 'y',
      },
      {
        type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
        fill: 'black', xName: 'x', yName: 'y1',
      },
      {
        type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
        fill: 'blue', xName: 'x', yName: 'y3'
      },
    ],
  }
);
```

```

        type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
        fill: 'red', 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 area series.

- [fill](#) – Specifies the color of the area series.
- [opacity](#) – Specifies the opacity of [fill](#).
- [dashArray](#) – Specifies the dashes of series.

INDEX.TS

```

import { Chart, StackingAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
Chart.Inject(StackingAreaSeries, DateTime);
let chartData: any[] = [
  { 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: 1.0, y1: 0.09, y2: 0.61, y3: 0.2 },
    { 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: 0.2, 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: 0.51 },
    { 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: 1.7, 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 },
];
let chart: Chart = new Chart(
{
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
            fill: 'blue', xName: 'x', yName: 'y', dashArray: '5.5',
            border: { width: 2, color: 'black' }
        },
        {
            type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
            fill: 'black', xName: 'x', yName: 'y1', dashArray: '5.5',
            border: { width: 2, color: 'black' }
        },
        {
            type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
            fill: 'blue', xName: 'x', yName: 'y3', dashArray: '5.5',
            border: { width: 2, color: 'black' }
        },
        {
            type: 'StackingArea100', opacity: 0.7, dataSource: chartData,
            fill: 'red', xName: 'x', yName: 'y2', dashArray: '5.5',
            border: { width: 2, color: 'black' }
        },
    ],
},
'#element'
);

```

INDEX.HTML

```

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript control

Stacked area

To render a stacked area series, use series [type](#) as `StackingArea` and inject `StackingAreaSeries` module using `Chart.Inject(StackingAreaSeries)` method.

INDEX.TS

```

import { Chart, StackingStepAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingStepAreaSeries, DateTime);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked area series
            type: 'StackingStepArea',
        },
        {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingStepArea',
        }
    ],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, StackingStepAreaSeries, DateTime } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingStepAreaSeries, DateTime);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked area series
            type: 'StackingStepArea', fill: 'red', opacity: 0.5,
            border: {width: 2, color: 'yellow'}, dashArray: '5,5', step:
'Center'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2', opacity:
0.5,
            type: 'StackingStepArea', fill: 'green', border: {width: 2,
color: 'yellow'},
            dashArray: '5,5', step: 'Center'
        }
    ]
});

```



```
    ],
  }, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript control

Step area

To render a step area series, use series [type](#) as **StepArea** and inject **StepAreaSeries** module using **Chart.Inject(StepAreaSeries)** method.

INDEX.TS

```
import { Chart, StepAreaSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(StepAreaSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData,
    xName: 'x', yName: 'y',
    type: 'StepArea'
  }],
```

```
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, DateTime, StepAreaSeries } from '@syncfusion/ej2-charts';
Chart.Inject(StepAreaSeries, DateTime);
let chart: Chart = new Chart(
  {
    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 }, { x: 2008, y: 500 },
          { x: 2009, y: 473 }, { x: 2010, y: 520 }, { x: 2011, y: 520 }
        ]
      }
    ]
  }
);
```

```

    ],
    fill: 'blue', xName: 'x', yName: 'y', border: { width:2, color:
'red'},
    opacity: 0.4, step: 'Right', dashArray: '5,5'
  },
  ],
},
'#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>

```

See also

- [Data label](#)
- [Tooltip](#)

Spline area Chart in EJ2 JavaScript control

Spline area

To render a spline area series, use series [type](#) as `SplineArea` and inject `SplineAreaSeries` module using `Chart.Inject(SplineAreaSeries)` method.

INDEX.TS

```

import { Chart, SplineAreaSeries, Category } from '@syncfusion/ej2-charts';
import { polarCategory } from './datasource.ts';

```

```

Chart.Inject(SplineAreaSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series:[{
    dataSource: polarCategory,
    xName: 'x', yName: 'y',
    // Series type as SplineArea
    type: 'SplineArea'
  }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import {
  Chart, DateTime, SplineAreaSeries,

```

```

} from '@syncfusion/ej2-charts';
Chart.Inject(SplineAreaSeries, DateTime);
let chart: Chart = new Chart(
{
    series: [
        {
            type: 'SplineArea',
            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 }, { x: 2008, y: 500 },
                { x: 2009, y: 473 }, { x: 2010, y: 520 }, { x: 2011, y: 520 }
            ],
            fill: 'blue', 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](#)

Column Chart in EJ2 JavaScript control

Column

To render a [column series](#), use series [type](#) as `Column` and inject `ColumnSeries` module using `Chart.Inject(ColumnSeries)` method.

INDEX.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as column series
    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>
```

Column space and width

The [columnSpacing](#) and [columnWidth](#) properties are used to customize the space between columns.

INDEX.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        // Series type as column series
        type: 'Column'
    },
    {
        dataSource: columnData,
        xName: 'country',
        yName: 'silver',
        columnWidth: 0.75,
        columnSpacing: 0.5,
        // Series type as column series
        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>
```

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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new 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>
</body>
<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.TS

```
import { Chart, ColumnSeries, Category, Tooltip } from '@syncfusion/ej2-
charts';
import { cylindricalData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, Tooltip);
let chart: Chart = new 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.TS

```
import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        // Series type as column series
        type: 'Column', fill: 'red', border:{ width: 2, color: 'black'}
    }],
}, '#element');
```

INDEX.HTML

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

```

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript control

Range Column

To render a range column series, use series [type](#) as `RangeColumn` and inject `RangeColumnSeries` module using `Chart.Inject(RangeColumnSeries)` method.

INDEX.TS

```

import { Chart, RangeColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(RangeColumnSeries, Category);
let data: any[] = [
    { 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 }
];
let data2: any[] = [
    { 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 }
];

```

```
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series: [
    {
      // Series type as range column series
      type: 'RangeColumn',
      dataSource: data, xName: 'x', high: 'high', low: 'low',
    }, {
      type: 'RangeColumn',
      dataSource: data2, xName: 'x', high: 'high', low: 'low',
    }
  ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, RangeColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(RangeColumnSeries, Category);
let data: any[] = [
    { 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 }
];
let data2: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            // Series type as range column series
            type: 'RangeColumn', fill: 'red', border:{ width: 2, color:
'green'},
            dataSource: data, xName: 'x', high: 'high', low: 'low',
        }, {
            type: 'RangeColumn', fill: 'green', border:{ width: 2, color:
'red'},
            dataSource: data2, xName: 'x', high: 'high', low: 'low',
        }
    ],
}, '#element');
```

INDEX.HTML

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

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

```

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript control

Stacked column

To render a stacked column series, use series [type](#) as `StackingColumn` and inject `StackingColumnSeries` module using `Chart.Inject(StackingColumnSeries)` method.

INDEX.TS

```

import { Chart, StackingColumnSeries, Category } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, Category);
let chart: Chart = new Chart({
    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'
        }
    ],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, StackingColumnSeries, Category } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            type: 'StackingColumn'
            //Stacking group for stacked column series
            stackingGroup: 'UKAndGermany'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'StackingColumn', stackingGroup: 'UKAndGermany'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingColumn', stackingGroup: 'FranceAndItaly'
        }
    ],
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```

```

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, StackingColumnSeries, Category } from '@syncfusion/ej2-
charts';
import { cylindricalData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, Category);
let chart: Chart = new 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.TS

```
import { Chart, StackingColumnSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as stacked column
            type: 'StackingColumn', border: { width: 1.5, color: 'blue' }
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'StackingColumn', border: { width: 1.5, color:
'yellow' }
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingColumn', border: { width: 1.5, 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>
```

See also

- [Data label](#)
- [Tooltip](#)

Stacked column Chart in EJ2 JavaScript control

100% Stacked column

To render a [100% stacked column](#) series, use series [type](#) as `StackingColumn100` and inject `StackingColumnSeries` module using `Chart.Inject(StackingColumnSeries)` method.

INDEX.TS

```
import { Chart, StackingColumnSeries, DateTime } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, DateTime);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            //Series type as 100% stacked column series
            type: 'StackingColumn100'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'StackingColumn100'
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'StackingColumn100'
        }
    ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, StackingColumnSeries, DateTime } from '@syncfusion/ej2-charts';
import { cylindricalData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, DateTime);
let chart: Chart = new 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.TS

```

import { Chart, StackingColumnSeries, Category } from '@syncfusion/ej2-
charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingColumnSeries, Category);
let chart: Chart = new Chart({
  series: [
    {
      dataSource: stackedData, xName: 'x', yName: 'y',
      //Series type as stacked column
      type: 'StackingColumn100', border: { width: 1.5, color:
'blue' }
    }, {
      dataSource: stackedData, xName: 'x', yName: 'y1',
      type: 'StackingColumn100', border: { width: 1.5, color:
'yellow' }
    }
  ]
});

```

```

    }, {
      dataSource: stackedData, xName: 'x', yName: 'y2',
      type: 'StackingColumn100', border: { width: 1.5, 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>

```

See also

- [Data label](#)
- [Tooltip](#)

Bar Chart in EJ2 JavaScript control

Bar

To render a [bar series](#), use series [type](#) as `Bar` and inject `BarSeries` module using `Chart.Inject(BarSeries)` method.

INDEX.TS

```

import { Chart, BarSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(BarSeries, Category);
let chart: Chart = new Chart({

```

```

series:[{
    dataSource: numData,
    xName: 'x', yName: 'y',
    // Series type as bar series
    type: 'Bar'
}],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, BarSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(BarSeries, Category);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData,
    xName: 'x', yName: 'y',
    // Series type as bar series
    type: 'Bar'
  },
  {
    dataSource: numData,
    xName: 'x',

```

```

        yName: 'y1',
        columnSpacing: 0.5,
        columnWidth: 0.75,
        // Series type as bar series
        type: 'Bar',
    }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, BarSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(BarSeries, Category);
let chart: Chart = new 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.TS

```
import { Chart, BarSeries, Category } from '@syncfusion/ej2-charts';
import { cylindricalData } from './datasource.ts';
Chart.Inject(BarSeries, Category);
let chart: Chart = new 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.TS

```
import { Chart, BarSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(BarSeries, Category);
export let numData: object[] = [
  { 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 }];
let chart: Chart = new Chart(
  {
    series: [
      {
        dataSource: numData,
        xName: 'x',
        yName: 'y',
        opacity: 0.5,
        border: { width: 4, color: 'red' },
        dashArray: '2',
        fill: 'blue',
        // Series type as bar series
        type: 'Bar',
      },
    ],
  },
  '#element'
);
```

INDEX.HTML

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

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

```

See also

- [Data label](#)
- [Tooltip](#)

Stack bar Chart in EJ2 JavaScript control

Stacked bar

To render a stacked bar series, use series [type](#) as `StackingBar` and inject `StackingBarSeries` module using `Chart.Inject(StackingBarSeries)` method.

INDEX.TS

```

import { Chart, StackingBarSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            //Series type as stacked bar
            type: 'StackingBar',
            dataSource: stackedData, xName: 'x', yName: 'y'
        }, {
            type: 'StackingBar',
            dataSource: stackedData, xName: 'x', yName: 'y1'
        }, {
            type: 'StackingBar',
            dataSource: stackedData, 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>

```

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.TS

```

import { Chart, StackingBarSeries } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries);
let chart: Chart = new Chart({
    series: [
        {
            type: 'StackingBar',
            //Stacking group for stacked bar
            stackingGroup: 'JohnAndAndrew',
            dataSource: stackedData, xName: 'x', yName: 'y'
        }, {
            type: 'StackingBar', name: 'Andrew', stackingGroup:
'JohnAndAndrew',
            dataSource: stackedData, xName: 'x', yName: 'y1'
        }, {
            type: 'StackingBar', name: 'Thomas', stackingGroup:
'ThomasAndMichael',
            dataSource: stackedData, 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>

```

Cylindrical stacked bar chart

To render a cylindrical stacked bar chart, set the [columnFacet](#) property to `Cylinder` in the chart series.

INDEX.TS

```

import { Chart, StackingBarSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            //Series type as stacked bar with cylinder shape
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: stackedData, xName: 'x', yName: 'y'
        }, {
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: stackedData, xName: 'x', yName: 'y1'
        }, {
            type: 'StackingBar', columnFacet: 'Cylinder',
            dataSource: stackedData, 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.TS

```

import { Chart, StackingBarSeries } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries);
let chart: Chart = new Chart({
    series: [
        {
            type: 'StackingBar',
            //Stacking group for stacked bar
            stackingGroup: 'JohnAndAndrew',
            dataSource: stackedData, xName: 'x', yName: 'y', border: {
width: 2, color: 'red' }
        }, {
            type: 'StackingBar', name: 'Andrew', stackingGroup:
'JohnAndAndrew',
            dataSource: stackedData, xName: 'x', yName: 'y1', border: {
width: 2, color: 'grey' }
        }, {
            type: 'StackingBar', name: 'Thomas', stackingGroup:
'ThomasAndMichael',
            dataSource: stackedData, xName: 'x', yName: 'y2', border: {
width: 2, color: 'lime' }
        }
    ],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript control

100% Stacked bar

To render a [100% stacked bar](#) series, use series [type](#) as `StackingBar100` and inject `StackingBarSeries` module using `Chart.Inject(StackingBarSeries)` method.

INDEX.TS

```

import { Chart, StackingBarSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries, Category);
let chart: Chart = new Chart({
  series: [
    {
      //Series type as 100% stacked bar
      type: 'StackingBar100',
      dataSource: stackedData, xName: 'x', yName: 'y'
    }, {
      type: 'StackingBar100',
      dataSource: stackedData, xName: 'x', yName: 'y1'
    }, {

```



```

        type: 'StackingBar100',
        dataSource: stackedData, 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>

```

100% Cylindrical stacked bar chart

To render a 100% cylindrical stacked bar chart, set the [columnFacet](#) property to **Cylinder** in the chart series.

INDEX.TS

```

import { Chart, StackingBarSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries, Category);
let chart: Chart = new Chart({
  series: [
    {
      //Series type as 100% stacked bar with cylindrical shape
      type: 'StackingBar100', columnFacet: 'Cylinder',
      dataSource: stackedData, xName: 'x', yName: 'y'
    },
    {
      type: 'StackingBar100', columnFacet: 'Cylinder',
      dataSource: stackedData, xName: 'x', yName: 'y1'
    }
  ]
});

```

```

    },
    {
        type: 'StackingBar100', columnFacet: 'Cylinder',
        dataSource: stackedData, 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.TS

```

import { Chart, StackingBarSeries, Category } from '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingBarSeries, Category);
let chart: Chart = new Chart({
  series: [
    {

```

```

//Series type as 100% stacked bar
type: 'StackingBar100',border: { width: 1.5, color: 'red'},
dataSource: stackedData, xName: 'x', yName: 'y'
}, {
type: 'StackingBar100',border: { width: 1.5, color: 'red'},
dataSource: stackedData, xName: 'x', yName: 'y1'
}, {
type: 'StackingBar100',border: { width: 1.5, color: 'red'},
dataSource: stackedData, 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>

```

See also

- [Data label](#)
- [Tooltip](#)

Scatter Chart in EJ2 JavaScript control

Scatter Charts

To render a scatter series, use series [type](#) as `Scatter` and inject `ScatterSeries` module using `Chart.Inject(ScatterSeries)` method.

INDEX.TS

```
import { Chart, ScatterSeries, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ScatterSeries, Legend);
let series1: Object[] = [];
let series2: Object[] = [];
let point1: Object;
let value: number = 80;
let value1: number = 70;
let i: number;
for (i = 1; i < 50; 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 < 50; 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);
}
let chart: Chart = new Chart({
    series: [
        {
            //Series type as scatter
            type: 'Scatter',
            dataSource: series1, xName: 'x', yName: 'y',
        }, {
            type: 'Scatter',
            dataSource: series2, 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>

```

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.TS

```

import { Chart, ScatterSeries, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ScatterSeries, Legend);
let series1: Object[] = [];
let series2: Object[] = [];
let point1: Object;
let value: number = 80;
let value1: number = 70;
let i: number;
for (i = 1; i < 50; 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 < 50; 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);
}
let chart: Chart = new Chart({

```

```

series: [
    {
        //Series type as scatter
        type: 'Scatter', fill: 'black',
        marker: { visible: true, shape: 'Diamond', height: 10, width: 10
    },
        dataSource: series1, xName: 'x', yName: 'y',
    }, {
        type: 'Scatter', fill: 'red',
        marker: { visible: true, shape: 'Triangle', height: 10, width:
10 },
        dataSource: series2, 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](#)

Bubble Chart in EJ2 JavaScript control

Bubble

To render a [bubble series](#), use series [type](#) as `Bubble` and inject `BubbleSeries` module using `Chart.Inject(BubbleSeries)` method.

INDEX.TS

```
import { Chart, BubbleSeries } from '@syncfusion/ej2-charts';
import { EmitType } from '@syncfusion/ej2-base';
Chart.Inject(BubbleSeries);
let chart: Chart = new Chart({
    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',
        },
    ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, BubbleSeries } from '@syncfusion/ej2-charts';
import { bubbleData } from './datasource.ts';
import { EmitType } from '@syncfusion/ej2-base';
Chart.Inject(BubbleSeries);
let chart: Chart = new Chart({
    series: [
        {
            type: 'Bubble',
            dataSource: bubbleData,
            xName: 'x', yName: 'y',
            //size property used to map size values from data source
            size: 'size',
        },
    ],
}, '#element');
```

INDEX.HTML

```

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

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

    <div id="container">
        <div id="element"></div>
    </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 **bubble** series.

- [fill](#) – Specifies the color of the series.
- [opacity](#) – Specifies the opacity of [fill](#).

INDEX.TS

```
import { Chart, BubbleSeries } from '@syncfusion/ej2-charts';
import { bubbleData } from './datasource.ts';
import { EmitType } from '@syncfusion/ej2-base';
Chart.Inject(BubbleSeries);
let chart: Chart = new Chart({
    series: [
        {
            type: 'Bubble',
            dataSource: bubbleData,
            xName: 'x', yName: 'y', fill: 'blue', opacity: 0.5,
            //size property used to map size values from data source
            size: 'size',
        },
    ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, PolarSeries } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, PolarSeries);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        //Series type as polar
        type: 'Polar',
        // Series draw type as line
        drawType: '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>

```

Spline

To render a spline draw type, use series [drawType](#) as **Spline** and inject **SplineSeries** module using **Chart.Inject(SplineSeries)** method.

INDEX.TS

```

import { Chart, SplineSeries, Category, PolarSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(SplineSeries, Category, PolarSeries);
import { polarCategory } from './datasource.ts';
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: polarCategory,
        xName: 'x', yName: 'y',
        // Series type as Polar series
        type: 'Polar'
        // Series draw type as spline
        drawType: 'Spline'
    }],
}, '#element');

```

INDEX.HTML

```

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

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { Chart, AreaSeries, PolarSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(AreaSeries, PolarSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        xName: 'x', yName: 'y',
        // Series type as polar series
        type: 'Polar'
        // Series draw type as area
        drawType: '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>
<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.TS

```
import { Chart, StackingAreaSeries, Category, PolarSeries } from
'@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingAreaSeries, Category, PolarSeries);
let chart: Chart = new Chart({
  series: [
    {
      dataSource: stackedData, xName: 'x', yName: 'y',
      // Series type as polar series
      type : 'Polar',
      //Series draw type as stacked area series
      drawType: 'StackingArea',
    }, {
      dataSource: stackedData, xName: 'x', yName: 'y1',
      type : 'Polar',
      drawType: 'StackingArea',
    }, {
      dataSource: stackedData, xName: 'x', yName: 'y2',
      type : 'Polar',
      drawType: 'StackingArea',
    },
  ],
}, '#element');
```

INDEX.HTML

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

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, Category, PolarSeries, ColumnSeries } from '@syncfusion/ej2-
charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries, ColumnSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
        drawType: '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>

```

Stacked Column

To render a stacked column draw type, use series [drawType](#) as **StackingColumn**.

INDEX.TS

```

import { Chart, Category, PolarSeries, StackingColumnSeries } from
'@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(Category, PolarSeries, StackingColumnSeries);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            type: 'Polar',
            //Series draw type as stacked column
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'Polar',
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'Polar',
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y3',
            type: 'Polar',
            drawType: 'StackingColumn',
        }
    ],
}, '#element');

```

INDEX.HTML

```

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

```

```

<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, Category, PolarSeries, RangeColumnSeries } from
'@syncfusion/ej2-charts';
Chart.Inject(Category, PolarSeries, RangeColumnSeries);
let data: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Polar',
            // Series draw type as range column series
            drawType: 'RangeColumn',
            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.TS

```

import { Chart, ScatterSeries, Legend, Category, PolarSeries } from
'@syncfusion/ej2-charts';
import { polarCategory } from './datasource.ts';
Chart.Inject(ScatterSeries, Legend, Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series: [{
    dataSource: polarCategory,
    xName: 'x', yName: 'y',
    // Series type as Polar series
    type: 'Polar',
    // Series draw type as scatter
    drawType: 'Scatter'
  }],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, Category, PolarSeries } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    startAngle: 90
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Polar',
    // Series draw type as column series
    drawType: '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>

```

Radius

You can customize the radius of the polar series using [coefficient](#) property. By default, `coefficient` is 100.

INDEX.TS

```

import { Chart, Category, PolarSeries } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    coefficient: 50
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Polar',
    // Series draw type as column series
    drawType: '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>

```

See Also

- [Data label](#)
- [Tooltip](#)

Radar Chart in EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, RadarSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, RadarSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData, xName: 'x', yName: 'y',
        //Series type as polar
        type: 'Radar',
        // Series draw type as line
        drawType: '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>
```

Customization

Start Angle

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

INDEX.TS

```
import { Chart, Category, RadarSeries, LineSeries } from '@syncfusion/ej2-
charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, RadarSeries, LineSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    startAngle: 120
  },
  series: [{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Radar',
    // Series draw type as column series
  }]
```

```

        drawType: '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>

```

Radius

You can customize the radius of the polar series using [coefficient](#) property. By default, `coefficient` is 100.

INDEX.TS

```

import { Chart, Category, RadarSeries } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, RadarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
    coefficient: 50
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Radar',
    // Series draw type as column series
    drawType: '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>
```

See Also

- [Data label](#)
- [Tooltip](#)

High low Chart in EJ2 JavaScript 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.TS

```
import { Chart, HiloSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(HiloSeries, Category);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart = new Chart({
    primaryXAxis: {
      valueType: 'Category',
    },
    series:[
      {
        dataSource: chartData,
        xName: 'x', high: 'high', low: 'low',
        //Series type as Hilo
        type: 'Hilo'
      }
    ],
  }, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS


```
import { Chart, HiloSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(HiloSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[
        {
            dataSource: chartData, fill: 'blue',
            xName: 'x', high: 'high', low: 'low',
            //Series type as Hilo
            type: 'Hilo'
        }
    ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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 EJ2 JavaScript 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.TS

```
import { Chart, HiloOpenCloseSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(HiloOpenCloseSeries, Category);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[
    {
      dataSource: chartData,
      xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
      // 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.TS

```

import { Chart, HiloSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(HiloSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: chartData, fill: 'blue',
            xName: 'x', high: 'high', low: 'low',
            //Series type as Hilo
            type: 'Hilo', bearFillColor: 'red', bullFillColor: 'green'
        }
    ],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript 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.TS

```

import { Chart, CandleSeries, Category, DateTime } from '@syncfusion/ej2-
charts';
Chart.Inject(CandleSeries, Category, DateTime);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[
        {

```

```

        dataSource: chartData,
        xName: 'x', open: 'open', close: 'close', high: 'high', low:
        'low',
        // Series type as candle series
        type: 'Candle'
    },
    ],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, CandleSeries, Category, DateTime } from '@syncfusion/ej2-charts';
import { hocData } from './datasource.ts';
Chart.Inject(CandleSeries, Category, DateTime);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: hocData,
            xName: 'x', open: 'open', close: 'close', high: 'high', low:
'low',
            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>
<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 EJ2 JavaScript 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.TS

```

import {
    Chart, Category, ILoadedEventArgs,
    IPointRenderEventArgs, BoxAndWhiskerSeries, Tooltip, getElement,
    BoxPlotMode
} from '@syncfusion/ej2-charts';
Chart.Inject(Category, BoxAndWhiskerSeries, Tooltip);
let chart: Chart = new 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.TS

```

import {
  Chart, Category, IPointRenderEventArgs, BoxAndWhiskerSeries, Tooltip,
  getElement, BoxPlotMode
} from '@syncfusion/ej2-charts';
import { boxPlot } from '../datasource.ts';

```



```

Chart.Inject(Category, BoxAndWhiskerSeries, Tooltip);
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    valueType: 'Category',
  },
  //Initializing Chart Series
  series: [
    {
      type: 'BoxAndWhisker',
      boxPlotMode: 'Normal',
      dataSource: boxPlot,
      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.TS

```
import {
    Chart, Category, ILoadedEventArgs,
    IPointRenderEventArgs, BoxAndWhiskerSeries, Tooltip, getElement,
    BoxPlotMode
} from '@syncfusion/ej2-charts';
import { boxPlot } from './datasource.ts';
Chart.Inject(Category, BoxAndWhiskerSeries, Tooltip);
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category',
    },
    //Initializing Chart Series
    series: [
        {
            type: 'BoxAndWhisker',
            dataSource: boxPlot,
            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 EJ2 JavaScript 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.TS

```
import { Chart, WaterfallSeries, Category, DataLabel } from
 '@syncfusion/ej2-charts';
Chart.Inject(WaterfallSeries, Category, DataLabel);
let chartData: any[] = [
  { 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' }
];
let chart: Chart = new 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.TS

```

import { Chart, WaterfallSeries, Category, DataLabel } from
'@syncfusion/ej2-charts';
Chart.Inject(WaterfallSeries, Category, DataLabel);
let chartData: any[] = [
    { 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' }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series: [
        {
            dataSource: chartData,
            xName: 'x', yName: 'y', intermediateSumIndexes: [4], sumIndexes:
[8],
            //Series type as Waterfall
            type: 'Waterfall', summaryFillColor: "black",
            negativeFillColor: 'green'
        }
    ],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
<title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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 EJ2 JavaScript 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.TS

```

import { Chart, HistogramSeries, DataLabel } from '@syncfusion/ej2-charts';
Chart.Inject(HistogramSeries, DataLabel);
let chartData: Object[] = [];
let points: number[] = [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((value: number) => {
    chartData.push({
        y: value
    });
});
let chart: Chart = new Chart({
    primaryXAxis: {
        minimum: 0, maximum: 100
    },
    legendSettings: { visible: false },
    primaryYAxis: {
        minimum: 0, maximum: 50, interval: 10,
    },
    //Initializing Chart Series
    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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, ErrorBar);
let chartData: any[] = [{ 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 }];
let chart: Chart = new Chart({
  series: [{
    dataSource: chartData,
    xName: 'x', yName: 'y',
    errorBar: {
      visible: true,
    },
    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>
<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.TS

```

import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, ErrorBar);
let chart: Chart = new Chart({
    series: [{
        dataSource: numData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            type: 'Custom',
            mode: 'Both'
            verticalPostiveError: 3,
            horizontalPositiveError: 2,
            verticalNegativeError: 3,
            horizontalNegativeError: 2
        },
        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>

```

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.TS

```

import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, ErrorBar);
let chart: Chart = new Chart({
  series: [{
    dataSource: numData,
    xName: 'x', yName: 'y',
    errorBar: {
      visible: true,
      mode: 'Horizontal'
    },
    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>

```

Error Bar Direction

To change the error bar direction to plus, minus or both side using [direction](#) option.

INDEX.TS

```

import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, ErrorBar);
let chart: Chart = new Chart({
    series: [{
        dataSource: numData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            mode: 'Vertical',
            direction: 'Minus'
        },
        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>
```

Customizing Error Bar Cap

To customize the error bar cap length, width and fill color, you can use [errorBarCap](#) option.

INDEX.TS

```
import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, ErrorBar);
let chart: Chart = new Chart({
  series: [{
    dataSource: numData,
    xName: 'x', yName: 'y',
    errorBar: {
      visible: true,
      errorBarCap: {
        length: 10,
        width: 10,
        color: '#0000ff'
      }
    },
    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>
```

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.TS

```
import { Chart, LineSeries, ErrorBar } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, ErrorBar);
let chart: Chart = new Chart({
    series: [{
        dataSource: numData,
        xName: 'x', yName: 'y',
        errorBar: {
            visible: true,
            errorBarColorMapping: 'color',
            verticalError: 'error'
        },
        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

- [Data label](#)
- [Tooltip](#)

Vertical Chart in EJ2 JavaScript 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.TS

```
import { Chart, SplineSeries, Category } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(SplineSeries, Category);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    type: 'Spline',
  }],
  isTransposed: 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](#)

Pareto Chart in EJ2 JavaScript 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.TS

```

import { ChartTheme, Chart, ColumnSeries, Legend, Tooltip, ILoadedEventArgs } from '@syncfusion/ej2-charts';
import { ParetoSeries, Category, LineSeries } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(ColumnSeries, Category, ParetoSeries, LineSeries, Legend, Tooltip);
/**
 * Sample for Pareto chart
 */
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    interval: 1,
    valueType: 'Category',
    majorGridLines: { width: 0 }, minorGridLines: { width: 0 },
    majorTickLines: { width: 0 }, minorTickLines: { width: 0 },
    lineStyle: { width: 0 }, labelIntersectAction: 'Rotate45'
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      title: 'Frequency of Occurrence',
      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 }
            ],
            xName: 'x', yName: 'y', name: 'Defect', width: 2,
            marker: {
                visible: true
            },
        },
    ],
    width: Browser.isDevice ? '100%' : '75%',
    //Initializing Chart title
    title: 'Pareto chart - Defects in Shirts',
    legendSettings: { visible: false },
    //Initializing User Interaction
    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.TS

```

import { ChartTheme, Chart, ColumnSeries, Legend, Tooltip, ILoadedEventArgs
} from '@syncfusion/ej2-charts';
import { ParetoSeries, Category, LineSeries, Highlight } from
 '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(ColumnSeries, Category, ParetoSeries, LineSeries, Legend,
Tooltip, Highlight);
/**
 * Sample for Pareto chart
 */
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    interval: 1,
    valueType: 'Category',
    majorGridLines: { width: 0 }, minorGridLines: { width: 0 },
    majorTickLines: { width: 0 }, minorTickLines: { width: 0 },
    lineStyle: { width: 0 }, labelIntersectAction: 'Rotate45'
  },
  //Initializing Primary Y Axis
  primaryYAxis: {
    {
      title: 'Frequency of Occurrence',
      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: Browser.isDevice ? 4 : 6, topRight:
Browser.isDevice ? 4 : 6 }
    }
    ],
    //Initializing Chart title
    title: 'Defects in Shirts',
    legendSettings: { visible: true, enableHighlight: true },
    //Initializing User Interaction
    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 EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, PolarSeries } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, PolarSeries);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    series:[{
        dataSource: chartData,
        xName: 'x', yName: 'y',
        //Series type as polar
        type: 'Polar',
        // Series draw type as line
        drawType: '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>

```

Spline

To render a spline draw type, use series [drawType](#) as **Spline** and inject **SplineSeries** module using **Chart.Inject(SplineSeries)** method.

INDEX.TS

```

import { Chart, SplineSeries, Category, PolarSeries } from '@syncfusion/ej2-
charts';
Chart.Inject(SplineSeries, Category, PolarSeries);
import { polarCategory } from './datasource.ts';
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series:[{
        dataSource: polarCategory,
        xName: 'x', yName: 'y',
        // Series type as Polar series
        type: 'Polar'
        // Series draw type as spline
        drawType: 'Spline'
    }],
}, '#element');
```

INDEX.HTML

```

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, AreaSeries, PolarSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(AreaSeries, PolarSeries);
let chart: Chart = new Chart({
    series:[{
        dataSource: numData,
        xName: 'x', yName: 'y',
        // Series type as polar series
        type: 'Polar'
        // Series draw type as area
        drawType: '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>
<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.TS

```

import { Chart, StackingAreaSeries, Category, PolarSeries } from
 '@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(StackingAreaSeries, Category, PolarSeries);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            // Series type as polar series
            type : 'Polar',
            //Series draw type as stacked area series
            drawType: 'StackingArea',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type : 'Polar',
            drawType: 'StackingArea',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type : 'Polar',
            drawType: 'StackingArea',
        },
    ],
}, '#element');
```

INDEX.HTML

```

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

```

```

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, Category, PolarSeries, ColumnSeries } from '@syncfusion/ej2-
charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries, ColumnSeries);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: columnData,
        xName: 'country', yName: 'gold',
        // Series type as polar series
        type: 'Polar',
        // Series draw type as column series
        drawType: '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>

```

Stacked Column

To render a stacked column draw type, use series [drawType](#) as **StackingColumn**.

INDEX.TS

```

import { Chart, Category, PolarSeries, StackingColumnSeries } from
'@syncfusion/ej2-charts';
import { stackedData } from './datasource.ts';
Chart.Inject(Category, PolarSeries, StackingColumnSeries);
let chart: Chart = new Chart({
    series: [
        {
            dataSource: stackedData, xName: 'x', yName: 'y',
            type: 'Polar',
            //Series draw type as stacked column
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y1',
            type: 'Polar',
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y2',
            type: 'Polar',
            drawType: 'StackingColumn',
        }, {
            dataSource: stackedData, xName: 'x', yName: 'y3',
            type: 'Polar',
            drawType: 'StackingColumn',
        }
    ],
}, '#element');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>

```

```

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="description" content="Typescript UI Controls">
<meta name="author" content="Syncfusion">
<link href="index.css" rel="stylesheet">

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, Category, PolarSeries, RangeColumnSeries } from
'@syncfusion/ej2-charts';
Chart.Inject(Category, PolarSeries, RangeColumnSeries);
let data: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Polar',
            // Series draw type as range column series
            drawType: 'RangeColumn',
            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.TS

```

import { Chart, ScatterSeries, Legend, Category, PolarSeries } from
 '@syncfusion/ej2-charts';
import { polarCategory } from './datasource.ts';
Chart.Inject(ScatterSeries, Legend, Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  series:[{
    dataSource: polarCategory,
    xName: 'x', yName: 'y',
    // Series type as Polar series
    type: 'Polar'
    // Series draw type as scatter
    drawType: 'Scatter'
  }],

```

```
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, LineSeries, RadarSeries } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, RadarSeries);
let chart: Chart = new Chart({
  series:[{
    dataSource: numData, xName: 'x', yName: 'y',
    //Series type as polar
    type: 'Radar',
    // Series draw type as line
    drawType: '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>

```

*Customization***Start Angle**

You can customize the start angle of the polar series using [startAngle](#) property. By default, `startAngle` is 0 degree.

INDEX.TS

```

import { Chart, Category, PolarSeries } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Polar',
    // Series draw type as column series
    drawType: '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>

```

Coefficient in axis

You can customize the radius of the polar series using [coefficient](#) property. By default, **coefficient** is 100.

INDEX.TS

```

import { Chart, Category, PolarSeries } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(Category, PolarSeries);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: columnData,
    xName: 'country', yName: 'gold',
    // Series type as polar series
    type: 'Polar',
    // Series draw type as column series
    drawType: '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>
```

See Also

- [Data label](#)
- [Tooltip](#)

Chart series in EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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

Chart allows you to render [combination of different series](#) like Column, line, area etc.

Bar series cannot be combined with any other series as the axis orientation is different from other series.

INDEX.TS

```
import { Chart, StackingColumnSeries, LineSeries } from '@syncfusion/ej2-charts';
import { ColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(StackingColumnSeries, LineSeries, Category, ColumnSeries);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, Legend, Tooltip, ILoadedEventArgs,
ChartTheme, Category, ColumnSeries, } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(LineSeries, DateTime, Category, ColumnSeries, Legend, Tooltip);
/**
 * Sample for Line series
 */
export let data: any[] = [
  {group: { x: 'Aaa', y: 10}, y: 20},
  {group: { x: 'Baa', y: 30}, y: 10}
];

let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {

```



```

        valueType: 'Category'
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column',
            dataSource: data,
            enableComplexProperty: true,
            xName: 'group.x', yName: 'group.y',
        },
        {
            type: 'Column',
            dataSource: data,
            enableComplexProperty: true,
            xName: 'group.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>

```

<!-- markdownlint-disable MD036 -->

Technical indicators in EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries,
  AccumulationDistributionIndicator, Tooltip, Crosshair,
  IAxisLabelRenderEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries,
  AccumulationDistributionIndicator, Tooltip, Crosshair);
let chartData: any[] = [
  { 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: Chart = new 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: IAxisLabelRenderEventArgs) => {
    if (args.axis.name === 'secondary') {
        let value: number = Number(args.text) / 1000000000;
        args.text = String(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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, AtrIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, AtrIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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>
<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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, BollingerBands, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, BollingerBands, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, BollingerBands, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, BollingerBands, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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>
  <script>
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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, EmaIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, EmaIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
    {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: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, MomentumIndicator,
Tooltip, Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, MomentumIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, MomentumIndicator,
Tooltip, Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, MomentumIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
    {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: Chart = new 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries, MacdIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, MacdIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, MacdIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, MacdIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
    {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: Chart = new 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(Rsindicator)`. 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries, RsiIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, RsiIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
    {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},
];
let chart: Chart = new 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.TS

```

import { Chart, LineSeries, DateTime, CandleSeries, SmaIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, SmaIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries, StochasticIndicator,
Tooltip, Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, StochasticIndicator,
Tooltip, Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries, StochasticIndicator,
Tooltip, Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries, StochasticIndicator,
Tooltip, Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```

import { Chart, LineSeries, CandleSeries, DateTime, TmaIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, CandleSeries, DateTime, TmaIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
    {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: Chart = new 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 show/hides the overBought and overSold regions. It is applicable for RSI and stochastic indicators.

INDEX.TS

```
import { Chart, LineSeries, CandleSeries, DateTime, TmaIndicator, Tooltip,
Crosshair } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, CandleSeries, DateTime, TmaIndicator, Tooltip,
Crosshair);
let chartData: any[] = [
  {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: Chart = new 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.TS

```
import { Chart, LineSeries, DateTime, CandleSeries,
AccumulationDistributionIndicator, Tooltip, Crosshair,
IAxisLabelRenderEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, CandleSeries,
AccumulationDistributionIndicator, Tooltip, Crosshair);
let chartData: any[] = [
    { 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: Chart = new 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: IAxisLabelRenderEventArgs) => {
    if (args.axis.name === 'secondary') {
        let value: number = Number(args.text) / 1000000000;
        args.text = String(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 EJ2 JavaScript 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries, Tooltip,
LineSeries, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Tooltip,
Trendlines Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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>
<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 a exponential trendline, use trendline [type](#) as **Exponential** and inject **Trendlines** module using **Chart.Inject(Trendlines)**.

INDEX.TS

```

import { Chart, ScatterSeries, Trendlines, SplineSeries, Category,
LineSeries, Tooltip, Crosshair, Legend, DateTime } from '@syncfusion/ej2-
charts';
Chart.Inject(ScatterSeries, LineSeries, SplineSeries, Category, Tooltip,
Crosshair, Trendlines, Legend, DateTime);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
  point = { x: i, y: yValue[j] };
  data.push(point);
  j++;
}
let chart: Chart = new 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries,
LineSeries, Tooltip, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries,
LineSeries, Tooltip, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries,
LineSeries, Tooltip, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let powerData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries,
LineSeries, Tooltip, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);

```

```

let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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.TS

```
import { Chart, Category, Trendlines, ScatterSeries, SplineSeries, Tooltip,
LineSeries, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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>
  <script>
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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries, Tooltip,
LineSeries, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
  point = { x: i, y: yValue[j] };
  data.push(point);
  j++;
}
let chart: Chart = new 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.TS

```
import { Chart, Category, Trendlines, ScatterSeries, SplineSeries,
LineSeries, Tooltip, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Trendlines,
Tooltip, Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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.TS

```

import { Chart, Category, Trendlines, ScatterSeries, SplineSeries, Tooltip,
LineSeries, TrendlineTypes} from '@syncfusion/ej2-charts';
Chart.Inject(Chart, ScatterSeries, SplineSeries, LineSeries, Tooltip,
Trendlines Category);
let data: Object[] = [];
let yValue: number[] = [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];
let point: Object;
let i: number; let j: number = 0;
for (i = 1973; i <= 2013; i++) {
    point = { x: i, y: yValue[j] };
    data.push(point);
    j++;
}
let chart: Chart = new 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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';

```



```

Chart.Inject(LineSeries, Category);
let chart: Chart = new 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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            type: 'Line',
            dataSource: numData, xName: 'x', yName: 'y',
            marker: {
                visible: true,
                shape: 'Diamond',
                height: 10, width: 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>
```

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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
```

```
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, Category);
let chart: Chart = new Chart({
  series: [
    {
      type: 'Line',
      dataSource: numData, xName: 'x', yName: 'y',
      //Marker shape as image
      marker: {
        visible: true,
        width: 10, height: 10,
        shape: 'Image',
        imageUrl: 'sun_annotation.png'
      }
    }
  ],
}, '#element');
```

INDEX.HTML

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
```

```

Chart.Inject(LineSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            type: 'Line',
            dataSource: numData, xName: 'x', yName: 'y',
            marker: {
                visible: true,
                fill: 'Red', height: 10, width: 10,
                border:{width: 2, color: 'blue'},
            }
        }
    ],
}, '#element');

```

INDEX.HTML

```

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

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```

import { Chart, LineSeries, IPointRenderEventArgs } from '@syncfusion/ej2-
charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries);

```

```
let chart: Chart = new Chart({
  series: [
    {
      type: 'Line',
      dataSource: numData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
      }
    }
  ],
  // pointRender event for chart
  pointRender: (args: IPointRenderEventArgs) => {
    if (args.point.index === 3) {
      args.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">

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

  <div id="container">
    <div id="element"></div>
  </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.TS

```
import { Chart, LineSeries, Category } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
```

```

Chart.Inject(LineSeries, Category);
let chart: Chart = new Chart({
    series: [
        {
            type: 'Line',
            dataSource: numData, xName: 'x', yName: 'y',
            marker: {
                visible: true, isFilled: true,
                height: 10, width: 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>

```

See also

- [Customize the marker with different shape](#)

Data labels in EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Column',
            dataSource: columnData, xName: 'country', yName: 'gold',
            marker: {
                //Data label for chart series
                dataLabel: { 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>
```

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.TS

```
import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    primaryYAxis:
    {
        labelFormat: '{value}°C'
    },
    series: [
        {
            type: 'Column',
            dataSource: columnData, xName: 'country', yName: 'gold',
            marker: {
                //Data label position as middle
                dataLabel: { visible: true, position: 'Middle' }
            }
        }
    ],
}, '#element');
```

INDEX.HTML

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```
import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Column',
            dataSource: columnData, xName: 'country', yName: 'gold',
            marker: {
                dataLabel: { visible: true, template: '<div
style="background:#f5f5f5; border: 1px solid black; padding: 3px 3px 3px
3px"><div>${point.x}</div><div>${point.y}</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>

```

Text Mapping

Text from the data source can be mapped using **name** property.

INDEX.TS

```

import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Column',
            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',
            marker: {
                visible: true,
                dataLabel: { visible: true, name: 'text' }
            }
        }
    ],
}, '#element');

```

INDEX.HTML

```

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

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

    <div id="container">
        <div id="element"></div>
    </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.TS

```

import { Chart, ColumnSeries, Category, Legend, DataLabel, Tooltip,
ISeriesRenderEventArgs,
ITextRenderEventArgs } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(ColumnSeries, DataLabel, Category, Legend, Tooltip);
let total: any = [];
let chart: Chart = new Chart({
  //Initializing Primary X and Y Axis
  primaryXAxis: {
    valueType: 'Category',
    interval: 1,
    majorGridLines: { width: 0 },
  },
  chartArea: { border: { width: 0 } },
  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',
          format: 'n2'
        },
      },
    },
    {
      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',
                format: 'n2'
            },
        },
    },
    {
        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',
                format: 'n2'
            },
        },
    },
],
//Initializing Chart Title
width: Browser.isDevice ? '100%' : '60%',
title: 'Olympic Medal Counts - RIO',
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>

```

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.TS

```

import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-
charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Column',
            dataSource: columnData, xName: 'country', yName: 'gold',
            marker: {

```

```

        dataLabel: { visible: true,
                      border:{width: 1, color : 'red'},
                      margin:{
                        left:5,
                        right:5,
                        top:5,
                        bottom: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>

```

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.TS

```

import { Chart, ColumnSeries, Category, DataLabel } from '@syncfusion/ej2-
charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, DataLabel);
let chart: Chart = new Chart({

```

```

    primaryXAxis: {
        valueType: 'Category'
    },
    series: [
        {
            type: 'Column',
            dataSource: columnData, xName: 'country', yName: 'gold',
            marker: {
                dataLabel: { visible: true,
                    border:{width: 2, color : 'red'},
                    rx:10, ry: 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>

```

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.TS

```
import { Chart, LineSeries, DataLabel, IPointRenderEventArgs,
ITextRenderEventArgs } from '@syncfusion/ej2-charts';
import { numData } from './datasource.ts';
Chart.Inject(LineSeries, DataLabel);
let chart: Chart = new Chart({
  series: [
    {
      type: 'Line',
      dataSource: numData, xName: 'x', yName: 'y',
      marker: {
        visible: true,
        height: 10, width: 10,
        dataLabel: { visible: true }
      }
    }
  ],
  // pointRender event for chart
  pointRender: (args: IPointRenderEventArgs) => {
    if (args.point.index === 3) {
      args.fill = 'red'
    }
  },
  textRender: (args: ITextRenderEventArgs) => {
    if (args.point.index === 3) {
      args.text = 'Maximum Temperature';
      args.color = 'red';
    }
    else {
      args.cancel = 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 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.TS

```

import { Chart, ColumnSeries, Category, Legend, DataLabel, Tooltip,
ISeriesRenderEventArgs,
ITextRenderEventArgs } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
Chart.Inject(ColumnSeries, DataLabel, Category, Legend, Tooltip);
let total: any = [];
let chart: Chart = new Chart({
    //Initializing Primary X and Y Axis
    primaryXAxis: {
        valueType: 'Category',
        interval: 1,
        majorGridLines: { width: 0 },
    },
    chartArea: { border: { width: 0 } },
    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
width: Browser.isDevice ? '100%' : '60%',
title: 'Olympic Medal Counts - RIO',
tooltip: { enable: true },
seriesRender: (args: ISeriesRenderEventArgs) => {
    for (let 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: (args: ITextRenderEventArgs) => {
    let percentage: number | string = (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 EJ2 JavaScript 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 `<code>annotations</code> 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.TS

```
import { Chart, ColumnSeries, Category, ChartAnnotation } from
 '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, ChartAnnotation);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
```

```

    },
    annotations:[{
        content: '70 Gold Medals',
        coordinateUnits: 'Point',
        x: 'France',
        y: 50
    }],
    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>

```

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.TS

```
import { Chart, ColumnSeries, Category, ChartAnnotation } from
 '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, ChartAnnotation);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  annotations:[{
    content: '<div>Highest Medal Count</div>',
    region: 'Series',
    coordinateUnits: 'Point',
    x: 'France',
    y: 50
  }],
  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>
```

Co-ordinate Units

Specified the coordinates units of the annotation either **Pixel** or **Point**.

INDEX.TS

```
import { Chart, ColumnSeries, Category, ChartAnnotation } from
 '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, ChartAnnotation);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    annotations:[{
        content: '<div style="border: 1px solid black; padding: 5px 5px 5px
5px, background:#f5f5f5">Annotation in Pixel</div>',
        coordinateUnits: 'Pixel',
        x: 150,
        y: 50
    }],
    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>
```

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.TS

```
import { Chart, ColumnSeries, Category, ChartAnnotation } from
 '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, ChartAnnotation);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  annotations:[{
    content: '<div style="border: 1px solid black; padding: 5px 5px 5px
5px, background:#f5f5f5">Highest Medal Count</div>',
    x: 'France',
    y: 50,
    verticalAlignment:'Top',
    horizontalAlignment:'Near'
  }],
  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>

```

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.TS

```

import { Chart, ColumnSeries, Category, ChartAnnotation } from
'@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, ChartAnnotation);
let chart: Chart = new 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 EJ2 JavaScript 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);

```

```

let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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>
</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 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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>
  <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

You can align the legend as center, far or near to the chart using [alignment](#) property.

INDEX.TS

```
import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```
import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);

```

```

let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, LineSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(LineSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend, ILoadedEventArgs } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
  { 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
let colors: string[] = ['#00BDAE', '#404041', '#357CD2'];
let chart: Chart = new 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: ILoadedEventArgs) => {
        let chart: HTMLElement = document.querySelector('.e-chart');
        let legendTextCol: any =
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>
```

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.TS

```
import { Chart, ColumnSeries, Category, Legend, Selection } from
 '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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',
    selectionMode: 'Series',
    legendSettings: {
        visible: true,
        // series selection on legend
        toggleVisibility: 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>
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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```
import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, LineSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(LineSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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>
<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.TS

```

import { Chart, LineSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(LineSeries, Category, Legend);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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: {
    enablePages: false,
    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 EJ2 JavaScript 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.TS

```
import { Chart, StepLineSeries, DateTime, Tooltip } from '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
Chart.Inject(StepLineSeries, DateTime, Tooltip);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime'
    },
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            },
        },
    ],
    title: 'Unemployment Rates 1975-2010',
    //Default 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>
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.TS

```

import { Chart, StepLineSeries, DateTime, Tooltip } from '@syncfusion/ej2-
charts';
import { chartData } from './datasource.ts';
Chart.Inject(StepLineSeries, DateTime, Tooltip);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime'
  },
  primaryYAxis: {
    minimum: 0,
    maximum: 20,
    interval: 4
  },
  series: [
    {
      type: 'StepLine',
      dataSource: chartData, xName: 'x', yName: 'y',
      width: 2, name: 'China',
      marker: {
        visible: true, width: 10, height: 10
      },
    },
  ],
  title: 'Unemployment Rates 1975-2010',
  //Default 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.TS

```
import { Chart, SplineSeries, DateTime, Tooltip } from '@syncfusion/ej2-
charts';
import { chartData } from './datasource.ts';
Chart.Inject(SplineSeries, DateTime, Tooltip);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  series: [
    {
      type: 'Spline',
      dataSource: chartData, xName: 'x', yName: 'y',
      width: 2, name: 'China',
      marker: {
```

```

        visible: true, width: 10, height: 10
    },
    },
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {
        enable: true,
        //tooltip format for chart
        header: 'Unemployment',
        format: '<b>${point.x} : ${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 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.TS

```
import { Chart, StepLineSeries, Legend, Tooltip } from '@syncfusion/ej2-charts';
import { data } from './datasource.ts';
Chart.Inject(StepLineSeries, Legend, Tooltip);
let chart: Chart = new Chart({
    series: [
        {
            type: 'StepLine',
            dataSource: data, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            marker: {
                visible: true, width: 10, height: 10
            },
        }
    ],
    title: 'Unemployment Rates 1975-2010',
    tooltip: {
        enable: true,
        //tooltip template for chart
        template: '#Unemployment'
    }
}, '#element');
```

INDEX.HTML

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

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

```

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.TS

```

import { Chart, StepLineSeries, DateTime, Legend, Tooltip } from
 '@syncfusion/ej2-charts';
import { chartData } from './datasource.ts';
Chart.Inject(StepLineSeries, DateTime, Legend, Tooltip);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    //Highlight color for tooltip
    highlightColor: 'red',
    series: [
        {
            type: 'StepLine',
            dataSource: chartData, xName: 'x', yName: 'y',
            width: 2, name: 'China',
            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 EJ2 JavaScript 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 zoomin and zoomout 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.TS

```
import { Chart, AreaSeries, Legend, Zoom, DateTime } from '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    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.TS

```

import { Chart, AreaSeries, Legend, Zoom, DateTime } from '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  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.TS

```
import { Chart, AreaSeries, Legend, Zoom, DateTime } from '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    zoomSettings: {
        enableSelectionZooming: true, showToolbar: 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>

```

Enable pan

Using [enablePan](#) property you can able to pan the zoomed chart without help of toolbar items.

INDEX.TS

```

import { Chart, AreaSeries, Legend, Zoom, DateTime } from '@syncfusion/ej2-
charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        zoomFactor: 0.2, zoomPosition: 0.6
    },
    series: [
        {
            type: 'Area',
            dataSource: series1,
            name: 'Product X',
            xName: 'x',
            yName: 'y',
            border: { width: 0.5, color: '#00bdae' },
            animation: { enable: false }
        },
    ],
    zoomSettings:
    {
        enableSelectionZooming: true,
        enablePan: 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.TS

```

import { Chart, AreaSeries, Legend, Zoom, DateTime, ScrollBar } from
'@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom, ScrollBar);
let chart: Chart = new Chart({
    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 }
        },
    ],
    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.TS

```

import { Chart, AreaSeries, Legend, Zoom, DateTime } from '@syncfusion/ej2-
charts';
import { series1 } from './datasource.ts';
Chart.Inject(AreaSeries, DateTime, Legend, Zoom);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        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,
    },
    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 EJ2 JavaScript 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.TS

```

import { Chart, ColumnSeries, DateTime, Legend, LineSeries, Tooltip,
DataEditing } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DateTime, Legend, LineSeries, Tooltip,
DataEditing);
let chart: Chart = new 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 EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, Crosshair, DateTime, Legend } from
 '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Crosshair, Legend);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
    },
    series: [
        {
            type: 'Line', width: 2, name: 'Temperature',
            dataSource: series1, xName: 'x', yName: 'y'
        }
    ],
    //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.TS

```

import { Chart, LineSeries, Crosshair, DateTime, Legend } from
 '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Crosshair, Legend);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
    crosshairTooltip: { enable: true },
  },
  primaryYAxis:
  {
    crosshairTooltip: { enable: true }
  },
  series: [
    {
      type: 'Line', width: 2, name: 'Temperature',
      dataSource: series1, xName: 'x', yName: 'y'
    }
  ],
  //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.TS

```

import { Chart, LineSeries, Crosshair, DateTime, Legend } from
 '@syncfusion/ej2-charts';
import { series1 } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Crosshair, Legend);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        crosshairTooltip: { enable: true, fill: 'green' },
    },
    primaryYAxis:
    {
        crosshairTooltip: { enable: true, fill: 'green' },
    },
    series: [
        {
            type: 'Line', width: 2, name: 'Temperature',
            dataSource: series1, xName: 'x', yName: 'y'
        }
    ],
    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.TS

```

import { Tooltip, Crosshair, DateTime } from '@syncfusion/ej2-charts';
import { Chart, LineSeries, Legend } from '@syncfusion/ej2-charts';
import { trackData } from './datasource.ts';
Chart.Inject(LineSeries, DateTime, Tooltip, Crosshair, Legend);

```

```

let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'DateTime',
  },
  series: [
    {
      dataSource: trackData, name: 'John', xName: 'x',
      marker: { visible: true },
      type: 'Line', width: 2,
      yName: 'y'
    },
    {
      dataSource: trackData, name: 'Andrew', xName: 'x',
      marker: { visible: true },
      type: 'Line', width: 2,
      yName: 'y1'
    },
    {
      dataSource: trackData, name: 'Thomas', xName: 'x',
      marker: { visible: true },
      type: 'Line', width: 2,
      yName: 'y2'
    },
    {
      dataSource: trackData, name: 'Mark', xName: 'x',
      marker: { visible: true },
      type: 'Line', width: 2,
      yName: 'y3'
    },
    {
      dataSource: trackData, 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 EJ2 JavaScript 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.TS

```

import { Chart, AreaSeries, LineSeries, DateTime, Tooltip, IMouseEventArgs,
ITooltipRenderEventArgs, ILegendClickEventArgs } from '@syncfusion/ej2-
charts';
Chart.Inject(AreaSeries, LineSeries, DateTime, Tooltip);
import { Browser } from '@syncfusion/ej2-base';
import { synchronizedData } from './datasource.ts';
let charts: Chart[] = [];
let chart: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IMouseEventArgs) => {
        chart1.hideTooltip();
    },
    chartMouseMove: (args: IMouseEventArgs) => {
        if ((!Browser.isDevice && !chart.isTouch && !chart.isChartDrag) ||
chart.startMove) {
            chart1.startMove = chart.startMove;
            chart1.showTooltip(args.x, args.y);
        }
    },
    chartMouseUp: function (args: IMouseEventArgs) {
        if (Browser.isDevice && chart.startMove) {
            chart1.hideTooltip();
        }
    },
    title: 'US to EURO',
    titleStyle: { textAlign: 'Near' },
    tooltip: { enable: true, fadeOutDuration: Browser.isDevice ? 2500 :
1000, shared: true, header: '', format: '<b>€${point.y}</b> <br> ${point.x}
2023', enableMarker: false },
    });
chart.appendTo('#container1');
charts.push(chart);
let chart1: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IMouseEventArgs) => {
        if ((!Browser.isDevice && !chart1.isTouch && !chart1.isChartDrag) ||
chart1.startMove) {
            chart.startMove = chart1.startMove;
            chart.showTooltip(args.x, args.y);
        }
    },
    chartMouseLeave: (args: IMouseEventArgs) => {
        chart.hideTooltip();
    },
    chartMouseUp: function (args: IMouseEventArgs) {
        if (Browser.isDevice || chart1.startMove) {
            chart.hideTooltip();
        }
    },
    title: 'US to INR',
    titleStyle: { textAlignment: 'Near' },
    tooltip: { enable: true, fadeOutDuration: 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.TS

```

import { Chart, AreaSeries, SplineSeries, DateTime, Crosshair,
MouseEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(AreaSeries, SplineSeries, DateTime, Crosshair);
import { Browser } from '@syncfusion/ej2-base';
import { synchronizedData } from './datasource.ts';
let charts: Chart[] = [];
let chart: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IMouseEventArgs) => {
        chart1.hideCrosshair();
    },
    chartMouseMove: (args: IMouseEventArgs) => {
        if ((!Browser.isDevice && !chart.isTouch && !chart.isChartDrag) ||
chart.startMove) {
            chart1.startMove = chart.startMove;
            chart1.showCrosshair(args.x, args.y);
        }
    },
    chartMouseUp: function (args: IMouseEventArgs) {
        if (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);
let chart1: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IMouseEventArgs) => {
        if (!!Browser.isDevice && !chart1.isTouch && !chart1.isChartDrag) ||
chart1.startMove) {
            chart.startMove = chart1.startMove;
            chart.showCrosshair(args.x, args.y);
        }
    },
    chartMouseLeave: (args: IMouseEventArgs) => {
        chart.hideCrosshair();
    },
    chartMouseUp: function (args: IMouseEventArgs) {
        if (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.TS

```

import { Chart, SplineAreaSeries, LineSeries, DateTime, Zoom, ZoomSettings,
IZoomCompleteEventArgs, IMouseEventArgs, ITooltipRenderEventArgs,
ILegendClickEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(SplineAreaSeries, LineSeries, DateTime, Zoom);
import { Browser } from '@syncfusion/ej2-base';
import { synchronizedData } from './datasource.ts';
import { Axis } from '@syncfusion/ej2/charts';
let charts: Chart[] = [];
let zoomFactor: number = 0;
let zoomPosition: number = 0;
let chart: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IZoomCompleteEventArgs) => {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    },
    title: 'US to EURO',
    titleStyle: { textAlignment: 'Near' },
});
chart.appendTo('#container1');
charts.push(chart);
let chart1: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IZoomCompleteEventArgs) => {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    },
    title: 'US to INR',
    titleStyle: { textAlignment: 'Near' },
});
chart1.appendTo('#container2');
charts.push(chart1);
function zoomCompleteFunction(args: IZoomCompleteEventArgs): void {
    for (let i: number = 0; i < charts.length; i++) {
        if ((args.axis as 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 as
Axis).series[0].chart.zoomModule.isZoomed;
            charts[i].zoomModule.isPanning = (args.axis as
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.TS

```

import { Chart, SplineSeries, LineSeries, DateTime, Highlight, Zoom,
ZoomSettings, IZoomCompleteEventArgs, Selection,
ISelectionCompleteEventArgs, IMouseEventArgs, ITooltipRenderEventArgs,
ILegendClickEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(SplineSeries, LineSeries, DateTime, Zoom, Highlight,
Selection);
import { Browser } from '@syncfusion/ej2-base';
import { synchronizedData } from './datasource.ts';
import { Axis } from '@syncfusion/ej2/charts';
let charts: Chart[] = [];
let zoomFactor: number = 0;
let zoomPosition: number = 0;
let count: number = 0;
let chart: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IZoomCompleteEventArgs) => {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    },
    selectionComplete: function (args: ISelectionCompleteEventArgs | any) {
        selectionCompleteFunction(args);
    },
    selectionPattern: 'Box',
    selectionMode: 'Point',
    title: 'US to EURO',
    titleStyle: { textAlign: 'Near' }
});
chart.appendTo('#container1');
charts.push(chart);
let chart1: Chart = new 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: Browser.isDevice ? 'None' : 'Shift',
        labelRotation: Browser.isDevice ? -45 : 0,
        interval: 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: (args: IZoomCompleteEventArgs) => {
        if (args.axis.name === 'primaryXAxis') {
            zoomFactor = args.currentZoomFactor;
            zoomPosition = args.currentZoomPosition;
            zoomCompleteFunction(args);
        }
    },
    selectionComplete: function (args: ISelectionCompleteEventArgs | any) {
        selectionCompleteFunction(args);
    },
    selectionPattern: 'Box',
    selectionMode: 'Point',
    title: 'US to INR',
    titleStyle: { textAlignment: 'Near' }
});
chart1.appendTo('#container2');
charts.push(chart1);
function zoomCompleteFunction(args: IZoomCompleteEventArgs): void {
    for (let i: number = 0; i < charts.length; i++) {
        if ((args.axis as 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 as
Axis).series[0].chart.zoomModule.isZoomed;
            charts[i].zoomModule.isPanning = (args.axis as
Axis).series[0].chart.zoomModule.isPanning;
        }
    }
}
function selectionCompleteFunction(args: ISelectionCompleteEventArgs | any):
void {
    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 EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, Category, Legend, Selection } from
 '@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: selectionData,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: selectionData,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: selectionData,
    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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: selectionData ,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: selectionData ,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: selectionData ,
        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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts';

```

```

Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category',
  },
  series:[{
    dataSource: selectionData ,
    xName: 'country', yName: 'gold',
    name: 'Gold', type: 'Column'
  }, {
    dataSource: selectionData ,
    xName: 'country', yName: 'silver',
    name: 'Silver', type: 'Column'
  }, {
    dataSource: selectionData ,
    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.TS

```
import { Chart, ScatterSeries, Legend, Selection } from '@syncfusion/ej2-charts';
import { rectData } from './datasource.ts';
Chart.Inject(ScatterSeries, Legend, Selection);
let chart: Chart = new Chart({
  series: [
    {
      type: 'Scatter',
      dataSource: rectData,
      xName: 'x',
      yName: 'y', name: 'Product A',
      marker: {
        shape: 'Triangle',
        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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts';
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: selectionData ,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column'
    }, {
        dataSource: selectionData ,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column'
    }, {
        dataSource: selectionData ,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column'
    }],
});

```

```

selectionMode: 'Series',
// 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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts'
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
  primaryXAxis: {

```

```

        valueType: 'Category',
    },
    series:[{
        dataSource: selectionData ,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection1'
    }, {
        dataSource: selectionData ,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection2'
    }, {
        dataSource: selectionData ,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column',
        animation: {enable: false},
        selectionStyle: 'chartSelection3'
    }],
    selectionMode: 'Point',
    isMultiSelect: true,
    // Selcted data indexes for chart series
    selectedDataIndexes: [
        { series: 0, point: 1}, { series: 2, point: 3}
    ],
    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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection, Highlight } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts'
Chart.Inject(ColumnSeries, Category, Legend, Selection, Highlight);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: selectionData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        animation: {enable: false},
    }, {
        dataSource: selectionData,
        xName: 'country', yName: 'silver',
        name: 'Silver', type: 'Column',
        animation: {enable: false},
    }, {
        dataSource: selectionData,
        xName: 'country', yName: 'bronze',
        name: 'Bronze', type: 'Column',
        animation: {enable: false},
    }],
    // Selection through on legend
    legendSettings: { visible: true, toggleVisibility: false,
    enableHighlight: 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>

```

Customization for selection

You can apply custom style to selected points or series with [selectionStyle](#) property.

INDEX.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
'@syncfusion/ej2-charts';
import { selectionData } from './datasource.ts'
Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category',
    },
    series:[{
        dataSource: selectionData,
        xName: 'country', yName: 'gold',
        name: 'Gold', type: 'Column',
        //Selection style for chart series selection
        selectionStyle: 'chartSelection1'
    }, {
        dataSource: selectionData,
        xName: 'country', yName: 'silver',

```

```

        name: 'Silver', type: 'Column',
        selectionStyle: 'chartSelection2'
    }, {
        dataSource: selectionData,
        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 EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
import { Button } from '@syncfusion/ej2-buttons';
import { EmitType } from '@syncfusion/ej2-base';
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend, Export } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend, Export);
import { EmitType } from '@syncfusion/ej2-base';
let chart: Chart = new 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.exportModule.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.TS

```
import { Chart, ColumnSeries, Category, Legend, Export } from
 '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend, Export);
import { EmitType } from '@syncfusion/ej2-base';
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, LineSeries, DateTime, Category, Legend,
Export, AccumulationChart, AccumulationLegend, PieSeries,
AccumulationDataLabel } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, LineSeries, Category, DateTime, Legend, Export);
AccumulationChart.Inject(AccumulationLegend, PieSeries,
AccumulationDataLabel);
import { EmitType } from '@syncfusion/ej2-base';
let chart = new Chart({
    //Initializing Primary X Axis
    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');
let chart1: Chart = new 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');
let pie: AccumulationChart = new 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'
});
pie.appendTo('#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.TS

```

import { Chart, ColumnSeries, Category, Legend, Export } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend, Export);
import { EmitType } from '@syncfusion/ej2-base';
let chart: Chart = new 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');
let chart1: Chart = new 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 = () => {
    chart.exportModule.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>
<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.TS

```

import { Chart, ColumnSeries, Category, Legend, Tooltip } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend, Tooltip);
let chart: Chart = new 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 = () => {
    let svg: any = 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;
    let ctx: any = 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 EJ2 JavaScript 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, Legend);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, Legend, Selection } from
 '@syncfusion/ej2-charts';

```

```

Chart.Inject(ColumnSeries, Category, Legend, Selection);
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, ISeriesRenderEventArgs,
IPointRenderEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
  { 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 }
];
let colors: string[] = ['#00bdae', '#404041', '#357cd2', '#e56590',
    '#f8b883',
    '#70ad47', '#dd8abd', '#7f84e8', '#7bb4eb', '#ea7a57'];
let chart: Chart = new 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: IPointRenderEventArgs): void => {
        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.TS

```
import { Chart, ColumnSeries, Category, ISeriesRenderEventArgs } from
 '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, ColumnSeries, Category, ISeriesRenderEventArgs } from
'@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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>
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.TS

```

import { Chart, ColumnSeries, Category, ISeriesRenderEventArgs } from
 '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
  { 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 }
  ];
  let chart: Chart = new 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.TS

```
import { Chart, ColumnSeries, Category, ISeriesRenderEventArgs } from
 '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new 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.TS

```

import {
    Chart, ColumnSeries, Category, DataLabel, Tooltip, ILoadedEventArgs
} from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Category, Tooltip);
let count: number = 0;
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'Category', interval: 1, majorGridLines: { width: 0
    },
        tickPosition: 'Inside',
        labelPosition: 'Inside', labelStyle: { color: '#ffffff' }
    },
    chartArea: { border: { width: 0 } },
    //Initializing Primary Y Axis
    primaryYAxis:
    {
        minimum: 0, maximum: 300, interval: 50, majorGridLines: {
width: 0 },
        majorTickLines: { width: 0 }, lineStyle: { width: 0 },
labelStyle: { color: 'transparent' }
    },
    //Initializing Chart Series
    series: [
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y',
            dataSource: [
                { x: 'Egg', y: 106 },
                { x: 'Fish', y: 103 },

```

10

```

        { x: 'Misc', y: 198 },
        { x: 'Tea', y: 189 },
        { x: 'Fruit', y: 250 }
    ], name: 'Tiger',
    cornerRadius: {
        bottomLeft: 10, bottomRight: 10, topLeft: 10, topRight:
    },
    },
    ],
    legendSettings: { visible: false },
    //Initializing Chart title
    title: 'Trade in Food Groups', tooltip: { enable: false },
    loaded: (args: ILoadedEventArgs) => {
        let columninterval = setInterval(
            () => {
                if (document.getElementById('element')) {
                    if (count === 0) {
                        chart.series[0].dataSource = [
                            { x: 'Egg', y: 206 },
                            { x: 'Fish', y: 123 },
                            { x: 'Misc', y: 48 },
                            { x: 'Tea', y: 240 },
                            { x: 'Fruit', y: 170 }
                        ];
                        chart.animate();
                        count++;
                    }
                    else if (count === 1) {
                        chart.series[0].dataSource = [
                            { x: 'Egg', y: 86 },
                            { x: 'Fish', y: 173 },
                            { x: 'Misc', y: 188 },
                            { x: 'Tea', y: 109 },
                            { x: 'Fruit', y: 100 }
                        ];
                        chart.animate();
                        count++;
                    }
                    else if (count === 2) {
                        chart.series[0].dataSource = [
                            { x: 'Egg', y: 156 },
                            { x: 'Fish', y: 33 },
                            { x: 'Misc', y: 260 },
                            { x: 'Tea', y: 200 },
                            { x: 'Fruit', y: 30 }
                        ];
                        chart.animate();
                        count = 0;
                    }
                } else {
                    clearInterval(columninterval);
                }
            },
            2000
        );
    },
    },

```

```
}, '#element');
```

INDEX.HTML

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

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

```
import { Chart, StepLineSeries, DateTime, Legend, Tooltip } from
'@syncfusion/ej2-charts';
Chart.Inject(StepLineSeries, DateTime, Legend, Tooltip);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, StepLineSeries, DateTime, Legend, Tooltip } from
'@syncfusion/ej2-charts';
Chart.Inject(StepLineSeries, DateTime, Legend, Tooltip);
let chartData: any[] = [
    { 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 }
];
let chart: Chart = new Chart({
    primaryXAxis: {
        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.TS

```

import { Chart, StepLineSeries, DateTime, Legend, Tooltip } from
 '@syncfusion/ej2-charts';
Chart.Inject(StepLineSeries, DateTime, Legend, Tooltip);
let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, StepLineSeries, DateTime, Legend, Tooltip } from
 '@syncfusion/ej2-charts';
Chart.Inject(StepLineSeries, DateTime, Legend, Tooltip);

```

```

let chartData: any[] = [
  { 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 }
];
let chart: Chart = new 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.TS

```

import { Chart, LineSeries } from '@syncfusion/ej2-charts';
import { Legend, Category, Tooltip } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Legend, Category, Tooltip);
let chartData: any[] = [
  { 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: Chart = new Chart({
  primaryXAxis: {
    valueType: 'Category'
  },
  title: 'Sales Analysis',

```

```
textStyle:{size:'18px', color:'Red', textAlignment: 'Far', textOverflow:
'Wrap' },
series:[{
  dataSource: chartData,
  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 EJ2 JavaScript 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.

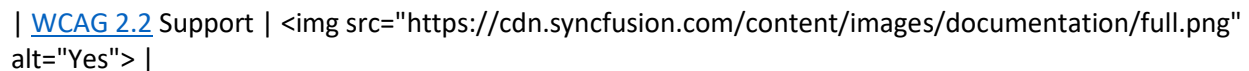
Accessibility in EJ2 JavaScript Chart control

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 |

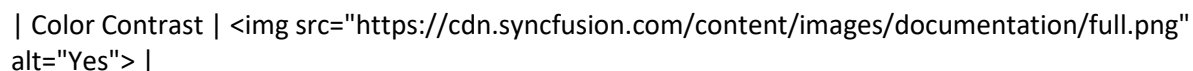
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

| Keyboard Navigation Support |  alt="Yes"> |

| [Accessibility Checker](#) Validation |  alt="Yes"> |

| [Axe-core](#) Accessibility Validation |  alt="Yes"> |

<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 EJ2 JavaScript controls](#)

Internationalization in EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, DataLabel, Tooltip } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Tooltip);
import { loadCldr, L10n, setCulture, setCurrencyCode }
from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
let chartData: any[] = [
    { 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 }
  ]
let chart: Chart = new 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>
  <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 EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, DataLabel, Zoom } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, DataLabel, Zoom);
import { L10n } from '@syncfusion/ej2-base';
let chartData: any[] = [
    { 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 }
]
L10n.load({
    'ar-AR': {
        'chart': {
            ZoomIn: 'تكبير',
            ZoomOut: 'تصغير',
            Zoom: 'زوم',
            Pan: 'مقلاة',
            Reset: 'إعادة تعيين',
            ResetZoom: 'زوم إعادة تعيين'
        }
    },
});
let chart: Chart = new 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 EJ2 JavaScript 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	<code>Property:annotations.angle\$("#chart").ejChart({ annotations: [{angle: 90}]});</code>	<code>Property:annotations.rotationlet chart: Chart = new Chart({annotations: [{rotation: 90}]});chart.appendTo('#chart');</code>
annotations	<code>Property:annotations.content\$("#chart").ejChart({ annotations: [{ content: 'Chart'}]});</code>	<code>Property:annotations.contentlet chart: Chart = new Chart({annotations: [{content: 'Chart'}]});chart.appendTo('#chart');</code>
coordinate unit for annotation	<code>Property:annotations.coordinateUnit\$("#container").ejChart({ annotations :[{coordinateUnit : "pixels"}]});</code>	<code>Property:annotations.coordinateUnitslet chart: Chart = new Chart({annotations: [{coordinateUnits: 'Pixel'}]});chart.appendTo('#chart');</code>
horizontalAlignment for annotation	<code>Property:annotations.horizontalAlignment\$("#container").ejChart({ annotations :[{ horizontalAlignment : "middle"}]});</code>	<code>Property:annotations.horizontalAlignmentlet chart: Chart = new Chart({annotations: [{horizontalAlignment: 'Center'}]});chart.appendTo('#chart');</code>
margin for annotation	<code>Property:annotations.margin\$("#container").ejChart({ annotations :[{ margin { right: 5, left: 5, top: 5, bottom: 5}}]});</code>	Not applicable
Opacity for annotation	<code>Property:annotations.opacity\$("#container").ejChart({ annotations :[{ opacity: 0.4 }]});</code>	Not applicable
Region for annotation with respect to chart or series	<code>Property:annotations.region\$("#container").ejChart({ annotations :[{ region : "chart"}]});</code>	<code>Property:annotations.regionlet chart: Chart = new Chart({annotations: [{region: 'Chart'}]});chart.appendTo('#chart');</code>

verticalAlignm ent for annotation	Property:annotations.verticalAlignment \$("#container").ejChart({ annotations :[{ verticalAlignment : "middle" }]});	Property:annotations.verticalAlignm entlet 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.xlet 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.xAxisNamelet chart: Chart = new Chart({annotations: [{ xAxisName: 'xAxis' }]});chart.appendTo('#chart');
Y offset for annotation	Property:annotations.y\$("#container").ejCha rt({ annotations : [{ y : "100" }]});	Property:annotations.ylet 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 :[{ xAxisName : "yAxis" }]});	Property:annotations.yAxisNamele t chart: Chart = new Chart({annotations: [{ xAxisName: '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").ej Chart({ columnDefinitions: []});	Property:columnslet chart: Chart = new Chart({ columns: []});chart.appendTo('#chart');
unit	Property:unit \$("#container").ejChart({ columnDefinitions : [{unit : "percentage" }]});	Not Applicable
width of columns in chart	Property:columnWidth\$("#chart").ejChar t({ columnDefinitions: [{ columnWidth: '50%' }]});	Property:widthlet chart: Chart = new Chart({ columns: [{ width:

		'300'}}]);chart.appendTo('#chart');
Line customization	Property:lineColor, lineWidth\$("#chart").ejChart({ columnDefinitions: [{ columnWidth: '50%', lineColor: 'brown', lineWidth: 2}]]});	Property:borderlet chart: Chart = new Chart({ columns: [{ width: '300', border: { width: 2, color: 'brown'}}]]});chart.appendTo('#chart');

CommonSeriesOptions

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
commonSeriesOptions	Property:commonSeriesOptions \$("#container").ejChart({ commonSeriesOptions: { } });	Not Applicable

Crosshair

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
crosshair	Property:visible\$("#container").ejChart({ crosshair: { visible: true}});	Property:enablelet chart: Chart = new Chart({ crosshair: { enable: false }});chart.appendTo('#chart');
trackballTooltipSettings	Property:trackballTooltipSettings\$("#container").ejChart({ crosshair: { trackballTooltipSettings: { border: { width:2 } }}});	Not applicable
marker	Property:marker\$("#container").ejChart({ crosshair: { marker	Not applicable

	<pre>: { border : { width :2 } } }});</pre>			
crosshair line style	<pre>Property:line\$ ("#contain er").ejCha rt({ crosshair : { line: { width: 2, color: 'red' } }});</pre>	<pre>let chart: Chart = new Chart({ crosshair: { border: { width: 2, color: 'black' }}});</pre>		
type	<pre>Property:type \$("#contai ner").ejCh art({ crosshair : { type: 'trackball ' }});</pre>	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:typelet 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: [{ standardDeviation: 2 }] });
		standard deviation	Property:standardDeviations\$("#container").ejChart({ indicators: [{ standardDeviations: 2 }] });	Property:standardDeviationlet chart: Chart = new

				<pre>Chart({ indicators: [{ standardDeviation: 2 }] });</pre>
	Field for indicator	Property: field\$("#container").ejChart({ indicators: [{ field: 'Close' }] });		Property: fieldlet chart: Chart = new Chart({ indicators: [{ field: 'Close' }] });
	Slow period for MACD indicator	Property: shortPeriod\$("#container").ejChart({ indicators: [{ shortPeriod: 12 }] });		Property: slowPeriodlet t chart: Chart = new Chart({ indicators: [{ slowPeriod: 12 }] });
	Fast period for MACD indicator	Property: longPeriod\$("#container").ejChart({ indicators: [{ longPeriod: 26 }] });		Property: fastPeriodlet t chart: Chart = new Chart({ indicators: [{ fastPeriod: 26 }] });
	Line style for MACD indicator	Property: macdLine\$("#container").ejChart({ indicators: [{ macdLine: { width: 2, fill: 'red' } }] });		Property: fastPeriodlet t chart: Chart = new Chart({ indicators: [{ macdLine: { width: 2, color: 'red' } }] });
	Type of MACD indicator	Property: macdType\$("#container").ejChart({ indicators: [{ macdType: 'both' }] });		Property: fastPeriodlet t chart: Chart = new Chart({ indicators: [{ macdType: 'Both' }] });
	Color of the positive bars in Macd indicators	Not applicable		Property: macdPositiveColorlet chart: Chart = new Chart({ indicators: [{ macdPositiveColor: 'red' }] });
	Color of the negative bars in Macd indicators	Not applicable		Property: macdNegativeColorlet 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").ej Chart({ 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").ej Chart({ 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").ej Chart({ 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").ej Chart({ indicators: [{ seriesName: '' }] });	Property:lowerLine let chart: Chart = new Chart({ indicators: [{ seriesName: '' }] });
		Options to customize the histogram in MACD indicator	Property:seriesName\$("#container").ej Chart({ 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:backgorundlet 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: { }, textAlignment: '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	Property:fill\$("#container").ejChart({ legend: { fill: '#EEFFCC' }});	Not applicable

		legend items		
		## 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 crossed	Property:crossesInAxis\$("#container").ejChart({ primaryXAxis: { crossesInAxis: '' }});	Property:crossesInAxislet chart: Chart = new Chart({ primaryXAxis: { crossesInAxis: '' }});chart.appendTo('#chart');
		axis element's placement	Property:showNextToAxisLine \$("#container").ejChart({ primaryXAxis:	Property:placeNextToAxisLinelet chart: Chart = new Chart({ primaryXAxis: { placeNextToAxisLine: '' }});chart.appendTo('#chart');

		ed wit h axis line	s: { showNextToA xisLine : true }));	
		axis line styl e	Property:axisLi ne.color\$("#c ontainer"). ejChart({ primaryXAxis: s: { axisLine: { color : 'red' } }});	Property:lineStyle.colorlet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { color: 'black' } } });chart.appendTo('#chart');
		axis line dash Array	Property:axisLi ne.color\$("#c ontainer"). ejChart({ primaryXAxis: s: { axisLine: { dashArray : '10, 5' } }});	Property:lineStyle.dashArraylet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { dashArray: '10, 5' } }});chart.appendTo('#chart');
		Offs et for axis	Property:axisLi ne.offset\$("#c ontainer"). ejChart({ primaryXAxis: s: { axisLine: { offset : 10 } }));	Property:plotOffsetlet chart: Chart = new Chart({ primaryXAxis: { plotOffset: 10 }});chart.appendTo('#chart');
		Visi ble of an axis	Property:axisLi ne.offset\$("#c ontainer"). ejChart({ primaryXAxis: s: { axisLine: { visible : false } }});	Property:visiblelet chart: Chart = new Chart({ primaryXAxis: { visible: false }});chart.appendTo('#chart');
		Wid th of an axis	Property:axisLi ne.width\$("#c ontainer"). ejChart({ primaryXAxis: s: {	Property:lineStyle.widthlet chart: Chart = new Chart({ primaryXAxis: { lineStyle: { width: 3 } } });chart.appendTo('#chart');

			axisLine: { width : 2 } }});	
		Column index of an axis	Property:columnIndex\$("#container").ejChart({ primaryXAxis: { columnIndex : 2 } });	Property:columnIndexlet chart: Chart = new Chart({ primaryXAxis: { columnIndex: 2 } });chart.appendTo('#chart');
		span of an axis to place horizontally or vertically	Property:columnSpan\$("#container").ejChart({ primaryXAxis: { columnIndex : 2 } });	Property:spanlet chart: Chart = new Chart({ primaryXAxis: { span: 2 } });chart.appendTo('#chart');
		Crosshair label of an axis	Property:crossHairLabel.visible\$("#container").ejChart({ primaryXAxis: { crossHairLabel: { visible: true } } });	Property:crossHairTooltip.enablelet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { enable: true } } });chart.appendTo('#chart');
		Crosshair label color of an axis	Not applicable	Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { fill: 'red' } } });chart.appendTo('#chart');

		Crosshair label text style	Not applicable	Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { textStyle: { } } } });chart.appendTo('#chart');
		Desired interval count for primary XAxis	Property:desiredIntervals\$("#container").ejChart({ primaryXAxis: { desiredIntervals: 4 } });	Property:desiredIntervals let chart: Chart = new Chart({ primaryXAxis: { desiredIntervals: 4 } });chart.appendTo('#chart');
		Edge label placement for primary XAxis	Property:edgeLabelPlacement\$("#container").ejChart({ primaryXAxis: { edgeLabelPlacement: 'none' } });	Property:edgeLabelPlacement let chart: Chart = new Chart({ primaryXAxis: { edgeLabelPlacement: 'Shift' } });chart.appendTo('#chart');
		Enable trim for axis labels	Property:enableTrim\$("#container").ejChart({ primaryXAxis: { enableTrim: true } });	Property:enableTrim let chart: Chart = new Chart({ primaryXAxis: { enableTrim: true } });chart.appendTo('#chart');
		Specify the interval of the axis	Property:enableAutoIntervalOnZooming\$("#container").ejChart({ primaryXAxis: { enableAutoIntervalOnZooming: true } });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true } });chart.appendTo('#chart');

		ording to the zoomed data of the chart		
		Specifies the interval of the axis according to the zoomed data of the chart	Property:enableAutoIntervalOnZooming \$("#container").ejChart({ primaryXAxis: { enableAutoIntervalOnZooming: true } });	Property:enableAutoIntervalOnZooming let chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true } });chart.appendTo('#chart');
		Font style for primary XAxis	Property:font \$("#container").ejChart({ primaryXAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: } });	Property:titleStyle let chart: Chart = new Chart({ primaryXAxis: { titleStyle: { } } });chart.appendTo('#chart');

			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');	
	Label Intersection	Property:labelIntersection\$("#container").ejChart({ primaryXAxis: { labelIntersectionAction: 'trim' } });	Property:labelIntersectionlet chart: Chart = new Chart({ primaryXAxis: { labelIntersectionAction: 'Trim' } });chart.appendTo('#chart');	

		labelPosition	Property:labelPosition\$("#container").ejChart({ primaryXAxis: { labelPosition: 'inside' } }));	Property:labelPositionlet chart: Chart = new Chart({ primaryXAxis: { labelPosition: 'Inside' } });chart.appendTo('#chart');
		labelPlacement 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	Property:majorGridLines.visibility	Not Applicable

		grid line	<code>e\$("#container").ejChart({ primaryXAxis: { majorGridLines: { visible: true } } });</code>	
		Width of Major Grid Lines	Property:majorGridLines.width <code>e\$("#container").ejChart({ primaryXAxis: { majorGridLines: { width: 2 } } });</code>	Property:majorGridLines.width <code>let chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { width: 2 } } });chart.appendTo('#chart');</code>
		Color of Major Grid Lines	Property:majorGridLines.color <code>e\$("#container").ejChart({ primaryXAxis: { majorGridLines: { color: 'black' } } });</code>	Property:majorGridLines.color <code>let chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { color: 'black' } } });chart.appendTo('#chart');</code>
		Dash Array of Major Grid Lines	Property:majorGridLines.dashArray <code>e\$("#container").ejChart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } } });</code>	Property:majorGridLines.dashArray <code>let chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');</code>
		Opacity of major	Property:majorGridLines.opacity <code>e\$("#container").ejChart({ primaryXAxis: {</code>	Not Applicable

		grid line	majorGridLines: { opacity: true} }));	
		Major or Tick line	Property:major TickLines.visible\$(" #container").ejChart({ primaryXAxis: { majorTickLines: { visible: true} }));	Not Applicable
		Width of Major Tick Lines	Property:major TickLines.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:major TickLines.size\$(" #container").ejChart({ primaryXAxis: { majorTickLines: { size: 2} }});	Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { height: 2} } });chart.appendTo('#chart');
		Color of Major Tick Lines	Property:major TickLines.color\$(" #container").ejChart({ primaryXAxis: { majorTickLines: { color: 'black' } }});	Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } } });chart.appendTo('#chart');

		Property:major TickLines.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:minor GridLines.visible\$("#container").ejChart({ primaryXAxis: { minorGridLines: { visible: true} }));	Not Applicable

		Width of minor Gridlines	Property:minorGridLines.width \$("#container").ejChart({ primaryXAxis: { minorGridLines: { width: 2} }}});	Property:minorGridLines.width let chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { width: 2 } } });chart.appendTo('#chart');
		Color of minor Gridlines	Property:minorGridLines.color \$("#container").ejChart({ primaryXAxis: { minorGridLines: { color: 'black' } }}});	Property:minorGridLines.color let chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { color: 'black' } } });chart.appendTo('#chart');
		DashArray of minor Gridlines	Property:minorGridLines.dashArray \$("#container").ejChart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } }}});	Property:minorGridLines.dashArray let chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
		Opacity of minor grid line	Property:minorGridLines.opacity \$("#container").ejChart({ primaryXAxis: { minorGridLines: { opacity: true } }}});	Not Applicable
		minor Tick line	Property:minorTickLines.visible \$("#container").ejChart({	Not Applicable

			<pre>primaryXAxis: { minorTickLines: { visible: true} }));</pre>	
		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>
		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

		Min or tick s per inte rval of pri mar yXA xis	Property:minor TicksPerInterva <code>l\$("#contain er").ejChar t({ primaryXAxi s: { minorTicksP erInterval: 4 }}});</code>	Property:minorTickLines.color <code>let chart: Chart = new Chart({ primaryXAxis: { minorTicksPerInterval: 4 }});chart.appendTo('#chart');</code>
		na me of the pri mar yXA xis	Property:name <code>l\$("#contain er").ejChar t({ primaryXAxi s: { name: 'primaryXA xis' }}});</code>	Property:name <code>let chart: Chart = new Chart({ primaryXAxis: { name: 'primaryXAxis' }});chart.appendTo('#chart');</code>
		Orie ntat ion of pri mar yXA xis	Property:orient ation <code>l\$("#con tainer").ej Chart({ primaryXAxi s: { orientation : 'Vertical' }}});</code>	Not Applicable
		Plot offs et for pri mar yXA xis	Property:plotO ffs et <code>l\$("#cont ainer").ejC hart({ primaryXAxi s: { plotOffset: 0 }}});</code>	Property:plotOffset <code>let chart: Chart = new Chart({ primaryXAxis: { plotOffset: 0 }});chart.appendTo('#chart');</code>
		min imu m for pri mar yXA xis	Property:range. minimum <code>l\$("#con tainer").ejChart({ primaryXAxi s: { range: { minimum: 10 }}}});</code>	Property:minimum <code>let chart: Chart = new Chart({ primaryXAxis: { minimum: 23 }});chart.appendTo('#chart');</code>

		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');
		RoundingPlaces in primaryXAxis	Property:roundingPlaces\$("#container").ejChart({ primaryXAxis: { roundingPlaces: 3 } });	Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: 'n3' } });chart.appendTo('#chart');
		Scrollbar settings of primaryXAxis	Property:scrollbarSettings\$("#container").ejChart({ primaryXAxis: { scrollbarSettings : { } } });	Not Applicable

		Tick Position in primary XAxis	Property: tickLinesPosition <code>\$("#container").ejChart({ primaryXAxis: { tickLinesPosition: 'Inside' } });</code>	Property: tickPosition <code>let chart: Chart = new Chart({ primaryXAxis: { tickPosition: 'Inside' } }); chart.appendTo('#chart');</code>
		value Type of primary XAxis	Property: valueType <code>\$("#container").ejChart({ primaryXAxis: { valueType: 'DateTime' } });</code>	Property: valueType <code>let chart: Chart = new Chart({ primaryXAxis: { valueType: 'DateTime' } }); chart.appendTo('#chart');</code>
		visible of primary XAxis	Property: visible <code>\$("#container").ejChart({ primaryXAxis: { visible: true } });</code>	Property: visible <code>let chart: Chart = new Chart({ primaryXAxis: { visible: true } }); chart.appendTo('#chart');</code>
		zoom Factor of primary XAxis	Property: zoomFactor <code>\$("#container").ejChart({ primaryXAxis: { zoomFactor: 0.3 } });</code>	Property: zoomFactor <code>let chart: Chart = new Chart({ primaryXAxis: { zoomFactor: 0.3 } }); chart.appendTo('#chart');</code>
		zoom Position of primary XAxis	Property: zoomPosition <code>\$("#container").ejChart({ primaryXAxis: { zoomPosition: 0.3 } });</code>	Property: zoomPosition <code>let chart: Chart = new Chart({ primaryXAxis: { zoomPosition: 0.3 } }); chart.appendTo('#chart');</code>
		label Border	Property: labelBorder <code>\$("#container").ejChart({</code>	Property: border <code>let chart: Chart = new Chart({ primaryXAxis: { border: { color: 'red', width: 3 } } }); chart.appendTo('#chart');</code>

		of primaryXAxis	primaryXAxis: { labelBorder: { color: 'red', width: 2} }));	
		title of primaryXAxis	Property:titleText\$("#container").ejChart({ primaryXAxis: { title: { text: 'Chart title' } } });	Property:titlelet chart: Chart = new Chart({ primaryXAxis: { title: 'Chart title' } });chart.appendTo('#chart');
		StripLine of primaryXAxis	Property:stripLine\$("#container").ejChart({ primaryXAxis: { stripLine: [] } });	Property:stripLineslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] } });chart.appendTo('#chart');
		MultiLevel labels of primaryXAxis	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	Not Applicable	Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }] });chart.appendTo('#chart'); ## primaryYAxis

		axes	Behavior	API in Essential JS 1	API in Essential JS 2
			Alternate grid band	Property:alternateGridBand <pre>\$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' }}}});</pre>	Not applicable
			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.append To('#chart');</pre>
			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.append To('#chart') ;</pre>
			axis elements placed with axis line	Property:showNextToAxisLine <pre>\$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true }});</pre>	Property:placeNextToAxisLine <pre>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLine: ' ' }});chart.append To('#chart') ;</pre>
			axis line style	Property:axisLine.color <pre>\$("#container").ejChart({ primaryYAxis: { axisLine: { color : 'red' } }});</pre>	Property:lineStyle. color <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } }});chart.append</pre>

				<pre>ndTo('#chart') ;</pre> <p>Property:lineStyle.dashArray</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { dashArray: '10, 5' } }});chart.append ndTo('#chart') ;</pre>
			<p>axis line dashArray</p> <pre>Property:axisLine.color\$("#co ntainer").ejChart({ primaryYAxis: { axisLine: { dashArray : '10, 5' } } });</pre>	
			<p>Offset for axis</p> <pre>Property:axisLine.offset\$("#c ontainer").ejChart({ primaryYAxis: { axisLine: { offset : 10 } } });</pre>	<p>Property:plotOffset</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { plotOffset: 10 }});chart.append ndTo('#chart') ;</pre>
			<p>Visible of an axis</p> <pre>Property:axisLine.offset\$("#c ontainer").ejChart({ primaryYAxis: { axisLine: { visible : false } } });</pre>	<p>Property:visible</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { visible: false }});chart.append ndTo('#chart') ;</pre>
			<p>Width of an axis</p> <pre>Property:axisLine.width\$("#c ontainer").ejChart({ primaryYAxis: { axisLine: { width : 2 } }});</pre>	<p>Property:lineStyle.width</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { width: 3 } }});chart.append ndTo('#chart') ;</pre>
			<p>Column index of an axis</p> <pre>Property:columnIndex\$("#co ntainer").ejChart({ primaryYAxis: { columnIndex: 2 } });</pre>	<p>Property:columnIndex</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 }});chart.append</pre>

				ndTo('#chart') ;
			span of an axis to place horizont ally or vertical y	Property:spanlet chart: Chart = new Chart({ primaryYAxis: { span: 2 });chart.appe ndTo('#chart') ;
			Crosshai r label of an axis	Property:crossHair Tooltip.enablelet chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: { enable: true }}});chart.app endTo('#chart');
			Crosshai r label color of an axis	Property:crossHair Tooltip.filllet chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: { fill: 'red' }}});chart.app endTo('#chart');
			Crosshai r label text style	Property:crossHair Tooltip.textStyle1 et chart: Chart = new Chart({ primaryYAxis: { crossHairToolt ip: { textStyle: { } }}});chart.app endTo('#chart');
			Desired interval count	Property:desiredIn tervalslet chart: Chart = new Chart({ primaryYAxis: { desiredInt ervals: { } }}});chart.app endTo('#chart');

				<p>for primary YAxis</p> <pre>primaryYAxis: { desiredIntervals: 4}});</pre>	<pre>new Chart({ primaryYAxis: { desiredIntervals: 4 }});chart.appendTo('#chart') ;</pre>
			Edges primary YAxis	<pre>Property:edgeLabelPlacement\$ ("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' }});</pre>	<pre>Property:edgeLabelPlacementlet chart: Chart = new Chart({ primaryYAxis: { edgeLabelPlacement: 'Shift' }});chart.appendTo('#chart') ;</pre>
			Enables trim for axis labels	<pre>Property:enableTrim\$("#container").ejChart({ primaryYAxis: { enableTrim: true }});</pre>	<pre>Property:enableTrimlet chart: Chart = new Chart({ primaryYAxis: { enableTrim: true }});chart.appendTo('#chart') ;</pre>
			Specifies the interval of the axis according to the zoomed data of the chart	<pre>Property:enableAutoIntervalOnZooming\$ ("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</pre>	<pre>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart') ;</pre>
			Specifies the interval of the axis according to the zoomed	<pre>Property:enableAutoIntervalOnZooming\$ ("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</pre>	<pre>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</pre>

				<p>data of the chart</p> <pre> rvalOnZooming: true });chart.append To('#chart') ; </pre>
			<p>Font style for primary YAxis</p> <pre> Property:font\$("#container ").ejChart({ primaryYAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} }}); </pre>	<pre> Property:titleStyle let chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } }});chart.append To('#chart') ; </pre>
			<p>Indexed for category axis</p> <pre> Property:isIndexed\$("#conta iner").ejChart({ primaryYAxis: { isIndexed: true }}); </pre>	<pre> Property:isIndexed let chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.append To('#chart') ; </pre>
			<p>Interval type for date time axis</p> <pre> Property:intervalType\$("#con tainer").ejChart({ primaryYAxis: { intervalType}: 'Auto' }}); </pre>	<pre> Property:intervalT ypelet chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto }});chart.append To('#chart') ; </pre>
			<p>Inversed axis</p> <pre> Property:isInversed\$("#cont ainer").ejChart({ primaryYAxis: { isInversed: true }}); </pre>	<pre> Property:isInverse dlet chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.append To('#chart') ; </pre>
			<p>Custom label format</p> <pre> Property:labelFormat\$("#con tainer").ejChart({ primaryYAxis: { </pre>	<pre> Property:labelFor matlet chart: Chart = new Chart({ </pre>

				<pre> labelFormat: '{value}K' }}); primaryYAxis: { labelFormat: '{value}K' }});chart.append To('#chart') ; </pre>
			Property:labelIntersectAction <pre> let chart: Chart = new Chart({ primaryYAxis: { labelIntersect Action: 'Trim' }});chart.append To('#chart') ; </pre>	
			Property:labelPosition <pre> let chart: Chart = new Chart({ primaryYAxis: { labelPosition: 'Inside' }});chart.append To('#chart') ; </pre>	
			Property:labelPlacement <pre> let chart: Chart = new Chart({ primaryYAxis: { labelPlacement : 'OnTicks' }});chart.append To('#chart') ; </pre>	
			Axis label alignment <pre> Property:alignment\$("#container").ejChart({ primaryYAxis: { alignment: 'center' }}); </pre>	Not Applicable
			Rotation of axis labels <pre> Property:labelRotation\$("#container").ejChart({ primaryYAxis: { labelRotation: 45 }}); </pre>	Property:labelRotation <pre> let chart: Chart = new Chart({ primaryYAxis: { </pre>

				<pre> labelRotation: 45 });chart.append To('#chart') ; Property:labelRota tionlet chart: Chart = new Chart({ primaryYAxis: { logBase: 10 });chart.append To('#chart') ; </pre>
			Log base value for logarith mic axis	<pre> Property:logBase\$("#contai ner").ejChart({ primaryYAxis: { logBase: 10 }}); </pre>
			Major grid line	<pre> Property:majorGridLines.visible \$("#container").ejChart ({ primaryYAxis: { majorGridLines: { visible: true} }}); </pre>
			Width of MajorGr idLines	<pre> Property:majorGridLines.widthlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines : { width: 2} });chart.append To('#chart') ; </pre>
			Color of MajorGr idLines	<pre> Property:majorGridLines.colorlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines : { color: 'black' } });chart.append To('#chart') ; </pre>
			DashArr ay of MajorGr idLines	<pre> Property:majorGridLines.dashA rray\$("#container").ejCh art({ primaryYAxis: { majorGridLines: { dashArray: 'black' } }); </pre>

				<pre> : { dashArray: 'black' } });chart.appe ndTo('#chart') ; </pre>
			Opacity of major grid line	<pre> Property:majorGridLines.opacit y\$("#container").ejChar t({ primaryYAxis: { majorGridLines: { opacity: true} })); </pre>
			Major Tick line	<pre> Property:majorTickLines.visible \$("#container").ejChart ({ primaryYAxis: { majorTickLines: { visible: true} })); </pre>
			Width of Major Tick Lines	<pre> Property:majorTickLines.width let chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { width: 2} });chart.appe ndTo('#chart') ; </pre>
			Height of Major Tick Lines	<pre> Property:majorTickLines.height let chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { height: 2} });chart.appe ndTo('#chart') ; </pre>
			Color of Major Tick Lines	<pre> Property:majorTickLines.color let chart: Chart = new Chart({ primaryYAxis: { majorTickLines : { color: 'black' } });chart.appe ndTo('#chart') ; </pre>

				<div>Property:majorTickLines.opacity</div> <div>Opacity of major Tick line</div> <div>y\$("#container").ejChart({ primaryYAxis: { majorTickLines: { opacity: true} }});</div>	Not Applicable
			<div>Property:maximumLabels</div> <div>maximum labels of primary YAxis</div> <div>Property:maximumLabels\$("#container").ejChart({ primaryYAxis: { maximumLabels: 5 }});</div>	<div>Property:maximumLabels</div> <div>let chart: Chart = new Chart({ primaryYAxis: { maximumLabels: 4 }});chart.appendTo('#chart');</div>	
			<div>Property:maximumLabelWidth</div> <div>maximum labels width of primary YAxis to trim</div> <div>Property:maximumLabelWidth\$("#container").ejChart({ primaryYAxis: { maximumLabelWidth: 40 }});</div>	<div>Property:maximumLabelWidth</div> <div>let chart: Chart = new Chart({ primaryYAxis: { maximumLabelWidth: 4 }});chart.appendTo('#chart');</div>	
			<div>Property:minorGridLines.visible</div> <div>minor grid line</div> <div>e\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true} }});</div>	Not Applicable	
			<div>Property:minorGridLines.width</div> <div>Width of minorGridLines</div> <div>Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2} }});</div>	<div>Property:minorGridLines</div> <div>let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { width: 2} }});chart.appendTo('#chart');</div>	
			<div>Property:minorGridLines.color</div> <div>Color of minorGridLines</div> <div>Property:minorGridLines.color\$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } }});</div>	<div>Property:minorGridLines</div> <div>let chart: Chart = new Chart({ primaryYAxis: {</div>	

				<pre> minorGridLines : { color: 'black' } });chart.appendTo('#chart') ; Property:minorGridLines.dashArray let chart: Chart = new Chart({ primaryYAxis: { minorGridLines : { dashArray: 'black' } });chart.appendTo('#chart') ; </pre>
			<p>DashArray of minorGridLines</p>	<pre> Property:minorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } }}); </pre>
			<p>Opacity of minor grid line</p>	<pre> Property:minorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { minorGridLines: { opacity: true} }}); </pre> <p>Not Applicable</p>
			<p>minor Tick line</p>	<pre> Property:minorTickLines.visible\$("#container").ejChart({ primaryYAxis: { minorTickLines: { visible: true} }}); </pre> <p>Not Applicable</p>
			<p>Width of minorTickLines</p>	<pre> Property:minorTickLines.width\$("#container").ejChart({ primaryYAxis: { minorTickLines: { width: 2} }}); </pre> <p>Property:minorTickLines.widthlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines : { width: 2} });chart.appendTo('#chart') ;</p>
			<p>Height of minorTickLines</p>	<pre> Property:minorTickLines.size\$("#container").ejChart({ primaryYAxis: { minorTickLines: { size: 2} }}); </pre> <p>Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines : { height: 2} });chart.appendTo('#chart')</p>

				<pre>ndTo('#chart') ;</pre>	
				<pre>Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { color: 'black' } });chart.appendTo('#chart') ;</pre>	
			Color of minorTickLines	<pre>Property:minorTickLines.color\$ ("#container").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } }});</pre>	
			Opacity of minor Tick line	<pre>Property:minorTickLines.opacity\$ ("#container").ejChart({ primaryYAxis: { minorTickLines: { opacity: true } }});</pre>	Not Applicable
			Minor ticks per interval of primary YAxis	<pre>Property:minorTicksPerInterval\$ ("#container").ejChart({ primaryYAxis: { minorTicksPerInterval: 4 }});</pre>	
			name of the primary YAxis	<pre>Property:name\$ ("#container").ejChart({ primaryYAxis: { name: 'primaryYAxis' }});</pre>	
			Orientat ion of primary YAxis	<pre>Property:orientation\$ ("#container").ejChart({ primaryYAxis: { orientation: 'Vertical' }});</pre>	Not Applicable
			Plot offset for	<pre>Property:plotOffset\$ ("#container").ejChart({ primaryYAxis: { plotOffset: 0 }});</pre>	Property:plotOffsetlet chart: Chart = new Chart({ primaryYAxis:

				<pre> primary YAxis { 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: { rangePadding: 'None' });chart.appe ndTo('#chart') ; Property:labelFor matlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' </pre>
				<pre> Property:range.minimum\$("# container").ejChart({ primaryYAxis: { range: { minimum: 10 }}}); Property:range.maximum\$("# container").ejChart({ primaryYAxis: { range: { maximum: 10 }}}); Property:range.interval\$("#co ntainer").ejChart({ primaryYAxis: { range: { interval: 1 }}}); Property:rangePadding\$("#co ntainer").ejChart({ primaryYAxis: { rangePadding: 'None' }}}); Property:roundingPlaces \$("#container").ejChart ({ primaryYAxis: { roundingPlaces: 3 }}); </pre>

				<pre> });chart.appendTo('#chart') ; </pre>
			ScrollBar settings of primary YAxis	<pre> Property:scrollbarSettings \$("#container").ejChart(({ primaryYAxis: { scrollbarSettings : { } }}}); </pre> <p>Not Applicable</p>
			TickPosition in primary YAxis	<pre> Property:tickPosition let chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' }});chart.appendTo('#chart') ; </pre>
			valueType of primary YAxis	<pre> Property:valueType let chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' }});chart.appendTo('#chart') ; </pre>
			visible of primary YAxis	<pre> Property:visible let chart: Chart = new Chart({ primaryYAxis: { visible: true }});chart.appendTo('#chart') ; </pre>
			zoomFactor of primary YAxis	<pre> Property:zoomFactor let chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 }});chart.appendTo('#chart') ; </pre>

				<p>Property:zoomPosition</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 }});chart.append To('#chart') ;</pre>
				<p>Property:labelBorder</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } }});chart.append To('#chart') ;</pre>
				<p>Property:title</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { title: { text: 'Chart title' } }});chart.append To('#chart') ;</pre>
				<p>Property:stripLine</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.append To('#chart') ;</pre>
				<p>Property:multiLevelLabels</p> <pre>let chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.append To('#chart') ;</pre>
				<p>zoomPosition of primary YAxis</p> <pre>Property:zoomPosition\$("#container").ejChart({ primaryYAxis: { zoomPosition: 0.3 }});</pre>
				<p>labelBorder of primary YAxis</p> <pre>Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2} }});</pre>
				<p>title of primary YAxis</p> <pre>Property:title.text\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } }});</pre>
				<p>StripLine of primary YAxis</p> <pre>Property:stripLine\$("#container").ejChart({ primaryYAxis: { stripLine: [] }});</pre>
				<p>Multilevel labels of primary YAxis</p> <pre>Property:multiLevelLabels\$("#container").ejChart({ primaryYAxis: { multiLevelLabels: [] }});</pre>

				<p>Property:skeleton let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]]});chart.app endTo('#chart');</p> <p>Property:skeleton Type let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]]});chart.app endTo('#chart');</p> <p>skeleton for an axes Not Applicable</p> <p>skeleton type for an axes Not Applicable</p> <p>## Axes</p> <p>Behavio ur API in Essential JS 1 API in Essential JS 2</p> <p>Property:altern ateGridBand Alte rnat e grid ban d \$("#contain er").ejChar t({ axes: [{ { alternat eGr idBand: { even: { fill: 'red' }]]}});</p> <p>Axis line cros s valu e Property:cross esAt \$("#cont ainer").ejC hart({ axes: [{ crossesAt: 0 }]]});</p> <p>axis na me wit h whi ch Property:cross esInAxis \$("#c ontainer"). ejChart({ axes: [{ crossesInAx Property:crossesAt let chart: Chart = new Chart({ axes: [{ crossesAt: 4}]]});chart.appendTo('#chart');</p> <p>Property:crossesInAxis let chart: Chart = new Chart({ axes: [{ crossesInAxis: '' }]]});chart.appendTo('#chart') ;</p>
--	--	--	--	--

			<pre> the is: '' axis }}}); line has to be crossed axis ele Property:show me NextToAxisLine Property:placeNextToAxisLinelet nts \$("#contain chart: Chart = new Chart({ er").ejChar axes: [{ plac t({ axes: [placeNextToAxisLine: '' ed { }}});chart.appendTo('#chart') wit { showNextToA ; h xisLine : axis true }}}); line Property:axisLi ne.color\$("#c axis ontainer"). Property:lineStyle.colorlet chart: line ejChart({ Chart = new Chart({ axes: [{ styl axes: [{ lineStyle: { color: 'black' } e axisLine: { }}});chart.appendTo('#chart') color : ; 'red' } }}}); Property:axisLi ne.color\$("#c axis ontainer"). Property:lineStyle.dashArraylet line ejChart({ chart: Chart = new Chart({ das axes: [{ lineStyle: { hAr axisLine: { }}});chart.appendTo('#chart') ray dashArray : ; '10, 5' } }}}); Property:axisLi ne.offset\$("#c Offs ontainer"). Property:plotOffsetlet chart: et ejChart({ Chart = new Chart({ axes: [{ for axes: [{ plotOffset: 10 axis axisLine: { }}});chart.appendTo('#chart') offset : 10 ; } }}}); Visi Property:axisLi Property:visiblelet chart: Chart ble ne.offset\$("#c = new Chart({ axes: [{ of ontainer"). visible: false </pre>
--	--	--	--

				<pre> an ejChart({ }));chart.appendTo('#chart') axis axes: [{ ; axisLine: { visible : false } }]); Property:axisLi Width:width\$("#c th ontainer").c of ejChart({ Property:lineStyle.width an axes: [{ let chart: axis axisLine: { Chart = new Chart({ axes: [{ width : 2 } lineStyle: { width: 3 } }]);chart.appendTo('#chart') }]); ; Col um Property:column n nIndex\$("#co ind ntainer").e ex jChart({ of axes: [{ an columnIndex ; axis : 2 }]); spa n of an axis to Property:column plac nSpan\$("#con e tainer").ej hori Chart({ zon axes: [{ tally columnIndex or : 2 }]); vert icall y Property:cross Cro HairLabel.visibl ssh e\$("#contai air ner").ejCha labe rt({ axes: l of [{ an crossHairLa axis bel: { visible: true }]); </pre>
--	--	--	--	---

				<p>Crosshair label color of an axis</p> <p>Property:crossHairTooltip.fill chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' }}}]);chart.appendTo('#chart');</p> <p>Not applicable</p>
				<p>Crosshair label text style</p> <p>Property:crossHairTooltip.textStyle let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } }}}]);chart.appendTo('#chart');</p> <p>Not applicable</p>
				<p>Desired intervals</p> <p>Property:desiredIntervals let chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 }]);chart.appendTo('#chart') ;</p> <p>desiredIntervals: 4</p>
				<p>Edge label placement</p> <p>Property:edgeLabelPlacement let chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' }]);chart.appendTo('#chart') ;</p> <p>edgeLabelPlacement: 'none'</p>
				<p>Enable trim for axis labels</p> <p>Property:enableTrim let chart: Chart = new Chart({ axes: [{ enableTrim: true }]);chart.appendTo('#chart') ;</p> <p>enableTrim: true</p>

			<p>Specify the interval of the axis according to the zoomed data of the chart</p> <pre> Property:enableAutoIntervalOnZooming\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] }); </pre>
			<p>Specify the interval of the axis according to the zoomed data of the chart</p> <pre> Property:enableAutoIntervalOnZooming\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] }); </pre>

				<pre> Property:font\$ ("#container").ejChart ({ axes: [{ font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: 'bold', opacity: 0.5, size: 12} }]}); Property:titleStylelet chart: Chart = new Chart({ axes: [{ titleStyle: { } }]});chart.appendTo('#chart') ; Property:isIndexedlet chart: Chart = new Chart({ axes: [{ isIndexed: true }]});chart.appendTo('#chart') ; Property:intervalTypelet chart: Chart = new Chart({ axes: [{ intervalType: 'Auto' }]});chart.appendTo('#chart') ; Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true }]});chart.appendTo('#chart') ; Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: '{value}K' }]});chart.appendTo('#chart') ; </pre>
--	--	--	--	---

				<p>Property:labelIntersectionAction</p> <pre> labelIntersectionAction\$("#container").ejChart({ axes: [{ labelIntersectionAction: 'Trim' }] });chart.appendTo('#chart') </pre> <p>Property:labelPosition</p> <pre> labelPosition\$("#container").ejChart({ axes: [{ labelPosition: 'Inside' }] });chart.appendTo('#chart') </pre> <p>Property:labelPlacement</p> <pre> labelPlacement\$("#container").ejChart({ axes: [{ labelPlacement: 'OnTicks' }] });chart.appendTo('#chart') </pre> <p>Property:align</p> <pre> AxisLabelAlignment\$("#container").ejChart({ axes: [{ alignment: 'center' }] }); </pre> <p>Property:labelRotation</p> <pre> labelRotation\$("#container").ejChart({ axes: [{ labelRotation: 45 }] });chart.appendTo('#chart') </pre> <p>Property:logBase</p> <pre> logBase\$("#container").ejChart({ axes: </pre>
				<p>Property:labelIntersectionAction</p> <pre> let chart: Chart = new Chart({ axes: [{ labelIntersectionAction: 'Trim' }] });chart.appendTo('#chart') </pre> <p>Property:labelPosition</p> <pre> let chart: Chart = new Chart({ axes: [{ labelPosition: 'Inside' }]});chart.appendTo('#chart') </pre> <p>Property:labelPlacement</p> <pre> let chart: Chart = new Chart({ axes: [{ labelPlacement: 'OnTicks' }]});chart.appendTo('#chart') </pre> <p>Not Applicable</p> <p>Property:labelRotation</p> <pre> let chart: Chart = new Chart({ axes: [{ labelRotation: 45 }]});chart.appendTo('#chart') </pre> <p>Property:labelRotation</p> <pre> let chart: Chart = new Chart({ axes: [{ logBase: 10 </pre>

				<pre>e [{ for logBase: 10 ; loga }}});chart.appendTo('#chart') rith mic axis</pre> <p>Property:major GridLines.visibility</p> <pre>Maj e\$("#contai or ner").ejCha grid rt({ axes: line [{ Not Applicable majorGridLi nes: { visible: true} }]]);</pre> <p>Property:major GridLines.width</p> <pre>Wid GridLines.width th \$("#contain of er").ejChar Maj t({ axes: [orG { ridL majorGridLi ines nes: { width: 2} }]]);</pre> <p>Property:major GridLines.color</p> <pre>Col GridLines.color or \$("#contain of er").ejChar Maj t({ axes: [orG { ridL majorGridLi ines nes: { color: 'black' } }]]);</pre> <p>Property:major GridLines.dash</p> <pre>Das GridLines.dash hAr Array\$("#con ray tainer").ej of Chart({ Maj axes: [{ orG majorGridLi ridL nes: { dashArray: 'black' } }]]);</pre> <p>Property:majorGridLines.widthlet chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2} }]]);chart.appendTo('#chart') ;</p> <p>Property:majorGridLines.colorlet chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }]]);chart.appendTo('#chart') ;</p> <p>Property:majorGridLines.dashArrayle t chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' } }]]);chart.appendTo('#chart') ;</p>
--	--	--	--	---

				<p>Property:major Opa GridLines.opaci city ty\$ ("#contai of ner").ejCha maj rt({ axes: Not Applicable or [{ grid majorGridLi line nes: { opacity: true} }]]);</p> <p>Property:major TickLines.visibl Maj e\$ ("#contai or ner").ejCha Tick rt({ axes: Not Applicable line [{ majorTickLi nes: { visible: true} }]]);</p> <p>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} }]]);</p> <p>Property:major Hei TickLines.size\$ ght ("#containe Property:majorTickLines.heightlet of r").ejChart chart: Chart = new Chart({ Maj ({ axes: [axes: [{ majorTickLines: { orTi { height: 2} ckLi majorTickLi }]]);chart.appendTo('#chart') nes nes: { ; size: 2} }]]);</p> <p>Property:major Col TickLines.color Property:majorTickLines.colorlet or \$ ("#contain chart: Chart = new Chart({ of er").ejChar axes: [{ majorTickLines: { Maj t({ axes: [color: 'black' } orTi { }]]);chart.appendTo('#chart') ckLi majorTickLi ; nes nes: { color:</p>
--	--	--	--	--

				<pre> 'black' } }]); Property:major Opa TickLines.opaci city ty\$("#contai of ner").ejCha maj rt({ axes: Not Applicable or [{ majorTickLi Tick nes: { line opacity: true} }]); max imu m labe Property:maxi ls of mumLabels\$("# Property:maximumLabelslet chart: pri) .ejChart({ Chart = new Chart({ axes: [{ mar axes: [{ maximumLabels: 4 yYA ls: 5 }]); xis max imu m labe Property:maxi ls mumLabelWidt Property:maximumLabelWidthlet wid h\$("#contai chart: Chart = new Chart({ th ner").ejCha axes: [{ maximumLabelWidth: of rt({ axes: 4 pri [{ mar maximumLabe yYA lWidth: 40 xis to trim Property:minor GridLines.visibl min e\$("#contai or ner").ejCha grid rt({ axes: Not Applicable line [{ minorGridLi nes: { visible: true} }]); </pre>
--	--	--	--	---

				<p>Property:minor Width GridLines.width \$("#contain er").ejChar t({ axes: [axes: [{ minorGridLines: { width: 2} minorGridLi nes: { width: 2} }] });</p> <p>Property:minor GridLines.color \$("#contain er").ejChar t({ axes: [axes: [{ minorGridLines: { color: 'black' }] }];</p> <p>Property:minor GridLines.dash Array\$("#con tainer").ej Chart({ axes: [{ minorGridLi nes: { dashArray: 'black' } }] });</p> <p>Property:minor GridLines.opaci ty\$("#contai ner").ejCha rt({ axes: [{ minorGridLi nes: { opacity: true } }] });</p> <p>Property:minor TickLines.visibl e\$("#contai ner").ejCha rt({ axes: [{ minorTickLi nes: {</p>
				<p>Property:minorGridLines.widthlet chart: Chart = new Chart({ axes: [{ minorGridLines: { width: 2} }] });chart.appendTo('#chart') ;</p> <p>Property:minorGridLines.colorlet chart: Chart = new Chart({ axes: [{ minorGridLines: { color: 'black' } }] });chart.appendTo('#chart') ;</p> <p>Property:minorGridLines.dashArrayle t chart: Chart = new Chart({ axes: [{ minorGridLines: { dashArray: 'black' } }] });chart.appendTo('#chart') ;</p> <p>Not Applicable</p> <p>Not Applicable</p>

				<pre> visible: true} }]]); Property:minor Wid TickLines.width th \$("#contain of er").ejChar min t({ axes: [orTi minorTickLi ckLi nes: { nes width: 2} }]]); Property:minor Hei TickLines.size\$ ght (\$("#containe of r").ejChart min ({ axes: [orTi minorTickLi ckLi nes: { nes size: 2} }]]); Property:minor Col TickLines.color or \$("#contain of er").ejChar min t({ axes: [orTi minorTickLi ckLi nes: { nes color: 'black' } }]]); Property:minor Opa TickLines.opaci city ty\$("#contai of ner").ejCha min rt({ axes: or [{ Tick minorTickLi line nes: { opacity: true} }]]); Min Property:minor or TicksPerInterva tick l\$("#contain s er").ejChar per t({ axes: [inte { </pre>
				<pre> Property:minorTickLines.widthlet chart: Chart = new Chart({ axes: [{ minorTickLines: { width: 2} }]]);chart.appendTo('#chart') ; Property:minorTickLines.heightlet chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2} }]]);chart.appendTo('#chart') ; Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }]]);chart.appendTo('#chart') ; Not Applicable Property:minorTickLines.colorlet chart: Chart = new Chart({ axes: [{ minorTicksPerInterval: 4 }]]);chart.appendTo('#chart') ; </pre>

			<pre> rval minorTicksP of erInterval: pri 4 }]); mar yYA 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' }]); xis Orie Property:orient ntat ation\$("#con ion tainer").ej of Chart({ pri axes: [{ Not Applicable mar orientation yYA : xis 'Vertical' }]); Plot offs Property:plotO et fffset\$("#cont Property:plotOffsetlet chart: for ainer").ejC Chart = new Chart({ axes: [{ pri hart({ plotOffset: 0 mar axes: [{ }]);chart.appendTo('#chart') yYA plotOffset: ; xis 0 }]); xis min imu Property:range. m minimum\$("## Property:minimumlet chart: for container") Chart = new Chart({ axes: [{ pri .ejChart({ minimum: 23 mar axes: [{ }]);chart.appendTo('#chart') yYA range: { ; xis minimum: 10 }]); }]); max Property:range. Property:maximumlet chart: imu maximum\$("## Chart = new Chart({ axes: [{ m container") maximum: 23 for .ejChart({ }]);chart.appendTo('#chart') pri axes: [{ ; range: { </pre>
--	--	--	--

			<pre> mar maximum: 10 yYA }}}}); xis inte Property:range. rval interval\$("#co for ntainer").e = new Chart({ axes: [{ pri jChart({ interval: 2 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 }}}}); Scr ollB Property:scroll ar barSettings sett \$("#contain ings er").ejChar of t({ axes: [pri { mar scrollbarSe yYA ttings : { xis } }}}}); Tick Property:tickLi Posi nesPosition\$("# tion #container" in).ejChart({ axes: [{ </pre>
--	--	--	---

			<pre> pri tickLinesPo mar sition: yYA 'Inside' xis }}}}); valu Property:value eTy Type\$("#cont pe ainer").ejC Property:valueTypelet chart: of hart({ Chart = new Chart({ axes: [{ pri axes: [{ valueType: 'DateTime' mar valueType: }]);chart.appendTo('#chart') yYA 'DateTime' ; xis }}}}); visi ble Property:visible of \$("#contain Property:visiblelet chart: Chart er").ejChar = new Chart({ axes: [{ pri t({ axes: [visible: true mar { visible: }]);chart.appendTo('#chart') yYA true }]); ; xis zoo mFa Property:zoom ctor Factor\$("#con Property:zoomFactorlet chart: of tainer").ej Chart = new Chart({ axes: [{ pri Chart({ zoomFactor: 0.3 mar axes: [{ }]);chart.appendTo('#chart') yYA zoomFactor: ; xis 0.3 }]); zoo mP Property:zoom ositi Position\$("#c Property:zoomPositionlet chart: on ontainer"). Chart = new Chart({ axes: [{ of ejChart({ zoomPosition: 0.3 pri axes: [{ }]);chart.appendTo('#chart') mar zoomPositio n: 0.3 ; yYA }]); xis labe Property:labelB lBor order\$("#con Property:borderlet chart: Chart der tainer").ej = new Chart({ axes: [{ of Chart({ border: { color: 'red', pri axes: [{ width: 3 } mar labelBorder }]);chart.appendTo('#chart') : { color: ; mar 'red', </pre>
--	--	--	---

			<pre> yYA width: 2} xis }}}}); Property:title.t title ext\$("#conta of iner").ejCh pri art({ axes: mar [{ title: 'Chart title' yYA { text: xis 'Chart title' } }}}); Stri Property:stripli ne\$("#conta e of iner").ejCh pri art({ axes: mar [{ yYA stripLine: xis []]}}}); Mul Property:multiL tilev evellLabels\$("# el #container" labe).ejChart({ ls of axes: [{ axe multiLevelL s abels: [] }}}); skel eto n for Not Applicable an axe s skel eto n typ e Not Applicable for an axe s Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]);chart.appendTo('#chart') ; Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]);chart.appendTo('#chart') ; ## Rows Beh avio API in Essential API in Essential ur JS 1 JS 2 row Property:rowD Property:rows1 s in efinitions\$("## et chart: </pre>
--	--	--	---

				<pre> cha chart").ejC Chart = new rt hart({ Chart({ rowDefiniti rows: ons: [];}); [];});chart .appendTo('# #chart'); Property:unit \$("#contain er").ejChar t({ unit rowDefiniti Not Applicable ons :[{unit : "percentage "}]}); Property:height let chart: height\$("#cha Chart = new of rt").ejChar Chart({ row t({ rows: [{ s in rowDefiniti height: cha ons: [{ '300'}] }); rt rowHeight: chart.appen dTo('#chart '); Property:lineCo Property:borde lor, rlet chart: lineWidth\$("# Chart = new chart").ejC Chart({ Line hart({ rows: [{ cust rowDefiniti height: omi ons: [{ '300', zati rowHeight: border: { on '50%', width: 2, lineColor: color: 'brown', 'brown'}}]; lineWidth: });chart.ap 2});}); pendTo('#ch art'); ## Series Beha API in Essential JS API in viou 1 Essential JS r 2 Property:bearFillC Property:be arFillC arFillColor1 olor\$("#chart" et chart:).ejChart({ Chart = series: new [{bearFillCol new </pre>
--	--	--	--	--

				<pre> or: 'red' });}); </pre>	<pre> Chart({ series: [{bearFil lColor: 'red' }];});cha rt.append To('#char t'); </pre>
				<pre> Property:ro wslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2, dashArray: '10, 5' } }];}); </pre>	<pre> Property:ro wslet chart: Chart = new Chart({ series: [{ border: { color: 'red', width: 2} }];});cha rt.append To('#char t'); </pre>
				<pre> Property:boxPlot Mode\$("#chart ").ejChart({ series: [{ boxPlotMode: 'inclusive' }];}); </pre>	<pre> Property:ro wslet chart: Chart = new Chart({ series: [{ boxPlotMo de: 'Inclusiv e' }];});cha rt.append To('#char t'); </pre>
				<pre> Property:bubbleO ptions.minRadius \$("#chart").e jChart({ series: [{ bubbleOptions : { minRadius: 2} }];}); </pre>	<pre> Property:mi nRadiuslet chart: Chart = new Chart({ series: [{ minRadius : 2 }];}); </pre>

				<pre> series]});chart.append To('#chart'); Property:maximumRadius t chart: Chart = new Chart({ series: [{ maxRadius : 2 }]);chart.append To('#chart'); Property:bubbleOptions t chart: Chart = new Chart({ series: [{ maxRadius : 2 }]);chart.append To('#chart'); Property:bullFillColor t chart: Chart = new Chart({ series: [{bullFillColor: 'red' }]);chart.append To('#chart'); Property:cardinalSplineTension t chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]);chart.append To('#chart'); Property:columnWidth t chart: Chart = new Chart({ series: [{ columnWidth : 100 }]);chart.append To('#chart'); </pre>
--	--	--	--	--

				Width for rectangle angle series	<pre>").ejChart({ series: [{ columnWidth: 0.5 }]);}); let chart: Chart = new Chart({ series: [columnWid th: 0.5 }]);});cha rt.append To('#char t');</pre>
				Column spacing for rectangle angle series	<pre>Property:columnS glet chart: Chart = new Chart({ series: [columnSpa cing: 0.5 }]);});cha rt.append To('#char t');</pre>
				Top left radius for rectangle angle series	<pre>Property:cornerR adius.topLeft\$ (" #chart").ejCh art({ series: [topLeft: 0 }]);}); let chart: Chart = new Chart({ series: [topLeft: 0 }]);});cha rt.append To('#char t');</pre>
				top right radius for rectangle angle	<pre>Property:cornerR adius.topRight\$ ("#chart").ejC hart({ series: [topRight: 0 }]);}); Property:co rnerRadius. topRightle t chart: Chart = new Chart({ series:</pre>

				<pre> e serie s [{ 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 }];});cha rt.append To('#char t'); Property:da shArraylet chart: Chart = new Chart({ series: [dashArray : '10, 5' }];});cha rt.append </pre>
--	--	--	--	---

				<pre> To('#chart'); Property:dashboardArraylet chart: Chart = new Data Source\$("#chart").ejChart({ series: [dataSource: [[]]});}); chart.append To('#chart'); Property:drawTypelet chart: Chart = new Draw type\$("#chart").ejChart({ series: [drawType: 'Line']});}); chart.append To('#chart'); Property:emptyPoints intSettings.visible \$("#chart").ejChart({ series: [emptyPointSettings: { visible: false } }]);}); Property:emptyPointSettings.displayMode \$("#chart").ejChart({ series: [displayMode: 'gap']});}); </pre>
--	--	--	--	---

				<pre> displayMode: '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 }];});chart.append To('#chart'); Property:animation.enable let chart: Chart = new Chart({ series: [animation: { enable: false } } </pre>
--	--	--	--	---

				<pre> });});chart.append To('#chart'); Property:animation.duration let chart: chart: = new Chart({ series: [animation : { duration: 1000 }]);});chart.append To('#chart'); Property:animation.duration let chart: Chart = new Chart({ series: [animation : { delay: 100 }]);});chart.append To('#chart'); Drag Property:dragSettings\$("#chart") settings\$.ejChart({ series: [{ dragSettings: { mode: 'X' } }]);}); Error Property:errorBarSettings\$("#chart").ejChart(settings { series: [{ errorBarSettings for new Chart({ </pre>
--	--	--	--	---

				<pre> series: { series: [{ errorBarSettings: { isClosed: true } }] }; </pre>
				<pre> Property:isClosed let chart = new Chart({ series: [{ isClosed: true }] }); </pre>
				<pre> Property:isStacking let chart = new Chart({ series: [{ isStacking: true }] }); </pre>
				<pre> Property:lineCap let chart = new Chart({ series: [{ lineCap: 'butt' }] }); </pre>
				<pre> Property:lineJoin let chart = new Chart({ series: [{ lineJoin: 'round' }] }); </pre>
				<pre> Property:errorBarSettings let chart = new Chart({ series: [{ opacity: 0.7 }] }); </pre>
				<pre> Property:opacity let chart = new Chart({ series: [{ opacity: 0.7 }] }); </pre>

				<p>Property:outLierSettings</p> <pre> OutlierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20}} }];}); </pre> <p>Not Applicable</p>
				<p>Property:palette\$</p> <pre> Palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }];}); </pre> <p>Property:pointColorMappinglet</p> <pre> chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart'); </pre>
				<p>Property:positiveFill</p> <pre> PositiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }];}); </pre> <p>Property:pointColorMappinglet</p> <pre> chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart'); </pre>
				<p>Property:showMedian</p> <pre> ShowMedian\$("#chart").ejChart({ series: [{ showMedian: true }];}); </pre> <p>Property:pointColorMappinglet</p> <pre> chart: Chart = new Chart({ </pre>

				<p>and whis ker serie s</p> <p>To grou p the serie s of stac king colle ction</p> <p>Spec ifies the type of the serie s to rend er in char t.</p> <p>Defi nes the visibi lity of the serie s.</p>	<pre> series: [{ showMean: false }];});cha rt.append To('#char t); Property:st ackingGrou plet chart: Chart = new Chart({ series: [{ stackingG roup: 'group' }];});cha rt.append To('#char t); Property:ty pelet chart: Chart = new Chart({ series: [{ type: 'Line' }];});cha rt.append To('#char t); Property:vis iblelet chart: Chart = new Chart({ series: [{ visible: true }];});cha rt.append </pre>
--	--	--	--	---	--

				<pre> To('#chart'); Property:toggleVisibility let chart: Chart = new Chart({ legendSettings: [{ toggleVisibility: true }] });chart.append To('#chart'); Property:splineType let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }] });chart.append To('#chart'); Property:xAxisName let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }] });chart.append To('#chart'); </pre>
--	--	--	--	--

				<p>d with this serie s. Add an axis insta nce with this nam e to axes colle ction .</p> <p>Nam e of the prop erty in the data sour ce that cont ains x valu e for the serie s.</p> <p>Spec ifies the nam e of the y- axis</p>	<p>Property:xName\$ ("#chart").ej Chart({ series: [{ xName : 'x' }]);});</p> <p>Property:yAxisNa me\$("#chart") .ejChart({ series: [{ yAxisName : 'secondaryYAx is' }]);});</p>	<p>Property:xN amelet chart: Chart = new Chart({ series: [{ xName: 'x' }]);});cha rt.append To('#char t');</p> <p>Property:yA xisNamele t chart: Chart = new Chart({ series: [{ yAxisName</p>
--	--	--	--	---	---	---

				<div><div>that</div><div>has</div><div>to</div><div>be</div><div>asso</div><div>ciate</div><div>d</div><div>with</div><div>this</div><div>serie</div><div>s.</div><div>Add</div><div>an</div><div>axis</div><div>insta</div><div>nce</div><div>with</div><div>this</div><div>nam</div><div>e to</div><div>axes</div><div>colle</div><div>ction</div><div>.</div><div>Nam</div><div>e of</div><div>the</div><div>prop</div><div>erty</div><div>in</div><div>the</div><div>data</div><div>sour</div><div>ce</div><div>that</div><div>cont</div><div>ains</div><div>y</div><div>valu</div><div>e for</div><div>the</div><div>serie</div><div>s.</div><div>Nam</div><div>e of</div></div> <div><div>: 'secondar yYAxis' }];});cha rt.append To('#char t');</div><div><div>Property:yN amelet chart: Chart = new Chart({ series: [{ yName: 'y' }];});cha rt.append To('#char t');</div><div><div>Property:yName\$ ("#chart").ej Chart({ series: [{ yName : 'y' }];});</div><div><div>Property:high\$ (" #chart").ejCh ghlet</div></div></div></div></div>
--	--	--	--	---

				<p>the art({ series: chart: [{ high : 'y' Chart = new Chart({ series: [{ high: 'y' }]);cha rt.append To('#char t);</p> <p>prop</p> <p>erty</p> <p>in</p> <p>the</p> <p>data</p> <p>sour</p> <p>ce</p> <p>that</p> <p>cont</p> <p>ains</p> <p>high</p> <p>valu</p> <p>e for</p> <p>the</p> <p>serie</p> <p>s.</p> <p>Nam</p> <p>e of</p> <p>the</p> <p>prop</p> <p>erty</p> <p>in</p> <p>the</p> <p>data</p> <p>sour</p> <p>ce</p> <p>that</p> <p>cont</p> <p>ains</p> <p>low</p> <p>valu</p> <p>e for</p> <p>the</p> <p>serie</p> <p>s.</p> <p>Nam</p> <p>e of</p> <p>the</p> <p>prop</p> <p>erty</p> <p>in</p> <p>the</p> <p>data</p> <p>sour</p>
				<p>Property:low\$("# new chart").ejCha rt({ series: [{ low : 'y' }]);cha rt.append To('#char t);</p> <p>Property:low\$("# new chart").ejCha rt({ series: [{ low : 'y' }]);cha rt.append To('#char t);</p> <p>Property:close\$("#chart").ejC hart({ series: [{ close : 'y' }]);</p> <p>Property:close\$("#chart").ejC hart({ series: [{ close : 'y' }]);</p>

				<pre> ce that cont ains close valu e for the serie s. Nam e of the prop erty in the data sour ce that cont ains open valu e for the serie s. Property:open\$ ("#chart").ejC hart({ series: [{ open : 'y' }];}); Property:open\$ ("#chart").ejC hart({ series: [{ open : 'y' }];});cha rt.append To('#char t'); Property:tr endLinesle t chart: Chart = new Chart({ series: [{ open: 'y' }];});cha rt.append To('#char t'); Property:trendLin es\$ ("#chart"). ejChart({ series: [{ trendLines : [{}] }];}); Property:highlight Settings\$ ("cha rt").ejChart(</pre>
				<pre> rt.append To('#char t'); Property:op enlet chart: Chart = new Chart({ series: [{ open: 'y' }];});cha rt.append To('#char t'); Property:tr endLinesle t chart: Chart = new Chart({ series: [{ trendLine s : [{}]}];});cha rt.append To('#char t'); Not applicable. </pre>

				<pre> cust { series: [{ omiz highlightSett ing ings : {} the }];}); the appe aran ce of the serie s or data poin t whil e highl ighti ng. Opti ons for cust omiz ing the Property:selection the Settings appe \$("#chart").e aran jChart({ Not ce of series: [{ applicable. the selectionSett serie ings : {} s/da }];}); ta poin t on selec tion. ## marker visib Property:visible\$(ility "#chart").ejCh siblelet of art({ series: chart: mar [{ marker: { Chart = ker visible: true new }]}); Chart({ series: [{ </pre>
--	--	--	--	---

				<pre> marker: { visible: false } }];});ch art.appe ndTo('#c hart); Property:vi siblelet chart: Chart = new Property:fill\$("#c hart").ejChart for ({ series: [{ marker: { fill : 'red' } }];}); </pre>
				<pre> Property:vi siblelet chart: Chart = new Property:opacity\$ ("#chart").ejCh art({ series: [{ marker: { opacity : 0.5 } }];}); </pre>
				<pre> Property:shape\$("# chart").ejCha rt({ series: [{ marker: { shape : 'Circle' } }];}); </pre>

				<pre> e' }]});ch art.appe ndTo('#c hart); Property:im ageUrllet t chart: Chart = new Property:imageUrl \$("#chart").ej Chart({ series: [{ marker: { imageUrl : '' } }]);});]});ch art.appe ndTo('#c hart); Property:s hapelet chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } 2, } }]);});]});ch art.appe ndTo('#c hart); Property:heightlet t\$("#chart").ej jChart({ series: [{ marker: { size: { height: 30} } } }]);});]});ch art.appe ndTo('#c hart); </pre>
--	--	--	--	--

				<pre> height: 25 } });});ch art.appe ndTo('#c hart); Property:w idthlet chart: Chart = new Property:size.widt h\$("#chart").e Width h\$("#chart").e h of jChart({ mar series: [{ ker marker: { size: { width: { width: 30} } }]);}); 25 } });});ch art.appe ndTo('#c hart); Property:w idthlet chart: Chart = new Property:size.widt h\$("#chart").e Width h\$("#chart").e h of jChart({ mar series: [{ ker marker: { size: { width: { width: 30} } }]);}); 25 } });});ch art.appe ndTo('#c hart); Property: marker.dat aLabellet chart: Chart = new Property:marker.d ataLabel\$("#char t").ejChart({ series: [{ marker: {dataLabel: { } } }]);}); { dataLabe l: { } } });});ch art.appe </pre>
--	--	--	--	---

				<pre> ndTo('#c hart); Property:d ataLabel.vi siblelet chart: Chart = new Property:dataLabel new Chart({ series: [{ marker: { dataLabe l: { visible: true } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.n amelet chart: Chart = new Property:dataLabel new Chart({ series: [{ marker: { dataLabe l: { name: '' } } }];});ch art.appe ndTo('#c hart); Property:d ataLabel.fil let chart: Chart = new Property:dataLabel new Chart({ series: [{ marker: { fill: 'pink' } } }];}); </pre>
--	--	--	--	--

				<pre> dataLabel: { fill: 'pink' } } });});chart.appendTo('#chart'); Property: dataLabel.fill let chart; Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.6 } } } }] });});chart.appendTo('#chart'); Property: dataLabel.position let chart; Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } }] });});chart.appendTo('#chart'); Align Property: dataLabel Property: dataLabel.alignment men .verticalAlignment ataLabel.al t of \$("#chart").ej ignmentle </pre>
--	--	--	--	--

					<pre> data Chart({ t chart: label series: [{ Chart = marker: new {dataLabel: { Chart({ verticalAlignm series: ent: 'near' } [{ }]});}; marker: { dataLabe l: { alignmen t: 'Near' } } }]);};ch art.appe ndTo('#c hart); </pre>
					<pre> Property:d ataLabel.al ignmentle t chart: Chart = new Property:dataLabel .border\$("#char t").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4} } }]});}; </pre>
					<pre> Property:dataLabel .offset\$("#chart ").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }]});}; </pre>
					<pre> Property:d ataLabel.m </pre>

				<pre> of t").ejChart({ data series: [{ label marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10}} } }];}); let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } } }];});chart.appendTo('#chart'); Property: dataLabel let chart: Chart = new Chart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' } } } } }];}); </pre>
--	--	--	--	--

				<pre> 'red' }} } }};});ch art.appe ndTo('#c hart); Property:d ataLabel.te mplatelet chart: Chart = new Chart({ series: [{ marker: {dataLab el: { template : ' ' } } } }};});ch art.appe ndTo('#c hart); Property:d ataLabel.rx let chart: Chart = new Chart({ series: [{ marker: {dataLab el: { rx: 10 } } }};});ch art.appe ndTo('#c hart); Property:d ataLabel.ry let chart: Chart = new </pre>
				<pre> HTML L tem plat e in data Label Property: dataLabel .template\$ ("#cha rt").ejChart({ series: [{ marker: {dataLabel: { template: ' ' } } } }};}); </pre>
				<pre> Rou nde d corn er radi us X Not Applicable </pre>
				<pre> Rou nde d corn er Not Applicable </pre>

				<pre> radi us Y Chart({ series: [{ marker: {dataLabel: { ry: 10 } } }]});chart.append ndTo('#c hart); </pre>
				<pre> Maxi Property:dataLabel mu .maximumLabelWi m dth\$("#chart") Labe .ejChart({ series: [{ Not widt marker: Applicable h for {dataLabel: { data maximumLabelWi label dth: 20}} }]]}); </pre>
				<pre> Enab Property:dataLabel le .enableWrap\$("# wra chart").ejChar ppin t({ series: [{ Not g of marker: Applicable 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 }} }]]}); </pre>
				<pre> To Property:dataLabel sho .showEdgeLabels\$ w ("#chart").ejC Not edge hart({ series: Applicable label [{ marker: for {dataLabel: { showEdgeLabels </pre>

				<pre> data : true }} } label }};)); ## TrendLines Be ha vi API in Essential JS API in Essential ou 1 JS 2 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] }] }); Property:trend Lines.typelet Ty Property:trendLi chart: pe nes.type\$("#ch Chart = new of art").ejChart Chart({ tre ({ series: [{ series: [{ nd trendLines: trendLines: Lin [type: [type: e 'linear' 'Polynomial]]] }] }); char t.appendTo('#chart'); Na Property:trendLi Property:trend m nes.name\$("#c Lines.namele e hart").ejChar t chart: of t({ series: Chart = new tre [{ Chart({ nd trendLines: [series: [{ name: trendLines: </pre>
--	--	--	--	---

				<pre> Lin 'trendLine' [name: e]}}}); 'trendLine' }}});chart .appendTo('#chart'); Property:trend Lines.periodle Pe Property:trendLi t chart: rio nes.period\$("#c Chart = new d hart").ejChar Chart({ of t({ series: series: [{ tre [{ trendLines: trendLines: nd trendLines: [[period: Lin period: 45 45]]}}]);cha e]}}}); rt.appendTo ('#chart'); Po ly no mi al or de Property:trend r nes.polynomial Lines.polynomi for nes.polynomial alOrderlet Po rder\$("#chart chart: ly ").ejChart({ Chart = new series: [{ series: [{ no trendLines: trendLines: mi polynomialOrd polynomialO al er: 3]}}]); rder: ty 3]]}}]);char pe t.appendTo(tre '#chart'); nd Lin es Ba Property:trend ck nes.backward Lines.backward w nes.backwardfor dforecastlet ar ecast\$("#chart chart: d ").ejChart({ Chart = new for series: [{ series: [{ ec trendLines: trendLines: os backwardforec [t ast: 3]}}]); backwardfor for ecast: 3]]}}]);char </pre>
--	--	--	--	--

				<pre> tre t.appendTo(nd '#chart'); Lin es Fo Property:trend rw Lines.forwardF ar orecastlet d Property:trendLi chart: for nes.forwardFore Chart = new ec cast\$("#chart" Chart({ os).ejChart({ series: [{ t series: [{ trendLines: for trendLines: [[tre forwardForeca forwardFore nd st: 3]]]}); cast: Lin 3]]]);char es t.appendTo('#chart'); Property:trend Lines.filllet Fill Property:trendLi chart: for nes.fill\$("#char Chart = new t").ejChart({ Chart({ tre series: [{ series: [{ nd trendLines: [trendLines: Lin fill: [fill: es '#EEFFCC' 'EEFFCC']] });chart.ap pendTo('#ch art'); Property:trend Lines.widthle Wi Property:trendLi t chart: dt nes.width\$("#c Chart = new h hart").ejChar Chart({ for t({ series: series: [{ tre [{ trendLines: nd trendLines: [[width: Lin width: 2 2]]]);char es]]]}); t.appendTo('#chart'); Int Property:trendLi Property:trend er nes.intercept\$(" Lines.intercept ce #chart").ejCh let chart: pt art({ series: Chart = new val [{ Chart({ ue trendLines: [series: [{ trendLines: </pre>
--	--	--	--	--

				<pre> for intercept: 2 [tre]}}}); nd intercept: Lin 2}}});char es t.appendTo('#chart'); </pre>
				<pre> Le Property:trend ge Lines.legendSh nd apelet sh chart: ap Chart = new e Chart({ for series: [{ tre trendLines: nd [Lin legendShape es : 'Rectangle']}}});chart .appendTo('#chart'); </pre>
				<pre> An Property:trend im Lines.animation ati nlet chart: on Chart = new se Chart({ tti series: [{ ng trendLines: s [for animation: tre { enable: nd true Lin }]]}});char es t.appendTo('#chart'); </pre>
				<pre> M Property:trend ar Lines.marker1 ke et chart: r Chart = new se Chart({ tti series: [{ ng trendLines: s [{marker: for { visible: tre true nd }]]}}});cha Lin rt.appendTo es ('#chart'); </pre>

				<pre> Property:trend Lines.enableTo oltiplet chart: Chart = new Chart({ series: [{ trendLines: [[{ tooltip: { {enableTool tip: true }}]]});char t.appendTo('#chart'); Da sh Property:trendLi Ar nes.dashArray\$(ra "#chart").ejC y hart({ series: [{ trendLines: [{ dashArray: '10, 5' } }}]]}); es Vis ibl Property:trendLi e nes.visibleOnLeg on end\$("#chart" le).ejChart({ ge series: [{ trendLines: [for { tre visibleOnLege nd: true } nd }}}]); Lin es ## Striplines Be hav API in Essential API in Essential iou JS 1 JS 2 r Def Property:primar Property:prima aul yXAxis.striplines ryXAxis.stripli t \$("#chart"). neslet be ejChart({ chart: </pre>
--	--	--	--	---

				<pre> hav primaryXAxis Chart = new iou : { Chart({ r stripLines: primaryXAxis for [{ visible: s: { stri true }]]}); stripLines: pli [{ nes visible: true .appendTo('#chart'); }]]});chart Property:stripLines.borderle Property:stripLines.borderle es.borderColor\$ Chart = new bor ("#chart").ejChart({ der jChart({ primaryXAxis for primaryXAxis s: { stri : { stripLines: pli stripLines: [{ border: ne [{ { color: borderColor: 'red', 'pink' width: 2} }]]}); chart .appendTo('#chart'); }]]}); Property:stripLines.borderle Bac Property:stripLines.borderle kgr es.color\$("#chart").ejChart = new ou es.color\$("#chart").ejChart({ nd t({ Chart({ col primaryXAxis s: { or : { stripLines: for stripLines: [{ color: stri [{ color: 'red'}}]]}); pli 'pink' chart.append ne }]]}); dTo('#chart '); Property:stripLines.startlet Sta Property:stripLines.startlet rt es.start\$("#chart").ejChart = new val es.start\$("#chart").ejChart({ ue t({ Chart({ for primaryXAxis s: { stri : { stripLines: pli stripLines: [{ start: ne [{ start: 5}}]]});char 10 }]]}); t.appendTo('#chart'); </pre>
--	--	--	--	---

				<pre> Property:stripLines.endlet En Property:stripLines.endlet d es.end\$("#chart").ejChart val rt").ejChart ue ({ for primaryXAxis stri : { pli stripLines: ne [{ end: 10 }]]}); chart.appendTo('#chart'); </pre>
				<pre> Property:stripLines.startFromAxislet Sta Property:stripLines.startFromAxislet rtfr es.startFromAxis om \$("#chart"). ejChart({ Axi primaryXAxis s : { for stripLines: stri [{ pli startFromAxis: ne s: true }]]}); chart.appendTo('#chart'); </pre>
				<pre> Property:stripLines.textlet Property:stripLines.textlet es.text\$("#chart").ejChart Tex rt").ejChart t in ({ stri primaryXAxis pli : { ne stripLines: [{ text: 'StripLine; }]]}); chart.appendTo('#chart'); </pre>
				<pre> Property:stripLines.horizontalAlignmentlet Tex Property:stripLines.horizontalAlignmentlet t es.textAlignment alig t\$("#chart"). nm .ejChart({ ent primaryXAxis in : { stri stripLines: [{ textAlignment: </pre>

				<pre> pli t: 'Far; ne }}}}); horizontalA lignment: 'Far'}}})); chart.appen dTo('#chart '); </pre>
				<pre> Ver tica l Tex t alig nm ent in stri pli ne </pre> <p>Not Applicable</p> <pre> Property:stripl ines.verticalAli gnmentlet chart: Chart = new Chart({ primaryXAxis s: { stripLines: [{ verticalAli gnment: 'Far'}}})); chart.appen dTo('#chart '); </pre>
				<pre> Property:stripl ines.sizelet chart: Siz es.width\$("#ch e art").ejChar of t({ stri primaryXAxis pli : { ne stripLines: [{ width: 10; }]]}); </pre> <pre> Property:stripl ines.sizelet chart: Chart = new Chart({ primaryXAxis s: { stripLines: [{ size: 10 }]]});chart .appendTo('# #chart'); </pre>
				<pre> ZIn es.zIndex\$("#c dex hart").ejCha of rt({ stri primaryXAxis pli : { ne stripLines: [{ zIndex: 'Behind' }]]}); </pre> <pre> Property:stripl ines.sizelet chart: Chart = new Chart({ primaryXAxis s: { stripLines: [{ zIndex: 'Behind' }]]});chart .appendTo('# #chart'); </pre>
				<pre> Fo Property:stripl nt es.fontStyle\$(" Property:stripl ines.textStyle1 </pre>

				<pre> styl #chart").ejC et chart: e hart({ Chart = new of primaryXAxis Chart({ : { primaryXAxis stripLines: s: { pli [{ stripLines: ne fontStyle: [{ {}]]]]); textStyle: {}]]]]); {} }}]); chart .appendTo('# #chart'); ## Multilevel Labels Be hav API in Essential JS API in iou 1 Essential JS 2 r Def Property:prim aul aryXAxis.muti t levellLabelsle be Property:primary t chart: XAxis.multilevell 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') ; Def Property:prim aul aryXAxis.muti t levellLabelsle be Property:primary t chart: XAxis.multilevell 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') ; </pre>
--	--	--	--	---

				<pre> Property:mult ilevelLabels.ali gnmentlet t Property:multile chart: t velLabels.textAlig Chart = alig nment\$("#char new nm t").ejChart({ Chart({ ent primaryXAxis: primaryXAx for { is: { mu multilevelLab multilevel ltile els: [{ Labels: [vel textAlignment {alignment Lab : 'Near' : 'Near' els }}}}); }}}});char t.appendTo ('#chart') ; </pre>
				<pre> Property:mult ilevelLabels.o verFlowlet t Property:multile verFlowlet t velLabels.textOv chart: ove erFlow\$("#cha new rflo rt").ejChart(Chart({ w { primaryXAx for primaryXAxis: is: { mu { multilevel ltile multilevelLab Labels: [vel els: [{ {overflow: Lab textOverFlow: 'Trim' els 'Trim' }}}}); }}}});char t.appendTo ('#chart') ; </pre>
				<pre> Property:mult ilevelLabels.b orderlet t Property:multile orderlet t velLabels.border chart: Bor \$("#chart").e Chart = der jChart({ new for primaryXAxis: Chart({ mu { primaryXAx ltile multilevelLab is: { vel els: [{ multilevel Lab border: { Labels: [els width: 2, { border: color: 'red', { width: type: 'brace' 2, color: } }}}}); 'red', type: 'brace' } }}}});char </pre>

				<pre> t.appendTo ('#chart') ; Property:mult iLevelLabels.c ategories.star tlet chart: Chart = new Sta velLabels.start\$(Chart({ rt "#chart").ejC primaryXAx val hart({ is: { ue primaryXAxis: multilevel for { Labels: [lab multilevelLab { el els: [{ categories start: 45 } : [{ start: }}] } }}] } }}] } ;char t.appendTo ('#chart') ; Property:mult iLevelLabels.c ategories.end let chart: Chart = En Property:multile new d velLabels.start\$(Chart({ "#chart").ejC primaryXAx val hart({ is: { ue primaryXAxis: multilevel for { Labels: [lab multilevelLab { el els: [{ end: categories 45 } }] } } : [{ end: 45 } }] } } } } ;char t.appendTo ('#chart') ; Property:multile Property:mult Tex velLabels.text\$(iLevelLabels.c t "#chart").ejC ategories.text for hart({ let chart: lab primaryXAxis: Chart = el { new multilevelLab Chart({ els: [{ primaryXAx is: { </pre>
--	--	--	--	---

				<pre> text: 'Start' multilevel }]]]]); Labels: [{ categories : [{ text: 'text' }] }]]]]];char t.appendTo ('#chart') ; Property:multilevelLabels.categories.maximumTextWidth let chart: Chart = new Chart({ primaryXAxis: is: { multilevel Labels: [{ categories : [{ maximumTextWidth: 20 }] }]]]]];char t.appendTo ('#chart') ; Property:multilevelLabels.level\$("#chart").ejC Not hart({ primaryXAxis: applicable. { Categories are multilevelLabels used els: [{ level: 2 }]]]]]); ## Methods Be API in API in Essential JS ha Essential JS 1 2 </pre>
--	--	--	--	--

				<pre> vio ur ani ma tio n for ser ies </pre>	<pre> Property:chart.animate\$ ("#chart") .ejChart ({ animate: () { }}); </pre>	Not applicable
				<pre> Re dra w for cha rt </pre>	<pre> Property:chart.redraw\$ ("#chart") .ejChart ({ redraw: () { }}); </pre>	<pre> Property:chart.refresh() let chart: Chart = new Chart({});char t.appendTo('#c hart');chart.w idth = '400';chart.re fresh(); </pre>
				<pre> Exp ort </pre>	<pre> Property:chart.export\$ ("#chart") .ejChart ({ export: () { }}); </pre>	<pre> Property:chart.export() let chart: Chart = new Chart({});char t.export('JPEG ', 'chart');chart .appendTo('#ch art'); </pre>
				<pre> Pri nt </pre>	<pre> Property:chart.print\$ ("#chart") .ejChart ({ print: () { }}); </pre>	<pre> Property:chart.print() let chart: Chart = new Chart({});char t.print('chart ');chart.appen dTo('#chart'); </pre>
				<pre> Ad ds eri es </pre>	Not Applicable	<pre> Property:chart.addSeries() let chart: Chart = new Chart({});char t.appendTo('#c hart');chart.a ddSeries(); </pre>
				<pre> Re mo veS </pre>	Not Applicable	<pre> Property:chart.removeSeries() let chart: Chart = new </pre>

				<pre> eri Chart({});char es t.appendTo('#c hart');chart.r emoveSeries(); ## Events Beh API in avi API in Essential JS 1 our Essential JS 2 Fire s on Property:annotatio ann nClick\$("#chart" ota).ejChart({ tio annotationClic n k: () { }}); clic k Property:an imationCo mplete()le t chart: Chart = new Property:animation Complete\$("#cha rt").ejChart({ animationCompl ete: () { }}); e: () => { }});char t.append To('#cha rt'); Fire s on Property:axisLabelC axis lick\$("#chart"). lab ejChart({ el axisLabelClick clic : () { }}); k Fire s Property:axisLabelR bef endering\$("#char ore t").ejChart({ axis axisLabelRende Property:ax isLabelRend er()let chart: Chart = </pre>
--	--	--	--	--

				<pre> lab ring: () { el }); ren der new Chart({ axisLabe lRender: () => { }});char t.append To('#cha rt'); Fire s on Property:axisLabel axis MouseMove\$("#c lab hart").ejChart el ({ axisLabelMouse mo Move: () { use }); Mo ve Fire s on Property:axisLabelI axis nitialize\$("#chart lab ").ejChart({ el axisLabelIniti alizer alizer e Fire s before Property:axesRange axis eCalculate\$("#cha ran rt").ejChart({ ge axesRangeCalcu calc late: () { ula }); tio n Fire Property:axisTitleR s ending\$("#char on t").ejChart({ axis axisTitleRende </pre>
				<pre> Not applicable Not applicable Property:ax isRangeCalc ulated()let chart: Chart = new Chart({ axisRang eCalcula ted: () => { }});char t.append To('#cha rt'); Not applicable </pre>

				<pre> titl ring: () { e }); ren der ing Fire s on Property:afterResiz aft e\$("#chart").e Not er jChart({ applicable cha afterResize: rt () { }); resi ze Fire s on Property:beforeRes bef ize\$("#chart"). ore ejChart({ cha beforeResize: rt () { }); resi ze Property:re sizedlet chart: Chart = new Chart({ resized: () => { });char t.append To('#cha rt'); Property:ch artMouseCl icklet chart: Chart = new Chart({ chartMou seClick: () => { });char t.append To('#cha rt'); Fire s on Property:chartClick \$(\$("#chart").ej cha Chart({ rt chartClick: () clie { }); k Property:ch artMouseM ovelet chart: Chart = new Chart({ chartMou seMove\$("#char t").ejChart({ chartMouseMove : () { }); mo use </pre>
--	--	--	--	--

				<pre> move: () => { chart.appendTo('#chart'); } Property:chartMouseLeave: () => { chart: new Chart({ chartMouseLeave: () => { chart.appendTo('#chart'); } }); } Fire: () => { chart: new Chart({ chartDoubleClick: () => { chart.appendTo('#chart'); } }); } Property:chartDoubleClick: () => { chart: new Chart({ chartDoubleClick: () => { chart.appendTo('#chart'); } }); } Property:chartMouseUp: () => { chart: new Chart({ chartMouseUp: () => { chart.appendTo('#chart'); } }); } Property:chartMouseDown: () => { chart: new Chart({ chartMouseDown: () => { chart.appendTo('#chart'); } }); } </pre>
--	--	--	--	---

				<div><div>on cha rt mo use do wn</div><div>Fire s dur ing the calc ula tio n of cha rt are a bo un ds. You can use this eve nt to cus to miz e the bo un ds of cha rt</div></div>	<div><div>ownlet chart: Chart = new Chart({ chartmou seDown: () => { }});char t.append To('#cha rt');</div><div>Property:chartArea BoundsCalculate\$("#chart").ejCh art({ chartAreaBound sCalculate: () { } });</div><div>Not applicable</div></div>
--	--	--	--	---	---

				<pre> are a Fire s wh en Property:dragStart the \$("#chart").ej Not dra Chart({ applicable ggi dragStart: () ng { }); is star ted Fire s Property:dragging\$ whi (\$("#chart").ejC Not le hart({ applicable dra dragging: () { ggi }); ng Fire s wh en Property:dragEnd\$ dra (\$("#chart").ejC ggi hart({ ng dragEnd: () { is }); co mpl ete d Fire s wh en Property:destroy\$(cha "#chart").ejCh Not rt art({ destroy: applicable is () { }); des tro yed co </pre>
--	--	--	--	---

				<pre> 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. Fire s, wh en err or bar is ren der ing. Fire s dur ing the calc ula </pre>	<pre> Property:loaded let chart: Chart = new Chart({ loaded: () => { }});char t.append To('#cha rt'); Property:textRender let chart: Chart = new Chart({ textRender: () => { }});char t.append To('#cha rt'); Not applicable Not applicable </pre>
--	--	--	--	--	---

				<p>tion n of leg end bo un ds.</p> <p>Fire s on clic kin g the leg end ite m.</p> <p>Fire s wh en mo vin g mo use ove r leg end ite m</p> <p>Fire s bef ore ren der ing the leg end</p>	<p>Property:legendItemClick\$("#chart").ejChart({legendItemClick: () { }});</p> <p>Property:legendItemMouseMove\$("#chart").ejChart({legendItemMouseMove: () { }});</p> <p>Property:legendRenderlet chart: Chart = new Chart({legendRender: () => { }}); chart.append</p>	<p>Not applicable</p> <p>Not applicable</p> <p>Property:legendRender</p>
--	--	--	--	---	--	--

				<pre> ite m. Fire s bef ore loa din g the cha rt. Fire s, wh en mul ti lev el lab els are ren der ing. Fire s on clic kin ga poi nt in cha rt. Fire s wh en mo use </pre>	<pre> To('#cha rt'); Property:lo adlet chart: Chart = new Chart({ load: () => { }});char t.append To('#cha rt'); Property:ax isMultiLabe lRender let chart: Chart = new Chart({ axisMult iLabelRe nder : () => { }});char t.append 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({ </pre>
--	--	--	--	---	---

				<pre> is seMove: () { pointMov mo }}}; e : () ved => { ove }}};char ra t.append poi To('#cha nt. rt'); Fire s bef Property:preRende ore r\$("#chart").ej Not ren Chart({ applicable der preRender: () ing { }}}); cha rt. Property:po intRender let chart: Chart = new Chart({ pointRen der : () => { }});char t.append To('#cha rt'); Fire s aft er sel Property:rangeSele ect cted\$("#chart") Not ed .ejChart({ applicable the rangeSelected: dat () { }}}); a in cha rt. Fire Property:seriesRegi Not s onClick\$("#chart applicable aft ").ejChart({ </pre>
--	--	--	--	--

				<pre> er seriesRegionCl sel ick: () { }); ecti ng a seri es. Fire s bef ore Property:seriesRender rendering\$("#chart").ejChart({ seriesRenderin g: () { }); a seri es. Fire s bef ore render der Property:symbolRe ndering\$("#chart ").ejChart({ symbolRenderin g: () { }); r sy mb ols. Fire s bef ore Property:trendline Rendering\$("#cha rt").ejChart({ trendlineRende ring: () { }); the tre ndli ne </pre>
				<pre> Property:se riesRender let chart: Chart = new Chart({ seriesRe nder : () => { });char t.append To('#cha rt'); Property:symbolRe ndering\$("#chart ").ejChart({ symbolRenderin g: () { }); Not applicable Property:trendline Rendering\$("#cha rt").ejChart({ trendlineRende ring: () { }); Not applicable </pre>

				<pre> Fire s bef ore ren Property:titleRende der ring\$("#chart") ing .ejChart({ Not the titleRendering applicable : () { })); Cha rt titl e. Fire s bef ore ren Property:subTitleR der endering\$("#char Not ing t").ejChart({ applicable the subTitleRender Cha ing: () { })); rt sub titl e. Fire s bef ore ren Property:toolTipInit der ialize\$("#chart") ing .ejChart({ the toolTipInitial too ize: () { })); ltip . Fire s bef ore ren der ing Property:trackAxisT oolTip\$("#chart" Not).ejChart({ applicable trackAxisToolT ip: () { })); </pre>
--	--	--	--	---

				<pre> cro ssh air too ltip in axis Fire s bef ore ren Property:trackTool der Tip\$("#chart"). Not ing ejChart({ applicable tra trackToolTip: ckb () { })); all too ltip . Eve Property:sc nt rollStart trig let ger Property:scrollStart Chart = ed \$("#chart").ej new wh Chart({ Chart({ en scrollStart: scrollSt scr () { })); art : () oll => { star }});char ts. t.append To('#cha rt'); Eve 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'); </pre>
--	--	--	--	---

Event triggered when scroll changes.	Property:scrollChange	let chart: Chart = new Chart({ scrollChange: () => { }}); chart.appendTo('#chart');
Fire while performing minimum recording tangent zooming in chart.	Property:zoomComplete	let 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, x: 0, y: 1}]});	Property:selectedDataIndexes let chart: Chart = new Chart({selectedDataIndexes: [{ series: 0, point: 1}]}); chart.appendTo('#chart');
sideBySide Series	Property:sideBySideSeriesPlacement:\$(Property:sideBySidePlacement let chart:

				<pre> sPlac "#chart"). Chart = new emen ejChart({ Chart({ t for sideBySide sideBySideP colu SeriesPlac lacement: mn ument)); true});char base t.appendTo(d '#chart'); series </pre>
				<pre> Property:zoom Settingslet Property:zoo chart: ming:\$("#ch Chart = new art").ejCh Chart({ art({ zoomSetting zooming: { s: { enable: enable: true, true, enabledefe enablePinch rredZoom: Zooming: Zoom true, true, Setti enablePinc enableDeffe ngs h: true, redZooming: enableMous true eWheel: enableMouse true, WheelZoomin enableSrol g: true, lBar: enableSelec true, tionZooming toolBarIte : true, ms: [], enableScrol type: 'X' lBar: true }}}); chart.a ppendTo('#c hart'); </pre>
				<pre> Back Property:bac Property:backg grou kground roundlet nd chart: color \$("#contai Chart = new of ner").ejCh Chart({ the art({ background: chart : '#EEFFCC'}) 'transpare ;chart.appe nt'}}); ndTo('#char t'); </pre>
				<pre> URL Property:bac of kGroundImag the eUrl Not Applicable imag \$("#contai e to ner").ejCh art({ </pre>

				<pre> be backGround used imageUrl : as '../images chart /chart/whe at.png'}}); backg roun d. </pre>
				<pre> Property:bor Property:borde der Custo \$("#contai Chart = new mizin ner").ejCh Chart({ g art({ border: { bord border: { width: 2, er of width: 2, color: the color: '#CCEEFF'}} chart '#CCEEFF',);chart.app opacity: endTo('#cha 0.5}}); rt'); </pre>
				<pre> This Property:export provi Property:exp des ertSettings\$ optio "#containe ns for r").ejChar custo t({ mizin exportSett g ings: { expor filename : t "chart", settin angle: gs '45' })); export(type , fileName); </pre>
				<pre> Property:char Property:chartA tArea\$("#co realet ntainer"). chart: ejChart({ Chart = new chartArea: Chart({ Chart { Area chartArea: custo : mizat 'transparent' ion 't', border: { width: 2, opacity: color: 0.3, '#CCEEFF' color: });chart.a 'red', ppendTo('#c width: hart');char 2}})); t.export(ty </pre>

				pe, fileName);
--	--	--	--	-------------------

<tr>

<td>Behaviour</td>

<td>API in Essential JS 1</td>

<td>API in Essential JS 2</td>

</tr>

<tr>

<td>crosshair</td>

<td>

Property<i>visible</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
crosshair: { visible: true}  
});
```

</code>

</td>

<td>

Property<i>enable</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
crosshair: { enable: false }  
});  
chart.appendTo('#chart');  
</code></td>
```

</tr>

<tr>

<td>trackballTooltipSettings</td>

<td> Property:<i>trackballTooltipSettings</i> </br> </br> <code> \$("#container").ejChart({ crosshair : { trackballTooltipSettings : { border : { width :2 } } } }); </code> </td>	<td> Not applicable </td>
<td>marker</td> <td> Property:<i>marker</i> </br> </br> <code> \$("#container").ejChart({ crosshair : { marker : { border : { width :2 } } } }); </code> </td>	<td> Not applicable </td>
<td>crosshair line style</td> <td>	

```
<b>Property</b><i>line</i>
</br>
</br>
<code>
$("#container").ejChart({
  crosshair : { line: { width: 2, color: 'red' } }
});
</code>
</td>
<td>
let chart: Chart = new Chart({
  crosshair: { border: { width: 2, color: 'black' } }
});
</td>
</tr>
<tr>
<td><b>type</b></td>
<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	<code>Property:enable3D\$("#container").ejChart({ enable3D: true});</code>	Not applicable
Rotation of 3d chart	<code>Property:enableRotation\$("#container").ejChart({ enableRotation: false});</code>	Not applicable
depth	<code>Property:depth\$("#container").ejChart({ depth: 45});</code>	Not applicable

Canvas rendering

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
canvas rendering	<code>Property:enableCanvasRendering\$("#container").ejChart({ enableCanvasRendering: true});</code>	Not applicable

Indicators

<!-- markdownlint-disable MD033 -->

Behaviour	API in Essential JS 1	API in Essential JS 2
Type of Indicator	<code>Property:type\$("#container").ejChart({ indicators: [{ type: 'Sma' }]});</code>	<code>Property:typelet chart: Chart = new Chart({ indicators: [{ type: 'Sma' }]});</code>
Period for indicator	<code>Property:period\$("#container").ejChart({ indicators: [{ period: 14 }]});</code>	<code>Property:typelet chart: Chart = new Chart({ indicators: [{ period: 14 }]});</code>
Period for indicator	<code>Property:period\$("#container").ejChart({ indicators: [{ period: 14 }]});</code>	<code>Property:periodlet chart: Chart = new Chart({ indicators: [{ period: 14 }]});</code>
%K value in stochastic indicator	<code>Property:kPeriod\$("#container").ejChart({ indicators: [{ kPeriod: 14 }]});</code>	<code>Property:kPeriodlet chart: Chart = new Chart({ indicators: [{ kPeriod: 14 }]});</code>
%D value in stochastic indicator	<code>Property:dPeriod\$("#container").ejChart({ indicators: [{ dPeriod: 3 }]});</code>	<code>Property:dPeriodlet chart: Chart = new Chart({ indicators: [{ dPeriod: 3 }]});</code>
Shows overSold/overBought values	Not applicable	<code>Property:showZoneslet chart: Chart = new Chart({ indicators: [</code>

		<code>{ 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").ejChart({ indicators: [{ standardDeviations: 2 }]});</code>	Property:standardDeviation <code>let chart: Chart = new Chart({ indicators: [{ standardDeviation: 2 }]});</code>
standard deviation	Property:standardDeviations <code>\$("#container").ejChart({ indicators: [{ standardDeviations: 2 }]});</code>	Property:standardDeviation <code>let 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({ indicators: [{ field: 'Close' }]});</code>
Slow period for MACD indicator	Property:shortPeriod <code>\$("#container").ejChart({ indicators: [{ shortPeriod: 12 }]});</code>	Property:slowPeriod <code>let chart: Chart = new Chart({ indicators: [{ slowPeriod: 12 }]});</code>
Fast period for MACD indicator	Property:longPeriod <code>\$("#container").ejChart({ indicators: [{ longPeriod: 26 }]});</code>	Property:fastPeriod <code>let chart: Chart = new Chart({ indicators: [{ fastPeriod: 26 }]});</code>
Line style for MACD indicator	Property:macdLine <code>\$("#container").ejChart({ indicators: [{ macdLine: { width: 2, fill: 'red' } }]});</code>	Property:fastPeriod <code>let chart: Chart = new Chart({ indicators: [{ macdLine: { width: 2, color: 'red' } }]});</code>
Type of MACD indicator	Property:macdType <code>\$("#container").ejChart({ indicators: [{ macdType: 'both' }]});</code>	Property:fastPeriod <code>let chart: Chart = new Chart({ indicators: [{ macdType: 'Both' }]});</code>

Color of the positive bars in Macd indicators	Not applicable	Property:macdPositiveColor let chart: Chart = new Chart({ indicators: [{ macdPositiveColor: 'red' }] });
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: '#FFEEEE', 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: '#FFEEEE', 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: '#FFEEEE', 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	Property:seriesName \$("#container").ejChart({ indicators: [{ histogram: { } }] });	Not applicable

histogram in MACD indicator		
Enabling animation	<code>Property:enableAnimation\$("#container").ejChart({ indicators: [{ enableAnimation: true }]});</code>	<code>Property:animation.enable1 let chart: Chart = new Chart({ indicators: [{ animation: { enable: true } }]});</code>
Animation duration	<code>Property:animationDuration\$("#container").ejChart({ indicators: [{ animationDuration: 3000 }]});</code>	<code>Property:animation.duration let chart: Chart = new Chart({ indicators: [{ animation: { duration: 3000 } }]});</code>
Tooltip	<code>Property:tooltip\$("#container").ejChart({ indicators: [{ tooltip: { visible: true } }]});</code>	Not applicable
Trigger value of MACD indicator.	<code>Property:trigger\$("#container").ejChart({ indicators: [{ trigger: 14 }]});</code>	Not applicable
Fill color for indicator	<code>Property:fill\$("#container").ejChart({ indicators: [{ fill: '#EEDDCC' }]});</code>	<code>Property:animation.enable1 let chart: Chart = new Chart({ indicators: [{ fill: 'red' }]});</code>
Width for indicator	<code>Property:width\$("#container").ejChart({ indicators: [{ width: 2 }]});</code>	<code>Property:width let chart: Chart = new Chart({ indicators: [{ width: 3 }]});</code>
xAxis Name of indicator	<code>Property:xAxisName\$("#container").ejChart({ indicators: [{ xAxisName: '' }]});</code>	<code>Property:xAxisName let chart: Chart = new Chart({ indicators: [{ xAxisName: '' }]});</code>
yAxis Name of indicator	<code>Property:yAxisName\$("#container").ejChart({ indicators: [{ yAxisName: '' }]});</code>	<code>Property:yAxisName let 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:visible let chart: Chart = new Chart({</code>

		<pre>legendSettings: { visible: true } });</pre>
Legend height	<pre>Property: size.height\$("#container").ejChart({ legend: { size: { height: 50 } } });</pre>	<pre>Property: heightlet chart: Chart = new Chart({ legendSettings: { height: '30' } });</pre>
Legend width	<pre>Property: size.width\$("#container").ejChart({ legend: { size: { width: 20 } } });</pre>	<pre>Property: widthlet chart: Chart = new Chart({ legendSettings: { width: '30' } });</pre>
Legend location in chart	<pre>Property: location\$("#container").ejChart({ legend: { location: { x: 3, y: 45 } } });</pre>	<pre>Property: heightlet chart: Chart = new Chart({ legendSettings: { location: { x: 3, y: 45 } });</pre>
Legend position in chart	<pre>Property: position\$("#container").ejChart({ legend: { position: 'top' } });</pre>	<pre>Property: positionlet chart: Chart = new Chart({ legendSettings: { position: 'Top' } });</pre>
Legend padding	Not applicable	<pre>Property: paddinglet chart: Chart = new Chart({ legendSettings: { padding: 8 } });</pre>
Legend alignment	<pre>Property: position\$("#container").ejChart({ legend: { alignment: 'center' } });</pre>	<pre>Property: positionlet chart: Chart = new Chart({ 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	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:backgorundlet 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

<!-- 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 cross value	Property:crossesAt \$("#container").ejChart({ primaryXAxis: { crossesAt: 0 }});	Property:crossesAt let chart: Chart = new Chart({ primaryXAxis: { crossesAt: 4 }});chart.appendTo('#chart');
axis name with which the axis line has to be crossed	Property:crossesInAxis \$("#container").ejChart({ primaryXAxis: { crossesInAxis: '' }});	Property:crossesInAxis let chart: Chart = new Chart({ primaryXAxis: { crossesInAxis: '' }});chart.appendTo('#chart');
axis elements placed with	Property:showNextToAxisLine \$("#container").ejChart({ primaryXAxis: { showNextToAxisLine: true }});	Property:placeNextToAxisLine let chart: Chart = new Chart({ primaryXAxis: { placeNextToAxisLine: '' }});chart.appendTo('#chart');

axis line		
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.dashArray <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.visible <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { visible : false } } });</code>	Property:visible <code>let chart: Chart = new Chart({ primaryXAxis: { visible: false } });chart.appendTo('#chart');</code>
Width of an axis	Property:axisLine.width <code>\$("#container").ejChart({ primaryXAxis: { axisLine: { width : 2 } } });</code>	Property:lineStyle.width <code>let chart: Chart = new Chart({ primaryXAxis: { lineStyle: { width: 3 } } });chart.appendTo('#chart');</code>
Column index of an axis	Property:columnIndex <code>\$("#container").ejChart({ primaryXAxis: { columnIndex: 2 } });</code>	Property:columnIndex <code>let chart: Chart = new Chart({ primaryXAxis: { columnIndex: 2 } });chart.appendTo('#chart');</code>
span of an axis to place	Property:columnSpan <code>\$("#container").ejChart({ primaryXAxis: {</code>	Property:span <code>let chart: Chart = new Chart({ primaryXAxis: { span: 2 } });chart.appendTo('#chart');</code>

horizontally or vertically	<code>columnIndex: 2 }});</code>	
Cross hair 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>
Cross hair label color of an axis	Not applicable	<code>Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { fill: 'red' }}});chart.appendTo('#chart');</code>
Cross hair label text style	Not applicable	<code>Property:crossHairTooltip.textStylelet chart: Chart = new Chart({ primaryXAxis: { crossHairTooltip: { textStyle: { } }}});chart.appendTo('#chart');</code>
Desired interval count for primary X Axis	<code>Property:desiredIntervals\$("#container").ejChart({ primaryXAxis: { desiredIntervals: 4 }});</code>	<code>Property:desiredIntervalslet chart: Chart = new Chart({ primaryXAxis: { desiredIntervals: 4 }});chart.appendTo('#chart');</code>
Edge label placement for primary X Axis	<code>Property:edgeLabelPlacement\$("#container").ejChart({ primaryXAxis: { edgeLabelPlacement: 'none' }});</code>	<code>Property:edgeLabelPlacementlet chart: Chart = new Chart({ primaryXAxis: { edgeLabelPlacement: 'Shift' }});chart.appendTo('#chart');</code>
Enables trim for axis	<code>Property:enableTrim\$("#container").ejChart({ primaryXAxis: {</code>	<code>Property:enableTrimlet chart: Chart = new Chart({ primaryXAxis: { enableTrim: true }});chart.appendTo('#chart');</code>

label s	<code>enableTrim: true }));</code>	
Speci fies the inter val of the axis accor ding to the zoo med data of the chart	<code>Property:enableAut oIntervalOnZoomin g\$("#container").ejChart({ primaryXAxis: { enableAutoInter valOnZooming: true }));</code>	<code>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart');</code>
Speci fies the inter val of the axis accor ding to the zoo med data of the chart	<code>Property:enableAut oIntervalOnZoomin g\$("#container").ejChart({ primaryXAxis: { enableAutoInter valOnZooming: true }));</code>	<code>Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ primaryXAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart');</code>
Font style for prim aryX Axis	<code>Property:font\$("# container").ejC hart({ primaryXAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '',</code>	<code>Property:titleStylelet chart: Chart = new Chart({ primaryXAxis: { titleStyle: { } } });chart.appendTo('#chart');</code>

	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:intervalTy pe\$("#container") .ejChart({ primaryXAxis: { intervalType: 'Auto' } });	Property:intervalTypelet chart: Chart = new Chart({ primaryXAxis: { intervalType: 'Auto }});chart.appendTo('#chart');
Inver sed axis	Property:isInversed \$("#container") .ejChart({ primaryXAxis: { isInversed: true } });	Property:isInversedlet chart: Chart = new Chart({ primaryXAxis: { isInversed: true }});chart.appendTo('#chart');
Cust om label form at	Property:labelForm at\$("#container") .ejChart({ primaryXAxis: { labelFormat: '{value}K' } });	Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');
label inters ectA ction	Property:labelInters ectAction\$("#con tainer").ejChar t({ primaryXAxis: { labelIntersectA ction: 'trim' }});	Property:labelIntersectActionlet chart: Chart = new Chart({ primaryXAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');
label Positi on	Property:labelPositi on\$("#container") .ejChart({ primaryXAxis: { labelPosition: 'inside' } });	Property:labelPositionlet chart: Chart = new Chart({ primaryXAxis: { labelPosition: 'Inside' }});chart.appendTo('#chart');
label Place ment for categ	Property:labelPlace ment\$("#contain er").ejChart({ primaryXAxis: { labelPlacement: 'onTicks' } });	Property:labelPlacementlet chart: Chart = new Chart({ primaryXAxis: { labelPlacement: 'OnTicks' }});chart.appendTo('#chart');

Primary axis		
Axis label alignment	Property:alignment <code>\$("#container").ejChart({ primaryXAxis: { alignment: 'center' } } });</code>	Not Applicable
Rotation of axis labels	Property:labelRotation <code>\$("#container").ejChart({ primaryXAxis: { labelRotation: 45 } } });</code>	Property:labelRotation <code>let chart: Chart = new Chart({ primaryXAxis: { labelRotation: 45 } }); chart.appendTo('#chart');</code>
Log base value for logarithmic axis	Property:logBase <code>\$("#container").ejChart({ primaryXAxis: { logBase: 10 } } });</code>	Property:labelRotation <code>let chart: Chart = new Chart({ primaryXAxis: { logBase: 10 } }); chart.appendTo('#chart');</code>
Major grid line	Property:majorGridLines.visible <code>\$("#container").ejChart({ primaryXAxis: { majorGridLines: { visible: true } } } });</code>	Not Applicable
Width of Major Grid Lines	Property:majorGridLines.width <code>\$("#container").ejChart({ primaryXAxis: { majorGridLines: { width: 2 } } } });</code>	Property:majorGridLines.width <code>let chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { width: 2 } } }); chart.appendTo('#chart');</code>
Color of Major Grid Lines	Property:majorGridLines.color <code>\$("#container").ejChart({ primaryXAxis: { majorGridLines: { color: 'black' } } } });</code>	Property:majorGridLines.color <code>let chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { color: 'black' } } }); chart.appendTo('#chart');</code>

Dash Array of Major Grid Lines	Property:majorGridLines.dashArray\$ (" #container").ejChart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } } });	Property:majorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { majorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
Opacity of major grid line	Property:majorGridLines.opacity\$ (" #c ontainer").ejCh art({ primaryXAxis: { majorGridLines: { opacity: true } } });	Not Applicable
Major Tick line	Property:majorTickLines.visible\$ (" #co ntainer").ejCha rt({ primaryXAxis: { majorTickLines: { visible: true } } });	Not Applicable
Width of Major Tick Lines	Property:majorTickLines.width\$ (" #co ntainer").ejCha rt({ 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\$ (" #cont ainer").ejChart ({ primaryXAxis: { majorTickLines: { size: 2 } } });	Property:majorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { height: 2 } } });chart.appendTo('#chart');
Color of Major Tick Lines	Property:majorTickLines.color\$ (" #con tainer").ejChar t({ primaryXAxis: { majorTickLines: { color: 'black' } } });	Property:majorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { majorTickLines: { color: 'black' } } });chart.appendTo('#chart');
Opacity of	Property:majorTickLines.opacity\$ (" #c	Not Applicable

major Tick line	<pre> ontainer").ejChart({ primaryXAxis: { majorTickLines: { opacity: true} })); </pre>	
maximum labels of primary X Axis	<pre> Property:maximum Labels\$("#container").ejChart({ primaryXAxis: { maximumLabels: 5 })); </pre>	<pre> Property:maximumLabelslet chart: Chart = new Chart({ primaryXAxis: { maximumLabels: 4 }});chart.appendTo('#chart'); </pre>
maximum label width of primary X Axis to trim	<pre> Property:maximum LabelWidth\$("#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:minorGrid Lines.visible\$("#container").ejChart({ primaryXAxis: { minorGridLines: { visible: true} })); </pre>	Not Applicable
Width of minor Grid Lines	<pre> Property:minorGrid Lines.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:minorGrid Lines.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	Property:minorGridLines.dashArray\$ (" #container").ejChart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });	Property:minorGridLines.dashArraylet chart: Chart = new Chart({ primaryXAxis: { minorGridLines: { dashArray: 'black' } } });chart.appendTo('#chart');
Opacity of minor grid line	Property:minorGridLines.opacity\$ (" #c ontainer").ejCh art({ primaryXAxis: { minorGridLines: { opacity: true } } });	Not Applicable
minor Tick line	Property:minorTickLines.visible\$ (" #co ntainer").ejCha rt({ primaryXAxis: { minorTickLines: { visible: true } } });	Not Applicable
Width of minor Tick Lines	Property:minorTickLines.width\$ (" #co ntainer").ejCha rt({ primaryXAxis: { minorTickLines: { width: 2 } } });	Property:minorTickLines.widthlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { width: 2 } } });chart.appendTo('#chart');
Height of minor Tick Lines	Property:minorTickLines.size\$ (" #cont ainer").ejChart ({ primaryXAxis: { minorTickLines: { size: 2 } } });	Property:minorTickLines.heightlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { height: 2 } } });chart.appendTo('#chart');
Color of minor Tick Lines	Property:minorTickLines.color\$ (" #con tainer").ejChar t({ primaryXAxis: { minorTickLines: { color: 'black' } } });	Property:minorTickLines.colorlet chart: Chart = new Chart({ primaryXAxis: { minorTickLines: { color: 'black' } } });chart.appendTo('#chart');
Opacity of	Property:minorTickLines.opacity\$ (" #c	Not Applicable

minor Tick line	<pre> ontainer").ejChart({ primaryXAxis: { minorTickLines: { opacity: true} })); </pre>	
Minor ticks per interval of primaryX 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 primaryX Axis	<pre> Property:name\$("#container").ejChart({ primaryXAxis: { name: 'primaryXAxis' }}); </pre>	<pre> Property:namelet chart: Chart = new Chart({ primaryXAxis: { name: 'primaryXAxis' }});chart.appendTo('#chart'); </pre>
Orientati on of prim aryX Axis	<pre> Property:orientation\$("#container").ejChart({ primaryXAxis: { orientation: 'Vertical' })); </pre>	Not Applicable
Plot offset for prim aryX Axis	<pre> Property:plotOffset\$("#container").ejChart({ primaryXAxis: { plotOffset: 0 }}); </pre>	<pre> Property:plotOffsetlet chart: Chart = new Chart({ primaryXAxis: { plotOffset: 0 }});chart.appendTo('#chart'); </pre>
mini mum for prim aryX Axis	<pre> Property:range.minimum\$("#container").ejChart({ primaryXAxis: { range: { minimum: 10 }}}); </pre>	<pre> Property:minimumlet chart: Chart = new Chart({ primaryXAxis: { minimum: 23 }});chart.appendTo('#chart'); </pre>
maxi mum for prim	<pre> Property:range.maximum\$("#container").ejChart({ primaryXAxis: { range: { </pre>	<pre> Property:maximumlet chart: Chart = new Chart({ primaryXAxis: { maximum: 23 }});chart.appendTo('#chart'); </pre>

primaryX Axis	<pre>maximum: 10 }}});</pre>	
interval for primaryX Axis	<pre>Property:range.interval\$("#container").ejChart({ primaryXAxis: { range: { interval: 1 }}});</pre>	<pre>Property:intervallet chart: Chart = new Chart({ primaryXAxis: { interval: 2 }});chart.appendTo('#chart');</pre>
RangePadding for primaryX Axis	<pre>Property:rangePadding\$("#container").ejChart({ primaryXAxis: { rangePadding: 'None' }}});</pre>	<pre>Property:rangePaddinglet chart: Chart = new Chart({ primaryXAxis: { rangePadding: 'None' }});chart.appendTo('#chart');</pre>
Rounding Places in primaryX Axis	<pre>Property:roundingPlaces\$("#container").ejChart({ primaryXAxis: { roundingPlaces: 3 }});</pre>	<pre>Property:labelFormatlet chart: Chart = new Chart({ primaryXAxis: { labelFormat: 'n3' }});chart.appendTo('#chart');</pre>
Scrollbar settings of primaryX Axis	<pre>Property:scrollbarSettings\$("#container").ejChart({ primaryXAxis: { scrollbarSettings : { } }});</pre>	Not Applicable
TickPosition in primaryX Axis	<pre>Property:tickLinesPosition\$("#container").ejChart({ primaryXAxis: { tickLinesPosition: 'Inside' }}});</pre>	<pre>Property:tickPositionlet chart: Chart = new Chart({ primaryXAxis: { tickPosition: 'Inside' }});chart.appendTo('#chart');</pre>
valueType of primaryX Axis	<pre>Property:valueType\$("#container").ejChart({ primaryXAxis: { valueType: 'DateTime' }});</pre>	<pre>Property:valueTypelet chart: Chart = new Chart({ primaryXAxis: { valueType: 'DateTime' }});chart.appendTo('#chart');</pre>

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 prim	Property:multiLevelLabels\$("#container").ejChart({ primaryXAxis: { multiLevelLabels: [] } });	Property:multiLevelLabelslet chart: Chart = new Chart({ primaryXAxis: { stripLines: [] } });chart.appendTo('#chart');

aryX Axis																				
skele ton for an axes	Not Applicable	Property:skeleton <pre>let chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }] });chart.appendTo('#chart');</pre>																		
skele ton type for an axes	Not Applicable	Property:skeletonType <pre>let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }] });chart.appendTo('#chart');</pre> ## primaryYAxis <table> <thead> <tr> <th>Behaviour</th> <th>API in Essential JS 1</th> <th>API in Essential JS 2</th> </tr> </thead> <tbody> <tr> <td>Alternate grid band</td> <td> Property:alternateGridBand <pre>\$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });</pre> </td> <td>Not applicable</td> </tr> <tr> <td>Axis line cross value</td> <td> Property:crossesAt <pre>\$("#container").ejChart({ primaryYAxis: { crossesAt: 0 } });</pre> </td> <td> Property:crossesAt <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 } });chart.append To('#chart');</pre> </td> </tr> <tr> <td>axis name with which the axis line has to be crossed</td> <td> Property:crossesInAxis <pre>\$("#containe r").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });</pre> </td> <td> Property:crossesInAxis <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: ' ' } });chart.appendT o('#chart');</pre> </td> </tr> <tr> <td>axis elements placed with axis line</td> <td> Property:showNextToAxisLine <pre>\$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true } });</pre> </td> <td> Property:placeNextToAxisLine <pre>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLi ne: ' ' } });chart.appendT o('#chart');</pre> </td> </tr> <tr> <td>axis line style</td> <td> Property:axisLine.color <pre>\$("#containe r").ejChart({ primaryYAxis: { axisLine: { color : 'red' } } });</pre> </td> <td> Property:lineStyle.color <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } }</pre> </td> </tr> </tbody> </table>	Behaviour	API in Essential JS 1	API in Essential JS 2	Alternate grid band	Property:alternateGridBand <pre>\$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });</pre>	Not applicable	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.append To('#chart');</pre>	axis name with which the axis line has to be crossed	Property:crossesInAxis <pre>\$("#containe r").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });</pre>	Property:crossesInAxis <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: ' ' } });chart.appendT o('#chart');</pre>	axis elements placed with axis line	Property:showNextToAxisLine <pre>\$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true } });</pre>	Property:placeNextToAxisLine <pre>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLi ne: ' ' } });chart.appendT o('#chart');</pre>	axis line style	Property:axisLine.color <pre>\$("#containe r").ejChart({ primaryYAxis: { axisLine: { color : 'red' } } });</pre>	Property:lineStyle.color <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } }</pre>
Behaviour	API in Essential JS 1	API in Essential JS 2																		
Alternate grid band	Property:alternateGridBand <pre>\$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' } } } });</pre>	Not applicable																		
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.append To('#chart');</pre>																		
axis name with which the axis line has to be crossed	Property:crossesInAxis <pre>\$("#containe r").ejChart({ primaryYAxis: { crossesInAxis: ' ' } });</pre>	Property:crossesInAxis <pre>let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: ' ' } });chart.appendT o('#chart');</pre>																		
axis elements placed with axis line	Property:showNextToAxisLine <pre>\$("#container").ejChart({ primaryYAxis: { showNextToAxisLine : true } });</pre>	Property:placeNextToAxisLine <pre>let chart: Chart = new Chart({ primaryYAxis: { placeNextToAxisLi ne: ' ' } });chart.appendT o('#chart');</pre>																		
axis line style	Property:axisLine.color <pre>\$("#containe r").ejChart({ primaryYAxis: { axisLine: { color : 'red' } } });</pre>	Property:lineStyle.color <pre>let chart: Chart = new Chart({ primaryYAxis: { lineStyle: { color: 'black' } }</pre>																		

		<pre> });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: { width: 3 } });chart.appendT o('#chart'); Property:columnIndex1 et chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 });chart.appendT o('#chart'); Property:spanlet chart: Chart = new Chart({ primaryYAxis: { span: 2 });chart.appendT o('#chart'); </pre>
	<p>axis line dashArray</p>	<pre> Property:axisLine.color\$("#containe r").ejChart({ primaryYAxis: { axisLine: { dashArray : '10, 5' } }); </pre>
	<p>Offset for axis</p>	<pre> Property:axisLine.offset\$("#containe r").ejChart({ primaryYAxis: { axisLine: { offset : 10 } }); </pre>
	<p>Visible of an axis</p>	<pre> Property:axisLine.offset\$("#containe r").ejChart({ primaryYAxis: { axisLine: { visible : false } }); </pre>
	<p>Width of an axis</p>	<pre> Property:axisLine.width\$("#containe r").ejChart({ primaryYAxis: { axisLine: { width : 2 } }); </pre>
	<p>Column index of an axis</p>	<pre> Property:columnIndex\$("#container ").ejChart({ primaryYAxis: { columnIndex: 2 } }); </pre>
	<p>span of an axis to place horizontall y or vertically</p>	<pre> Property:columnSpan\$("#container ").ejChart({ primaryYAxis: { columnIndex: 2 } }); </pre>

		<p>Crosshair label of an axis</p> <p>Property:crossHairLabel.visible\$("#container").ejChart({ primaryYAxis: { crossHairLabel: { visible: true }}});</p>	<p>Property:crossHairTooltip.enablelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { enable: true }}});chart.appendTo('#chart');</p>
		<p>Crosshair label color of an axis</p> <p>Not applicable</p>	<p>Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { fill: 'red' }}});chart.appendTo('#chart');</p>
		<p>Crosshair label text style</p> <p>Not applicable</p>	<p>Property:crossHairTooltip.textStylelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { textStyle: { } }}});chart.appendTo('#chart');</p>
		<p>Desired interval count for primaryYAxis</p> <p>Property:desiredIntervals\$("#container").ejChart({ primaryYAxis: { desiredIntervals: 4 }});</p>	<p>Property:desiredIntervalslet chart: Chart = new Chart({ primaryYAxis: { desiredIntervals: 4 }});chart.appendTo('#chart');</p>
		<p>Edges primaryYAxis</p> <p>Property:edgeLabelPlacement\$("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' }});</p>	<p>Property:edgeLabelPlacementlet chart: Chart = new Chart({ primaryYAxis: { edgeLabelPlacement: 'Shift' }}});chart.appendTo('#chart');</p>
		<p>Enables trim for axis labels</p> <p>Property:enableTrim\$("#container").ejChart({ primaryYAxis: { enableTrim: true }});</p>	<p>Property:enableTrimlet chart: Chart = new Chart({ primaryYAxis: { enableTrim: true</p>

		<pre> });chart.appendT o('#chart'); </pre>
	<p>Specifies the interval of the axis according to the zoomed data of the chart</p>	<p>Property:enableAutoIntervalOnZooming</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendT o('#chart'); </pre>
	<p>Specifies the interval of the axis according to the zoomed data of the chart</p>	<p>Property:enableAutoIntervalOnZooming</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendT o('#chart'); </pre>
	<p>Font style for primaryYAxis</p>	<p>Property:fontStyle</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12} });chart.appendT o('#chart'); </pre>
	<p>Indexed for category axis</p>	<p>Property:isIndexed</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendT o('#chart'); </pre>
	<p>Interval type for date time axis</p>	<p>Property:intervalType</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto' }});chart.appendT o('#chart'); </pre>
	<p>Inversed axis</p>	<p>Property:isInversed</p> <pre> let chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendT o('#chart'); </pre>

		<pre> });chart.appendT o('#chart'); Property:labelFormat1 et chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' });chart.appendT o('#chart'); Property:labelIntersect Actionlet chart: Chart = new Chart({ primaryYAxis: { labelIntersectAct ion: 'Trim' });chart.appendT o('#chart'); Property:labelPosition1 et chart: Chart = new Chart({ primaryYAxis: { labelPosition: 'Inside' });chart.appendT o('#chart'); Property:labelPlacemen tlet chart: Chart = new Chart({ primaryYAxis: { labelPlacement: 'OnTicks' });chart.appendT o('#chart'); Property:alignment\$("#container") .ejChart({ primaryYAxis: { alignment: 'center' });}); Property:labelRotation let chart: Chart = new Chart({ primaryYAxis: { labelRotation: 45 });chart.appendT o('#chart'); Property:logBase\$("#container").e jChart({ primaryYAxis: { logBase: 10 });}); </pre>
	Custom label format	<pre> Property:labelFormat\$("#container") .ejChart({ primaryYAxis: { labelFormat: '{value}K' });}); </pre>
	labelIntersectAction	<pre> Property:labelIntersectAction\$("#con tainer").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' });}); </pre>
	labelPosition	<pre> Property:labelPosition\$("#container") .ejChart({ primaryYAxis: { labelPosition: 'inside' });}); </pre>
	labelPlacement for category axis	<pre> Property:labelPlacement\$("#contain er").ejChart({ primaryYAxis: { labelPlacement: 'onTicks' });}); </pre>
	Axis label alignment	<pre> Property:alignment\$("#container") .ejChart({ primaryYAxis: { alignment: 'center' });}); </pre>
	Rotation of axis labels	<pre> Property:labelRotation\$("#containe r").ejChart({ primaryYAxis: { labelRotation: 45 });}); </pre>
	Log base value for	<pre> Property:logBase\$("#container").e jChart({ primaryYAxis: { logBase: 10 });}); </pre>

		<pre> logarithmic axis logBase: 10 });chart.appendT o('#chart');</pre>	
	Major grid line	<pre> Property:majorGridLines.visible\$("#co ntainer").ejChart({ primaryYAxis: { majorGridLines: { visible: true} }));</pre>	Not Applicable
	Width of MajorGrid Lines	<pre> Property:majorGridLines.width\$("#co ntainer").ejChart({ primaryYAxis: { majorGridLines: { width: 2} }});</pre>	<pre> Property:majorGridLine s.widthlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { width: 2} }});chart.appendT o('#chart');</pre>
	Color of MajorGrid Lines	<pre> Property:majorGridLines.color\$("#con tainer").ejChart({ primaryYAxis: { majorGridLines: { color: 'black' } }));</pre>	<pre> Property:majorGridLine s.colorlet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { color: 'black' } }});chart.appendT o('#chart');</pre>
	DashArray of MajorGrid Lines	<pre> Property:majorGridLines.dashArray\$("# container").ejChart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } }));</pre>	<pre> Property:majorGridLine s.dashArraylet chart: Chart = new Chart({ primaryYAxis: { majorGridLines: { dashArray: 'black' } }});chart.appendT o('#chart');</pre>
	Opacity of major grid line	<pre> Property:majorGridLines.opacity\$("#c ontainer").ejChart({ primaryYAxis: { majorGridLines: { opacity: true} }));</pre>	Not Applicable
	Major Tick line	<pre> Property:majorTickLines.visible\$("#co ntainer").ejChart({ primaryYAxis: { majorTickLines: { visible: true} }));</pre>	Not Applicable

		<p>Width of Major Tick Lines</p> <p>Property: <code>majorTickLines.width</code> \$ (" # container "). ejChart ({ primaryYAxis: { majorTickLines: { width: 2 } } });</p>	<p>Property: <code>majorTickLines.width</code> let chart: Chart = new Chart ({ primaryYAxis: { majorTickLines: { width: 2 } } }); chart.appendTo (' # chart ');</p>
		<p>Height of Major Tick Lines</p> <p>Property: <code>majorTickLines.size</code> \$ (" # container "). ejChart ({ primaryYAxis: { majorTickLines: { size: 2 } } });</p>	<p>Property: <code>majorTickLines.size</code> let chart: Chart = new Chart ({ primaryYAxis: { majorTickLines: { height: 2 } } }); chart.appendTo (' # chart ');</p>
		<p>Color of Major Tick Lines</p> <p>Property: <code>majorTickLines.color</code> \$ (" # container "). ejChart ({ primaryYAxis: { majorTickLines: { color: 'black' } } });</p>	<p>Property: <code>majorTickLines.color</code> let chart: Chart = new Chart ({ primaryYAxis: { majorTickLines: { color: 'black' } } }); chart.appendTo (' # chart ');</p>
		<p>Opacity of major Tick line</p> <p>Property: <code>majorTickLines.opacity</code> \$ (" # container "). ejChart ({ primaryYAxis: { majorTickLines: { opacity: true } } });</p>	Not Applicable
		<p>maximum labels of primaryYAxis</p> <p>Property: <code>maximumLabels</code> \$ (" # container "). ejChart ({ primaryYAxis: { maximumLabels: 5 } });</p>	<p>Property: <code>maximumLabels</code> let chart: Chart = new Chart ({ primaryYAxis: { maximumLabels: 4 } }); chart.appendTo (' # chart ');</p>
		<p>maximum labels width of primaryYAxis to trim</p> <p>Property: <code>maximumLabelWidth</code> \$ (" # container "). ejChart ({ primaryYAxis: { maximumLabelWidth: 40 } });</p>	<p>Property: <code>maximumLabelWidth</code> let chart: Chart = new Chart ({ primaryYAxis: { maximumLabelWidth: 4 } }); chart.appendTo (' # chart ');</p>

		<p>minor grid line</p> <p>Property:minorGridLines.visible\$("#container").ejChart({ primaryYAxis: { minorGridLines: { visible: true} }));</p>	Not Applicable
		<p>Width of minorGrid Lines</p> <p>Property:minorGridLines.width\$("#container").ejChart({ primaryYAxis: { minorGridLines: { width: 2} }});</p>	<p>Property:minorGridLines.width\$let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { width: 2} }});chart.appendT o('#chart');</p>
		<p>Color of minorGrid Lines</p> <p>Property:minorGridLines.color\$("#container").ejChart({ primaryYAxis: { minorGridLines: { color: 'black' } }));</p>	<p>Property:minorGridLines.color\$let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { color: 'black' } }});chart.appendT o('#chart');</p>
		<p>DashArray of minorGrid Lines</p> <p>Property:minorGridLines.dashArray\$("#container").ejChart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } }));</p>	<p>Property:minorGridLines.dashArray\$let chart: Chart = new Chart({ primaryYAxis: { minorGridLines: { dashArray: 'black' } }});chart.appendT o('#chart');</p>
		<p>Opacity of minor grid line</p> <p>Property:minorGridLines.opacity\$("#container").ejChart({ primaryYAxis: { minorGridLines: { opacity: true} }));</p>	Not Applicable
		<p>minor Tick line</p> <p>Property:minorTickLines.visible\$("#container").ejChart({ primaryYAxis: { minorTickLines: { visible: true} }));</p>	Not Applicable
		<p>Width of minorTick Lines</p> <p>Property:minorTickLines.width\$("#container").ejChart({ primaryYAxis: { minorTickLines: { width: 2} }});</p>	<p>Property:minorTickLines.width\$let chart: Chart = new Chart({ primaryYAxis: { minorTickLines: {</p>

		<pre>width: 2} });chart.appendT o('#chart');</pre>
Height of minor Tick Lines	<pre>Property:minorTickLines.size\$("#cont ainer").ejChart({ primaryYAxis: { minorTickLines: { size: 2} }});</pre>	<pre>Property:minorTickLine s.heightlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { height: 2} });chart.appendT o('#chart');</pre>
Color of minor Tick Lines	<pre>Property:minorTickLines.color\$("#con tainer").ejChart({ primaryYAxis: { minorTickLines: { color: 'black' } }));</pre>	<pre>Property:minorTickLine s.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTickLines: { color: 'black' } });chart.appendT o('#chart');</pre>
Opacity of minor Tick line	<pre>Property:minorTickLines.opacity\$("#c ontainer").ejChart({ primaryYAxis: { minorTickLines: { opacity: true} }));</pre>	Not Applicable
Minor ticks per interval of primaryYA xis	<pre>Property:minorTicksPerInterval\$("#co ntainer").ejChart({ primaryYAxis: { minorTicksPerInterval: 4 }));</pre>	<pre>Property:minorTickLine s.colorlet chart: Chart = new Chart({ primaryYAxis: { minorTicksPerInte rval: 4 });chart.appendT o('#chart');</pre>
name of the primaryYA xis	<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>
Orientatio n of primaryYA xis	<pre>Property:orientation\$("#container").ejChart({ primaryYAxis: { orientation: 'Vertical' }));</pre>	Not Applicable

		<p>Plot offset for primaryYAxis</p> <p>Property:plotOffset <pre>let chart: Chart = new Chart({ primaryYAxis: { plotOffset: 0 } }); chart.appendT o('#chart');</pre> </p> <p>minimum for primaryYAxis</p> <p>Property:range.minimum <pre>let chart: Chart = new Chart({ primaryYAxis: { range: { minimum: 10 } } }); chart.appendT o('#chart');</pre> </p> <p>maximum for primaryYAxis</p> <p>Property:range.maximum <pre>let chart: Chart = new Chart({ primaryYAxis: { range: { maximum: 10 } } }); chart.appendT o('#chart');</pre> </p> <p>interval for primaryYAxis</p> <p>Property:range.interval <pre>let chart: Chart = new Chart({ primaryYAxis: { range: { interval: 1 } } }); chart.appendT o('#chart');</pre> </p> <p>RangePadding for primaryYAxis</p> <p>Property:rangePadding <pre>let chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' } }); chart.appendT o('#chart');</pre> </p> <p>Rounding Places in primaryYAxis</p> <p>Property:roundingPlaces <pre>let chart: Chart = new Chart({ primaryYAxis: { roundingPlaces: 3 } }); chart.appendT o('#chart');</pre> </p> <p>ScrollBar settings of primaryYAxis</p> <p>Property:scrollbarSettings <pre>let chart: Chart = new Chart({ primaryYAxis: { scrollbarSettings : { } } });</pre> </p>	<p>Property:plotOffset <pre>let chart: Chart = new Chart({ primaryYAxis: { plotOffset: 0 } }); chart.appendT o('#chart');</pre> </p> <p>Property:minimum <pre>let chart: Chart = new Chart({ primaryYAxis: { minimum: 23 } }); chart.appendT o('#chart');</pre> </p> <p>Property:maximum <pre>let chart: Chart = new Chart({ primaryYAxis: { maximum: 23 } }); chart.appendT o('#chart');</pre> </p> <p>Property:interval <pre>let chart: Chart = new Chart({ primaryYAxis: { interval: 2 } }); chart.appendT o('#chart');</pre> </p> <p>Property:rangePadding <pre>let chart: Chart = new Chart({ primaryYAxis: { rangePadding: 'None' } }); chart.appendT o('#chart');</pre> </p> <p>Property:labelFormat <pre>let chart: Chart = new Chart({ primaryYAxis: { labelFormat: 'n3' } }); chart.appendT o('#chart');</pre> </p> <p>Not Applicable</p>
--	--	--	--

		TickPosition in primaryYAxis Property:tickLinesPosition <pre> Property:tickLinesPosition\$("#container").ejChart({ primaryYAxis: { tickLinesPosition: 'Inside' }}); </pre>	Property:tickPosition <pre> let chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' }});chart.appendTo('#chart'); </pre>
		valueType of primaryYAxis Property:valueType <pre> Property:valueType\$("#container").ejChart({ primaryYAxis: { valueType: 'DateTime' }}); </pre>	Property:valueType <pre> let chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' }});chart.appendTo('#chart'); </pre>
		visible of primaryYAxis Property:visible <pre> Property:visible\$("#container").ejChart({ primaryYAxis: { visible: true }}); </pre>	Property:visible <pre> let chart: Chart = new Chart({ primaryYAxis: { visible: true }});chart.appendTo('#chart'); </pre>
		zoomFactor of primaryYAxis Property:zoomFactor <pre> Property:zoomFactor\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 }}); </pre>	Property:zoomFactor <pre> let chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 }});chart.appendTo('#chart'); </pre>
		zoomPosition of primaryYAxis Property:zoomPosition <pre> Property:zoomPosition\$("#container").ejChart({ primaryYAxis: { zoomPosition: 0.3 }}); </pre>	Property:zoomPosition <pre> let chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 }});chart.appendTo('#chart'); </pre>
		labelBorder of primaryYAxis Property:labelBorder <pre> Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2 } }}); </pre>	Property:border <pre> let chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } }});chart.appendTo('#chart'); </pre>
		title of primaryYAxis Property:title.text <pre> Property:title.text\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } }}); </pre>	Property:title <pre> let chart: Chart = new Chart({ primaryYAxis: { title: 'Chart </pre>

		<pre> title' });chart.appendT o('#chart'); Property:stripLineslet chart: Chart = new Chart({ primaryYAxis: { stripLines: [] });chart.appendT o('#chart'); Property:multiLevelLab elslet chart: Chart = new Chart({ primaryYAxis: { stripLines: [] });chart.appendT o('#chart'); Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]);chart.append To('#chart'); Property:skeletonType let chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]);chart.append To('#chart'); ## Axes Beha API in Essential JS 1 API in Essential JS 2 viour Property:alternateG ridBand Alter \$("#container") nate .ejChart({ grid axes: [{ Not applicable band nd: { even: { fill: 'red }}}}]); </pre>
--	--	---

		<p>Property:crossesAt</p> <p>Axis line cross value</p> <pre> \$("#container") .ejChart({ axes: [{ crossesAt: 0 }] }); </pre> <p>Property:crossesAt</p> <pre> let chart: Chart = new Chart({ axes: [{ crossesAt: 4 }] });chart.appendTo('#chart'); </pre> <p>axis nam e</p> <p>Property:crossesInA</p> <p>whic h the axis line has</p> <pre> \$("#container") .ejChart({ axes: [{ crossesInAxis: ' ' }] }); </pre> <p>Property:crossesInAxis</p> <pre> let chart: Chart = new Chart({ axes: [{ crossesInAxis: ' ' }] });chart.appendTo('#chart'); </pre> <p>to be cross ed</p> <p>Property:showNext</p> <p>elem ents place d</p> <pre> \$("#container") .ejChart({ axes: [{ showNextToAxisL ine : true }] }); </pre> <p>Property:placeNextToAxisLine</p> <pre> let chart: Chart = new Chart({ axes: [{ placeNextToAxisLine: ' ' }] });chart.appendTo('#chart'); </pre> <p>with axis line</p> <p>Property:axisLine.c</p> <p>axis line style</p> <pre> \$("#containe r").ejChart({ axes: [{ axisLine: { color : 'red' } }] }); </pre> <p>Property:axisLine.c</p> <pre> let chart: Chart = new Chart({ axes: [{ lineStyle: { color: 'black' } }] });chart.appendTo('#chart'); </pre> <p>Property:axisLine.c</p> <p>axis line dash Array</p> <pre> \$("#containe r").ejChart({ axes: [{ axisLine: { dashArray : '10, 5' } }] }); </pre> <p>Property:axisLine.c</p> <pre> let chart: Chart = new Chart({ axes: [{ lineStyle: { dashArray: '10, 5' } }] });chart.appendTo('#chart'); </pre> <p>Property:axisLine.of</p> <p>Offse t for axis</p> <pre> \$("#containe r").ejChart({ axes: [{ axisLine: { </pre> <p>Property:plotOffset</p> <pre> let chart: Chart = new Chart({ axes: [{ plotOffset: 10 }] });chart.appendTo('#chart'); </pre>
--	--	--

		<pre> offset : 10 } }}}); Property:axisLine.offset Visible of an axis fset\$("#container").ejChart({ axes: [{ axisLine: { visible : false } }] }); Property:axisLine.width Width of an axis idth\$("#container").ejChart({ axes: [{ axisLine: { width : 2 } } }] }); Property:columnIndex Column index of an axis ex\$("#container").ejChart({ axes: [{ columnIndex: 2 } }] }); Property:columnSpan span of an axis to place horizontal or vertically an\$("#container").ejChart({ axes: [{ columnIndex: 2 } }] }); Property:crossHairLabel Cross hair label of an axis abel.visible\$("#container").ejChart({ axes: [{ crossHairLabel: { visible: true } }] }); Property:crossHairLabelColor Cross hair label color of an axis Not applicable </pre>	<pre> Property:visible let chart: Chart = new Chart({ axes: [{ visible: false }] });chart.appendTo('#chart'); Property:lineStyle.width let chart: Chart = new Chart({ axes: [{ lineStyle: { width: 3 } }] });chart.appendTo('#chart'); Property:columnIndex let chart: Chart = new Chart({ axes: [{ columnIndex: 2 }] });chart.appendTo('#chart'); Property:span let chart: Chart = new Chart({ axes: [{ span: 2 }] });chart.appendTo('#chart'); Property:crossHairTooltip.enable let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { enable: true } }] });chart.appendTo('#chart'); Property:crossHairTooltip.fill let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { fill: 'red' } }] });chart.appendTo('#chart'); </pre>
--	--	---	---

		<p>Cross hair label text style</p> <p>Property:crossHairTooltip.textStyle</p> <pre>let chart: Chart = new Chart({ axes: [{ crossHairTooltip: { textStyle: { } } }] });chart.appendTo('#chart');</pre>
		<p>Desired interval for primary Y Axis</p> <p>Property:desiredIntervals</p> <pre>let chart: Chart = new Chart({ axes: [{ desiredIntervals: 4 }] });chart.appendTo('#chart');</pre>
		<p>Edge Label Placement</p> <p>Property:edgeLabelPlacement</p> <pre>let chart: Chart = new Chart({ axes: [{ edgeLabelPlacement: 'Shift' }] });chart.appendTo('#chart');</pre>
		<p>Enables trim for axis labels</p> <p>Property:enableTrim</p> <pre>let chart: Chart = new Chart({ axes: [{ enableTrim: true }] });chart.appendTo('#chart');</pre>
		<p>Specifies the interval of the axis according to the zoomed data of the chart</p> <p>Property:enableAutoIntervalOnZooming</p> <pre>let chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: true }] });chart.appendTo('#chart');</pre>

		<p>Specifies the interval value of the Property:enableAutoIntervalOnZooming according to the zooming of the chart</p> <pre> g\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] }); </pre> <p>Font style for primary Y Axis</p> <pre> Property:font\$(\$("#container").ejChart({ axes: [{ font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } }] }); </pre> <p>Indexed for category axis</p> <pre> Property:isIndexed\$(\$("#container").ejChart({ axes: [{ isIndexed: true }] }); </pre> <p>Interval type for date time axis</p> <pre> Property:intervalType\$(\$("#container").ejChart({ axes: [{ intervalType: 'Auto' }] }); </pre> <p>Inversed axis</p> <pre> Property:isInversed\$(\$("#container").ejChart({ axes: [{ isInversed: true }] }); </pre>
--	--	--

		<pre> isInversed: true]]]); Property:labelForm Cust at\$("#container om ").ejChart({ label axes: [{ form labelFormat: at '{value}K']]]); Property:labelInters label ectAction\$("#con nters tainer").ejChar ectA t({ axes: [{ ction labelIntersectA tion ction: 'trim']]]); Property:labelPositi label on\$("#container Positi ").ejChart({ on axes: [{ labelPosition: 'inside']]]); label Place Property:labelPlace ment ment\$("#contain for er").ejChart({ categ axes: [{ ory labelPlacement: axis 'onTicks']]]); Property:alignment Axis \$\$("#container") label .ejChart({ align axes: [{ ment alignment: 'center']]]); Rotat Property:labelRotat ion ion\$("#containe of r").ejChart({ axis axes: [{ label labelRotation: s 45]]]); Log Property:logBase\$(base "#container").e value jChart({ axes: for [{ logBase: 10 logar]]]); </pre>	<pre> Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: '{value}K' }}]);chart.appendTo('#chart'); Property:labelIntersectActionlet chart: Chart = new Chart({ axes: [{ labelIntersectAction: 'Trim' }}]);chart.appendTo('#chart'); Property:labelPositionlet chart: Chart = new Chart({ axes: [{ labelPosition: 'Inside' }}]);chart.appendTo('#chart'); Property:labelPlacementlet chart: Chart = new Chart({ axes: [{ labelPlacement: 'OnTicks' }}]);chart.appendTo('#chart'); Not Applicable Property:labelRotationlet chart: Chart = new Chart({ axes: [{ labelRotation: 45 }]]);chart.appendTo('#chart'); Property:labelRotationlet chart: Chart = new Chart({ axes: [{ logBase: 10 [{ logBase: 10 }]]];chart.appendTo('#chart'); }}]); </pre>
--	--	---	--

		<p>ithmi c axis</p> <p>Property:majorGrid</p> <p>Lines.visible\$("#co ntainer").ejCha rt({ axes: [{ Not Applicable majorGridLines: { visible: true}]});</p> <p>Property:majorGrid</p> <p>Width h of Lines.width\$("#co ntainer").ejCha rt({ axes: [{ Property:majorGridLines.widthlet chart: Chart = new Chart({ axes: [{ majorGridLines: { width: 2} }]});chart.appendTo('#chart');</p> <p>Major r Grid majorGridLines: { width: 2} Lines }]});</p> <p>Property:majorGrid</p> <p>Color of Lines.color\$("#con tainer").ejChar t({ axes: [{ Property:majorGridLines.colorlet chart: Chart = new Chart({ axes: [{ majorGridLines: { color: 'black' } }]});chart.appendTo('#chart');</p> <p>Major r Grid majorGridLines: { color: 'black' }]});</p> <p>Property:majorGrid</p> <p>Dash Array Lines.dashArray\$(" #container").ej of Chart({ axes: [Property:majorGridLines.dashArraylet chart: Chart = new Chart({ axes: [{ majorGridLines: { dashArray: 'black' }]});chart.appendTo('#chart');</p> <p>Major r Grid { dashArray: 'black' }]});</p> <p>Property:majorGrid</p> <p>Opac ity of Lines.opacity\$("#c ontainer").ejCh major r grid art({ axes: [{ Not Applicable majorGridLines: { opacity: true}]});</p> <p>Property:majorTick</p> <p>Lines.visible\$("#co ntainer").ejCha rt({ axes: [{ Not Applicable majorTickLines: { visible: true}]});</p> <p>Property:majorTick</p> <p>Width h of Property:majorTick Property:majorTickLines.widthlet chart: Chart = new Chart({ axes: [{</p>
--	--	---

		<pre> MajorTickLines: { width: 2} rt({ axes: [{ }]});chart.appendTo('#chart'); majorTickLines: { width: 2} }]]); Property:majorTick Height of majorTickLines.size\$("#container").ejChart MajorTickLines: { height: 2} rt({ axes: [{ majorTickLines: { height: 2} majorTickLines: { size: 2} }]]);chart.appendTo('#chart'); }]]); Property:majorTick Color of majorTickLines.color\$("#container").ejChart MajorTickLines: { color: 'black' } rt({ axes: [{ majorTickLines: { color: 'black' } majorTickLines: { color: 'black' } }]]);chart.appendTo('#chart'); }]]); Property:majorTick Opacity of majorTickLines.opacity\$("#container").ejChart MajorTickLines: { opacity: true} rt({ axes: [{ majorTickLines: { opacity: true} majorTickLines: { opacity: true} }]]);chart.appendTo('#chart'); }]]); Property:majorTick Maximum of primaryY Axis Property:maximum Labels\$("#container").ejChart rt({ axes: [{ maximumLabels: 4 majorTickLines: { maximumLabels: 5 } }]]);chart.appendTo('#chart'); }]]); Property:maximum Maximum of primaryY Axis Property:maximum Width of primaryY Axis Property:maximumLabelWidth rt({ axes: [{ maximumLabelWidth: 4 majorTickLines: { maximumLabelWidth: 40 } }]]);chart.appendTo('#chart'); }]]); </pre>
--	--	---

		<pre> Property:minorGrid Lines.visible\$("#co ntainer").ejCha rt({ axes: [{ Not Applicable minorGridLines: { visible: true} }]]}); </pre>
Width of minor grid Lines		<pre> Property:minorGrid Lines.width\$("#co ntainer").ejCha rt({ axes: [{ Property:minorGridLines.widthlet chart: minorGridLines: { width: 2} Chart = new Chart({ axes: [{ { width: 2} }]]});chart.appendTo('#chart'); }]]}); </pre>
Color of minor grid Lines		<pre> Property:minorGrid Lines.color\$("#con tainer").ejChar t({ axes: [{ Property:minorGridLines.colorlet chart: minorGridLines: { color: 'black' } Chart = new Chart({ axes: [{ { color: 'black' } }]]});chart.appendTo('#chart'); }]]}); </pre>
Dash Array of minor grid Lines		<pre> Property:minorGrid Lines.dashArray\$("# container").ej Chart({ axes: [Property:minorGridLines.dashArraylet chart: { minorGridLines: { dashArray: 'black' Chart = new Chart({ axes: [{ { dashArray: 'black' } }]]});chart.appendTo('#chart'); } }]]}); </pre>
Opacity of minor grid line		<pre> Property:minorGrid Lines.opacity\$("#c ontainer").ejCh art({ axes: [{ Not Applicable minorGridLines: { opacity: true} }]]}); </pre>
minor grid Tick line		<pre> Property:minorTick Lines.visible\$("#co ntainer").ejCha rt({ axes: [{ Not Applicable minorTickLines: { visible: true} }]]}); </pre>
Width of minor grid Tick Lines		<pre> Property:minorTick Lines.width\$("#co ntainer").ejCha rt({ axes: [{ Property:minorTickLines.widthlet chart: minorTickLines: { width: 2} Chart = new Chart({ axes: [{ }]]});chart.appendTo('#chart'); }]]}); </pre>

		<pre> rTick { width: 2} Lines }]]); Property:minorTick Height of Lines.size\$("#container").ejChart minorTickLines: { height: 2} rTick minorTickLines: { height: 2} Lines }]]); Property:minorTick Color of Lines.color\$("#container").ejChart minorTickLines: { color: 'black' } Lines { color: 'black' }]]]); Property:minorTick Opacity of Lines.opacity\$("#container").ejChart minorTickLines: { opacity: true }]]]); Minor ticks per interval of primaryY Axis Property:minorTick sPerInterval\$("#container").ejChart minorTicksPerInterval: 4 }]]); Property:minorTickLines.color let chart: Chart = new Chart({ axes: [{ minorTickLines: { height: 2} }]]);chart.appendTo('#chart'); Property:minorTickLines.color let chart: Chart = new Chart({ axes: [{ minorTickLines: { color: 'black' } }]]);chart.appendTo('#chart'); Property:minorTick Lines.opacity\$("#container").ejChart minorTickLines: { opacity: true }]]]); Not Applicable Property:minorTick Property:minorTickLines.color let chart: Chart = new Chart({ axes: [{ minorTicksPerInterval: 4 }]]);chart.appendTo('#chart'); Property:name\$("#container").ejChart Property:name let chart: Chart = new Chart({ axes: [{ name: 'primaryYAxis' }]]);chart.appendTo('#chart'); Property:orientation\$("#container").ejChart Not Applicable orientation: 'Vertical' }]]); </pre>
--	--	---

		<p>Plot offset Property:plotOffset <pre>\$("#container").ejChart({ axes: [{ plotOffset: 0 }] });</pre></p> <p>Primary Y Axis Property:plotOffset <pre>let chart: Chart = new Chart({ axes: [{ plotOffset: 0 }] }); chart.appendTo('#chart');</pre></p> <p>Minimum Property:range.minimum <pre>\$("#container").ejChart({ axes: [{ range: { minimum: 10 } }] });</pre></p> <p>Maximum Property:range.maximum <pre>\$("#container").ejChart({ axes: [{ range: { maximum: 10 } }] });</pre></p> <p>Interval Property:range.interval <pre>\$("#container").ejChart({ axes: [{ range: { interval: 1 } }] });</pre></p> <p>Range Padding Property:rangePadding <pre>\$("#container").ejChart({ axes: [{ rangePadding: 'None' }] });</pre></p> <p>Rounding Places Property:roundingPlaces <pre>\$("#container").ejChart({ axes: [{ roundingPlaces: 3 }] });</pre></p> <p>Scrollbar Settings Property:scrollbarSettings <pre>\$("#container").ejChart({ axes: [{</pre></p> <p>Label Format Property:labelFormat <pre>let chart: Chart = new Chart({ axes: [{ labelFormat: 'n3' }] }); chart.appendTo('#chart');</pre></p> <p>Not Applicable</p>
--	--	---

		<pre> of scrollbarSettin prim gs : { }]]]); aryY Axis TickP Property:tickLinesP ositi osition\$("#conta Property:tickPositionlet chart: Chart = on in iner").ejChart(new Chart({ axes: [{ tickPosition: prim { axes: [{ 'Inside' aryY tickLinesPositi }]]);chart.appendTo('#chart'); Axis on: 'Inside' }]]); value Property:valueType Type \$("#container") Property:valueTypelet chart: Chart = new of .ejChart({ Chart({ axes: [{ valueType: prim axes: [{ 'DateTime' aryY valueType: }]]);chart.appendTo('#chart'); Axis 'DateTime' }]]); visibl Property:visible\$ (" e of #container").ej Property:visiblelet chart: Chart = new prim Chart({ axes: [Chart({ axes: [{ visible: true aryY { visible: true }]]);chart.appendTo('#chart'); Axis }]]); zoo mFac Property:zoomFact tor or\$("#container Property:zoomFactorlet chart: Chart = of ").ejChart({ new Chart({ axes: [{ zoomFactor: prim axes: [{ 0.3 }]]);chart.appendTo('#chart'); aryY zoomFactor: 0.3 Axis }]]); zoo mPos Property:zoomPosit ition ion\$("#containe Property:zoomPositionlet chart: Chart = of r").ejChart({ new Chart({ axes: [{ zoomPosition: prim axes: [{ 0.3 }]]);chart.appendTo('#chart'); aryY zoomPosition: Axis 0.3 }]]); label Property:labelBord Bord er\$("#container Property:borderlet chart: Chart = new er of ").ejChart({ Chart({ axes: [{ border: { color: prim axes: [{ 'red', width: 3 } aryY labelBorder: { }]]);chart.appendTo('#chart'); Axis color: 'red', width: 2}]]]); </pre>
--	--	--

		<p>title Property:title.text\$ of ("#container"). Property:titlelet chart: Chart = new prim ejChart({ axes: Chart({ axes: [{ title: 'Chart aryY [{ title: { title' Axis text: 'Chart }]});chart.appendTo('#chart'); title' }]]);</p> <p>Strip Line Property:stripLine\$ of ("#container"). Property:stripLineslet chart: Chart = new prim ejChart({ axes: Chart({ axes: [{ stripLines: [] aryY [{ stripLine: }]});chart.appendTo('#chart'); Axis []]]);</p> <p>Multi Property:multiLevel level Labels\$("#contai Property:multiLevelLabelslet chart: Chart label ner").ejChart({ = new Chart({ axes: [{ stripLines: s of axes: [{ []]});chart.appendTo('#chart'); axes multiLevelLabel s: []]]);</p> <p>skele ton for Not Applicable Property:skeletonlet chart: Chart = new an Chart({ axes: [{ skeleton: 'yMd' axes }]});chart.appendTo('#chart');</p> <p>Property:skeletonTypelet chart: Chart = new Chart({ axes: [{ skeletonType: 'DateTime' }]});chart.appendTo('#chart');</p> <p>## Rows</p> <p>Beha viour API in Essential JS 1 API in Essential JS 2</p> <p>skele ton type Not Applicable Property:rowslet for rows Property:rowDefinit chart: Chart = an ions\$("#chart") new Chart({ axes in .ejChart({ rows: chart rowDefinitions: []});chart.app []}); endTo('#chart') ;</p> <p>Property:unit \$("#container") .ejChart({ unit rowDefinitions Not Applicable : [{unit : "percentage" }]});</p>
--	--	---

		<pre> Property:heightlet heig Property:rowHeight chart: Chart = ht of \$("#chart").ejC new Chart({ rows hart({ rows: [{ in rowDefinitions: height: chart [{ rowHeight: '300'}];});char '50%'}];}); t.appendTo('#ch art'); </pre>
		<pre> Property:lineColor, Property:borderle lineWidth\$("#cha t chart: Chart rt").ejChart({ = new Chart({ Line rowDefinitions: height: '300', custo [{ rowHeight: border: { miza '50%', width: 2, tion lineColor: color: 'brown', 'brown'}}];});c lineWidth: hart.appendTo('# 2}}];}); chart'); </pre>
		<pre> ## Series </pre>
	Behaviour	<pre> API in Essential JS 1 API in Essential JS 2 </pre>
		<pre> Property:bearF illColorlet chart: Property:bearFillColor\$ Chart = new bearFi ("#chart").ejChar Chart({ llColor t({ series: series: [{{bearFillColor: [{bearFillC 'red' }];}); olor: 'red' }];});chart .appendTo('# chart'); </pre>
	Border	<pre> Property:rows let chart: Property:border\$("#c Chart = new hart").ejChart({ Chart({ series: [{ series: [{ border: { color: border: { color: 'red', width: 2, color: 'red', dashArray: '10, width: 2} 5' } }];}); }];});chart .appendTo('# chart'); </pre>
	BoxPlotMode	<pre> Property:boxPlotMode Property:rows \$("#chart").ejCha let chart: rt({ series: [{ Chart = new boxPlotMode: Chart({ </pre>

		<pre> 'inclusive' }];}); </pre>	<pre> series: [{ boxPlotMode : 'Inclusive' }];});chart .appendTo('#chart'); </pre>
	<pre> Minim um radius of Bubbl e series </pre>	<pre> Property:bubbleOption s.minRadius\$("#char t").ejChart({ series: [{ bubbleOptions: { minRadius: 2} }];}); </pre>	<pre> Property:minR adiuslet chart: Chart = new Chart({ series: [{ minRadius: 2 }];});chart .appendTo('#chart'); </pre>
	<pre> Maxi mum radius of Bubbl e series </pre>	<pre> Property:bubbleOption s.maxRadius\$("#char t").ejChart({ series: [{ bubbleOptions: { maxRadius: 10 } }];}); </pre>	<pre> Property:maxR adiuslet chart: Chart = new Chart({ series: [{ maxRadius: 2 }];});chart .appendTo('#chart'); </pre>
	<pre> bullFill Color </pre>	<pre> Property:bullFillColor\$ ("#chart").ejChar t({ series: [{bullFillColor: 'red' }];}); </pre>	<pre> Property:bullFil lColorlet chart: Chart = new Chart({ series: [{bullFillC olor: 'red' }];});chart .appendTo('#chart'); </pre>
	<pre> Cardin al spline tensio n for spline series </pre>	<pre> Property:cardinalSpline Tension\$("#chart") .ejChart({ series: [{ cardinalSplineTen sion: 0.5 }];}); </pre>	<pre> Property:cardi nalSplineTensi onlet chart: Chart = new Chart({ series: [{ cardinalSpl ineTension: 0.5 </pre>

		<pre> });});chart .appendTo('#chart'); </pre>
	ColumnWidth ColumnWidth for rectangle series	<pre> Property:columnWidthlet chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }];});chart .appendTo('#chart'); </pre>
	ColumnSpacing ColumnSpacing for rectangle series	<pre> Property:columnSpacinglet chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }];});chart .appendTo('#chart'); </pre>
	TopLeftRadius TopLeftRadius for rectangle series	<pre> Property:cornerRadius.topLeft let chart: Chart = new Chart({ series: [{ topLeft: 0 }];});chart .appendTo('#chart'); </pre>
	TopRightRadius TopRightRadius for rectangle series	<pre> Property:cornerRadius.topRight let chart: Chart = new Chart({ series: [{ topRight: 0 }];});chart .appendTo('#chart'); </pre>
	BottomRightRadius BottomRightRadius for rectangle series	<pre> Property:cornerRadius.bottomRight let chart: Chart = new Chart({ series: [{ bottomRight: 0 }];});chart .appendTo('#chart'); </pre>

		<pre> for bottomRight: 0 rectan });}); gle series </pre>	<pre> 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 rRadius.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: [] }]);});chart .appendTo('#chart'); </pre>
		<pre> Draw Property:drawType\$("# type #chart").ejChart(for { series: [{ Polar drawType: 'Line' series }]);}); </pre>	<pre> Property:draw Typelet chart: Chart = new Chart({ series: [{ drawType: 'Line' }]);}); </pre>

		<pre> });});chart .appendTo('#chart'); </pre>
Empty Points etting s for series	Property:emptyPointSe ttings.visible\$("#char t").ejChart({ series: [{ emptyPointSetting s: { visible: false } }]);});	Not Applicable
Empty Point Displa Y mode	Property:emptyPointSe ttings.displayMode\$(" #chart").ejChart({ series: [{ displayMode: 'gap' }]});});	Property:emptyPointSettings. displayMode let chart: Chart = new Chart({ series: [{ displayMode : 'Average' }]);});chart .appendTo('#chart');
Empty Point color	Property:emptyPointSe ttings.color\$("#chart ").ejChart({ series: [{ color: 'red' }]});});	Property:emptyPointSettings. fill let chart: Chart = new Chart({ series: [{ fill: 'red' }]);});chart .appendTo('#chart');
Empty Point Borde r	Property:emptyPointSe ttings.border\$("#cha rt").ejChart({ series: [{ emptyPointSetting s: { color: 'red', width: 2 }]});});	Property:fill let chart: Chart = new Chart({ series: [{ emptyPoints ettings: { color: 'red', width: 2 }]});});chart .appendTo('#chart');
Enabl e anima	Property:enableAnimat ion\$("#chart").ejC hart({ series: [{	Property:anima tion.enable let chart:

		<pre> tion enableAnimation: Chart = new for true]];}); series </pre>	<pre> Chart({ series: [animation: { enable: false }]);});chart .appendTo('#chart'); </pre>
		<pre> Anima Property:animationDur tion ation\$("#chart").e durati jChart({ series: on for [{ series animationDuration : 1000 }]);}); </pre>	<pre> Property:anima tion.duration1 et chart: Chart = new Chart({ series: [animation: { duration: 1000 }]);});chart .appendTo('#chart'); </pre>
		<pre> Anima tion delay Not Applicable for series </pre>	<pre> Property:anima tion.duration1 et chart: Chart = new Chart({ series: [animation: { delay: 100 }]);});chart .appendTo('#chart'); </pre>
		<pre> Drag Property:dragSettings\$ settin ("#chart").ejChar gs for t({ series: [{ series dragSettings: { mode: 'X' } }]);}); </pre>	<pre> Not Applicable </pre>
		<pre> Errorb Property:errorBarSetti ar ngs\$("#chart").ejC settin hart({ series: [{ gs for errorBarSettings: series { } }]);}); </pre>	<pre> Property:error BarSettingslet chart: Chart = new Chart({ series: [{errorBarSe tttings: { }]);}); </pre>

			Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]});	Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]});	Property:isClosed\$("#chart").ejChart({ series: [{ isClosed: true }]});
			Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]});	Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]});	Property:isStacking\$("#chart").ejChart({ series: [{ isStacking: true }]});
			Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]});	Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]});	Property:lineCap\$("#chart").ejChart({ series: [{ lineCap: 'butt' }]});
			Property:lineJoin\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]});	Property:lineJoin\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]});	Property:lineJoin\$("#chart").ejChart({ series: [{ lineJoin: 'round' }]});
			Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]});	Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]});	Property:opacity\$("#chart").ejChart({ series: [{ opacity: 0.7 }]});
			Property:outLierSetting\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20 } } }]});	Property:outLierSetting\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20 } } }]});	Property:outLierSetting\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20 } } }]});
			Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }]});	Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }]});	Property:palette\$("#chart").ejChart({ series: [{ palette: "ColorFieldName" }]});

		<pre> 'color' }};});chart .appendTo('#chart'); Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ pointColorM apping: 'color' }};});chart .appendTo('#chart'); Positive fill for water fall series Property:positiveFill\$ ("#chart").ejChart ({ series: [{ positiveFill: "red" }];}); Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ showMean: false }};});chart .appendTo('#chart'); Show average value in box and whisker series Property:showMedian \$("#chart").ejCha rt({ series: [{ showMedian: true }];}); Property:point ColorMapping1 et chart: Chart = new Chart({ series: [{ showMean: false }};});chart .appendTo('#chart'); To group the series of stacking collection. Property:stackingGrou p\$("#chart").ejCh art({ series: [{ stackingGroup: 'group' }];}); Property:stacki ngGroup1 et chart: Chart = new Chart({ series: [{ stackingGro up: 'group' }};});chart .appendTo('#chart'); Specifies the type of the series to render in chart. Property:typel et chart: Chart = new Chart({ series: [{ type: 'Line' }};});chart .appendTo('#chart'); </pre>
--	--	--

		<p>Define the visibility of the series.</p> <p>Property:visibility\$("#chart").ejChart({ series: [{ visibility: true }]});</p> <p>Property:visiblelet chart: Chart = new Chart({ series: [{ visible: true }]});chart.appendTo('#chart');</p>
	<p>Enables or disables the visibility of legend item.</p> <p>Property:visibleOnLegend\$("#chart").ejChart({ series: [{ visibleOnLegend : true }]});</p> <p>Property:toggleVisibilitylet chart: Chart = new Chart({ legendSettings: [{ toggleVisibility: true }]});chart.appendTo('#chart');</p>	
	<p>Specifies the different types of spline curve.</p> <p>Property:splineType\$("#chart").ejChart({ series: [{ splineType : 'Natural' }]});</p> <p>Property:splineTypelet chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }]});chart.appendTo('#chart');</p>	
	<p>Specifies the name of the x-axis that has to be associated with this series. Add an</p> <p>Property:xAxisName\$("#chart").ejChart({ series: [{ xAxisName : 'secondaryXAxis' }]});</p> <p>Property:xAxisNamelet chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }]});chart.appendTo('#chart');</p>	

		<p>axis instance with this name to axes collection.</p> <p>Name of the property in the data source that contains value for the series.</p> <p>Specifies the name of the y-axis that has to be associated with this series.</p> <p>Add an axis instance with this name</p>
		<p>Property:xName</p> <pre>let chart; Chart = new Chart({ series: [{ xName: 'x' }]; }); chart .append('' #chart);</pre> <p>Property:yAxis Name</p> <pre>let chart; Chart = new Chart({ series: [{ yAxisName: 'secondaryY Axis' }]; }); chart .append('' #chart);</pre>

		<p>to axes collect ion.</p> <p>Name of the prope rty in the datas source that contai ns y value for the series.</p> <p>Name of the prope rty in the datas source that contai ns high value for the series.</p> <p>Name of the prope rty in the datas source that contai ns low value</p>	<p>Property:yName let chart: Chart = new Chart({ series: [{ yName: 'y' }]);});chart .appendTo('#chart');</p> <p>Property:high1 let chart: Chart = new Chart({ series: [{ high: 'y' }]);});chart .appendTo('#chart');</p> <p>Property:low1 let chart: Chart = new Chart({ series: [{ low: 'y' }]);});chart .appendTo('#chart');</p>
--	--	--	--

		<p>for 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 trend lines to chart.</p>	<p>Property:close</p> <pre>let chart: Chart = new Chart({ series: [{ close: 'y' }];});chart .appendTo('#chart');</pre> <p>Property:open</p> <pre>let chart: Chart = new Chart({ series: [{ open: 'y' }];});chart .appendTo('#chart');</pre> <p>Property:trendLines</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines : [{}] }];});chart .appendTo('#chart');</pre>
--	--	--	---

		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:visibility\$("#chart").ejChart({ series: [{ marker: { visible: true } }]});	Property:visibility\$("#chart").ejChart({ series: [{ marker: { visible: false } }]});chart.appendTo('#chart');

		<pre> Property:visibility let chart: Chart = new Property:fill\$("#chart") Fill for ".ejChart({ marke series: [{ marker: r { fill : 'red' } }];}); </pre>	<pre> Property:visibility let chart: Chart = new Property:opacity\$("#chart") Opaci Property:opacity\$("#chart") ty for art").ejChart({ marke series: [{ marker: r { opacity : 0.5 } }];}); </pre>	<pre> Property:shape let chart: Chart = new Property:shape\$("#chart") Shape Property:shape\$("#chart") of rt").ejChart({ marke series: [{ marker: r { shape : 'Circle' } }];}); </pre>	<pre> Property:imageUrl let chart: Chart = new Property:imageUrl\$("#chart") Image Property:imageUrl\$("#chart") Url of chart").ejChart({ marke series: [{ marker: r { imageUrl : '' } }];}); </pre>
--	--	--	---	--	---

		<pre> t.appendTo ('#chart'); Property:shape let chart: Chart = new Property:border\$("#ch art").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }];}); </pre>
Border of marker		<pre> Property:height\$("# #chart").ejChart({ series: [{ marker: { size: { height: 30 } } }];}); </pre>
Height of marker		<pre> Property:width\$("# chart").ejChart({ series: [{ marker: { size: { width: 30 } } }];}); </pre>
Width of marker		<pre> Property:width\$("# chart").ejChart({ series: [{ marker: { size: { width: 30 } } }];}); </pre>

		<pre>width: 25 } }]);});char t.appendTo ('#chart');</pre>
		<pre>Property:mar ker.dataLabel let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]);});char t.appendTo ('#chart');</pre>
DataL abelS etting s of marke r	Property:marker.dataLa bel\$("#chart").ejCh art({ series: [{ marker: {dataLabel: { } } }]);});	<pre>Property:data Label.visible1 et chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]);});char t.appendTo ('#chart');</pre>
Visibil ity of dataL abel	Property:dataLabel.visibl e\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { visible: true } } }]);});	<pre>Property:data Label.namele t chart: Chart = new Chart({ series: [{ marker: { dataLabel: { name: ' ' } } }]);});char t.appendTo ('#chart');</pre>
Text mappi ng name of dataL abel	Property:dataLabel.text MappingName\$("#cha rt").ejChart({ series: [{ marker: {dataLabel: { textMappingName: ' ' } } }]);});	<pre>Property:data Label.fill\$ (Property:data Label.filllet chart: Chart = new</pre>
Fill color of	Property:dataLabel.fill\$ ("#chart").ejChart({ series: [{ marker: {dataLabel: {	

		<pre> data fill: 'pink' } } label }};}); Property:data Label.filllet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { fill: 'pink' } } }];});char t.appendTo ('#chart); </pre>
	Opacity of data label	<pre> Property:dataLabel.opac ity\$("#chart").ejCh art({ series: [{ marker: {dataLabel: { opacity: 0.6 } } }];}); Property:data Label.position let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { opacity: 0.4 } } }];});char t.appendTo ('#chart); </pre>
	Text position of data label	<pre> Property:dataLabel.text Position\$("#chart"). ejChart({ series: [{ marker: {dataLabel: { textPosition: 'middle' } } }];}); Property:data Label.alignme ntlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { position: 'Top' } } }];});char t.appendTo ('#chart); </pre>
	Alignment of data label	<pre> Property:dataLabel.verti calAlignment\$("#char t").ejChart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }];}); Property:data Label.alignme ntlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }];}); </pre>

		<pre> });});chart t.appendTo ('#chart'); </pre>
		<pre> Property:data Label.alignme ntlet chart: chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } } }];});char t.appendTo ('#chart'); </pre>
	Border of data label	<pre> Property:dataLabel.border er\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4} } } }];}); </pre>
	Offset for data label	<pre> Property:dataLabel.offset t\$("#chart").ejCha rt({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }];}); </pre> <p>Not Applicable</p>
	Margin of data label	<pre> Property:data Label.margin1 et chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10}} } }];}); });});char t.appendTo ('#chart'); </pre>
	Font of	<pre> Property:dataLabel.border er\$("#chart").ejCha rt({ series: [{ marker: </pre> <p>Property:data Label.margin1 et chart: Chart =</p>

		<pre> data {dataLabel: { label font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight: '600', opacity: 0.5, size: 12, color: 'red' }} } }];}); new Chart({ series: [{ marker: {dataLabel : { font: { fontFamily : 'SegoeUI', fontStyle: 'italic', fontWeight : '600', opacity: 0.5, size: 12, color: 'red' }} } }];});char t.appendTo ('#chart'); Property:data Label.templat e let chart: Chart = new Chart({ series: [{ marker: {dataLabel : { template: ' ' } } } }];});char t.appendTo ('#chart'); Property:data Label.rx let chart: Chart = new Chart({ series: [{ marker: {dataLabel : { rx: 10 } } }];});char </pre>
		<pre> Property:dataLabel.tem plate\$("#chart").ej HTML Chart({ series: [{ templ marker: ate in {dataLabel: { datal template: ' abel Chart ' } } }];}); </pre>
		<pre> Roun ded corne r radius X Not Applicable </pre>

		<pre> t.appendTo ('#chart'); Property:data Label.rylet chart: Chart = new Chart({ series: [{ marker: {dataLabel : { ry: 10 } } }];});char t.appendTo ('#chart'); </pre>
	<p>Rounded corner radius</p> <p>Not Applicable</p>	<pre> Property:dataLabel.maxi mumLabelWidth\$("#ch art").ejChart({ series: [{ marker: {dataLabel: { maximumLabelWidth: 20}} } }];}); </pre> <p>Not Applicable</p>
	<p>Enable wrapping of text for data label</p> <p>Property:dataLabel.enableWrap\$("#chart").ejChart({ series: [{ marker: {dataLabel: {enableWrap: true}} } }];});</p> <p>Not Applicable</p>	
	<p>To show contrast color for data label</p> <p>Property:dataLabel.showContrastColor\$("#chart").ejChart({ series: [{ marker: {dataLabel: {showContrastColor: true}} } }];});</p> <p>Not Applicable</p>	
	<p>To show edge label for</p> <p>Property:dataLabel.showEdgeLabels\$("#chart").ejChart({ series: [{ marker: {dataLabel: {showEdgeLabels: true}} } }];});</p> <p>Not Applicable</p>	

		<pre> data label ## TrendLines Be hav iou r API in Essential JS 1 API in Essential JS 2 Property:series.trendLines let ndLines chart: Chart = new Chart({ series: [{ trendLines: []}]})};chart.appendTo('#chart'); Property:series.trendLines visibility\$("#chart").ejChart({ series: [{ trendLines: [visibility: true }]})}; Property:trendLine s.type let chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial' }]})};chart.appendTo('#chart'); Property:trendLine s.name let chart: Chart = new Chart({ series: [{ trendLines: [name: 'trendLine' }]})};chart.appendTo('#chart'); Property:trendLines period\$("#chart"). s.period let </pre>
--	--	---

		<pre> of ejChart({ tre series: [{ ndL trendLines: [ine period: 45]}] }); Pol yno mia l ord er Property:trendLines.p for olynomialOrder\$("#c Pol hart").ejChart({ yno series: [{ mia trendLines: [l polynomialOrder: typ 3]}] }); e tre ndL ine s Bac kw ard Property:trendLines.b for ackwardforecast\$("# eco chart").ejChart(st { series: [{ for trendLines: [tre backwardforecast ndL : 3]}] }); ine s For Property:trendLines.f wa orwardForecast\$("#c rd hart").ejChart({ for series: [{ eco trendLines: [st forwardForecast: for 3]}] }); tre ndL </pre>	<pre> chart: Chart = new Chart({ series: [{ trendLines: [period: 45]]] });chart. appendTo('#cha rt'); Property:trendLine s.polynomialOrder let chart: Chart = new Chart({ series: [{ trendLines: [polynomialOrde r: 3]]] });chart.a ppendTo('#char t'); Property:trendLine s.backwardforecast let chart: Chart = new Chart({ series: [{ trendLines: [backwardforeca st: 3]]] });chart.a ppendTo('#char t'); Property:trendLine s.forwardForecast1 et chart: Chart = new Chart({ series: [{ trendLines: [forwardForecas t: 3]]] });chart.a </pre>
--	--	--	---

		<pre> ine s </pre>	<pre> ppendTo('#chart'); </pre>
		<pre> Fill for tre ndL ine s </pre>	<pre> Property:trendLine s.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: '#EEFFCC']]}); chart.appendTo('#chart'); </pre>
		<pre> Wi dth for tre ndL ine 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 val ue for tre ndL 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 sha pe for tre ndL ine s </pre>	<pre> Property:trendLine s.legendShapelet chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']} });chart.appen dTo('#chart'); </pre>
		<pre> Ani ma tio n </pre>	<pre> Property:trendLine s.animationlet chart: Chart = new Chart({ </pre>

		<pre> set tin gs for tre ndL ine s Ma rke r set tin gs for tre ndL ine s To oltp p for tre ndL ine s Das hAr ray for tre ndL ine s Visi ble on leg en d for </pre>	<pre> 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'); Not Applicable. Not Applicable. </pre>
--	--	--	---

		<pre> tre ndL ine s ## Striplines Beh avio API in Essential JS 1 API in Essential JS 2 ur Property:primaryX Axis.striplineslet Defa Property:primaryXAx chart: Chart = ult is.striplines\$("#cha new Chart({ beh rt").ejChart({ primaryXAxis: avio primaryXAxis: { { striplines: ur primaryXAxis: { { [{ visible: for stripLines: [{ [{ true strip visible: true true lines }]]}); chart.ap pendTo('#chart '); Property:striplines .borderlet Property:striplines.b chart: Chart = bord orderColor\$("#cha new Chart({ er rt").ejChart({ primaryXAxis: for primaryXAxis: { { striplines: strip stripLines: [{ [{ border: { line borderColor: { color: 'red', width: 2} 'pink' }]]}); chart.ap pendTo('#chart '); Property:striplines .borderlet Back Property:striplines.c chart: Chart = grou or\$("#chart").e new Chart({ nd jChart({ primaryXAxis: colo primaryXAxis: { { striplines: r for stripLines: [{ [{ color: strip color: 'pink' 'red'}}]]}); cha line }]]}); rt.appendTo('# chart'); Property:striplines.st Property:striplines Star art\$("#chart").e .startlet chart: t jChart({ Chart = new valu primaryXAxis: { Chart({ e for stripLines: [{ primaryXAxis: start: 10 }]]}); { striplines: [{ start: </pre>
--	--	---

		<pre> strip 5}}}});chart.a line ppendTo('#char t'); Property:stripLines .endlet chart: End Property:stripLines.e Chart = new valu nd\$("#chart").ej Chart({ e for Chart({ strip primaryXAxis: { primaryXAxis: 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 ext\$("#chart").e Chart = new in jChart({ Chart({ strip primaryXAxis: { primaryXAxis: line stripLines: [{ { stripLines: text: [{ text: 'StripLine; 'stripline'}}} }}}}); });chart.appen dTo('#chart'); Property:stripLines .horizontalAlignme Text Property:stripLines.t ntlet chart: align extAlignment\$("#ch Chart = new men art").ejChart({ Chart({ t in primaryXAxis: { primaryXAxis: strip primaryXAxis: { { stripLines: line textAlignment: horizontalAlig 'Far; }]]}); nment: 'Far'}}}});cha rt.appendTo('# chart'); Property:stripLines .verticalAlignment Vert Not Applicable ical </pre>
--	--	--

		<pre> Text align men t in strip line </pre>	<pre> let chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ verticalAlignm ent: 'Far'}}]]});cha rt.appendTo('# chart'); </pre>
		<pre> Property:stripLines.w idth\$("#chart").e jChart({ primaryXAxis: { stripLines: [{ width: 10; }}]]}); </pre>	<pre> Property:stripLines .sizelet chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ size: 10 }}]]});chart.ap pendTo('#chart '); </pre>
		<pre> ZInd ex of strip line </pre>	<pre> Property:stripLines.zl ndex\$("#chart"). ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }}]]}); </pre>
		<pre> Font style of strip line </pre>	<pre> Property:stripLines.f ontStyle\$("#chart ").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }}]]}); </pre>
		<pre> ## Multilevel Labels </pre>	<pre> Property:stripLines .textStylelet chart: Chart = new Chart({ primaryXAxis: [{ stripLines: [{ textStyle: {} }}]]});chart.ap pendTo('#chart '); </pre>
		<pre> Beh avio ur </pre>	<pre> API in Essential JS 2 </pre>

		<pre> Default behavior available url for multiple levels labels els Property:primaryX Axis.multilevelLab elslet chart: Chart = new Chart({ primaryXAxis: primaryXAxis: { multilevelLabels multilevelLab els: [{ visible: true }]]]); chart.a ppendTo('#cha rt'); </pre>
		<pre> Default behavior available url for multiple levels labels els Property:primaryX Axis.multilevelLab elslet chart: Chart = new Chart({ primaryXAxis: primaryXAxis: { multilevelLabels multilevelLab els: [{ visible: true }]]]); chart.a ppendTo('#cha rt'); </pre>
		<pre> Text align ment for multiple levels labels els Property:multilev elLabels.alignmen tlet chart: Chart = new Chart({ primaryXAxis: { multilevelLab els: [{alignment: 'Near' }]]]); chart.a ppendTo('#cha rt'); </pre>
		<pre> Text overflow for multiple levels labels els Property:multiLev elLabels.overFlow let chart: Chart = new Chart({ primaryXAxis: primaryXAxis: { multilevelLab els: [{overflow: 'Trim' }]]]); chart.a </pre>

		<pre> appendTo('#chart'); Property:multiLevelLabels.borderlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ border: { width: 2, color: 'red', type: 'brace' } }]]]); appendTo('#chart'); </pre>
Border for multilevelLabels	<pre> Property:multiLevelLabels.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ start: 45 }]]]]); </pre>	<pre> appendTo('#chart'); Property:multiLevelLabels.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ categories: [{ start: 45} }]]]]]); chart.appendTo('#chart'); </pre>
End value for label	<pre> Property:multiLevelLabels.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ end: 45 }]]]]]); </pre>	<pre> appendTo('#chart'); Property:multiLevelLabels.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ categories: [{ end: 45}] }]]]]]); chart.appendTo('#chart'); </pre>

		<pre> Property:multiLevelLabels.categories s.textlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ categories: [{ text: 'text' }] } }}});chart.a ppendTo('#cha rt'); </pre>
		<pre> Property:multiLevelLabels.categories s.maximumTextW idthlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels els: [{ categories: [{ maximumTextWi dth: 20 }] } }}});chart.a ppendTo('#cha rt'); </pre>
		<pre> Property:multiLevelLabels.level\$("#chart").ejChart({ primaryXAxis: { multilevelLabels : [{ level: 2 } }}]); </pre>
		<p>Not applicable. Categories are used</p>
		<p>## Methods</p>
		<pre> Beh avio ur API in Essential JS 1 API in Essential JS 2 </pre>
		<pre> Property:chart.animate\$("#chart").ejChart ({ animate: () { }}); </pre>
		<p>Not applicable</p>

		<pre> series Property:chart.refresh() let chart: Chart = new Chart({});chart.ap pendTo('#chart');c hart.width = '400';chart.refres h(); Property:chart.export() let chart: Chart = new Chart({});chart.ex port('JPEG', 'chart');chart.app endTo('#chart'); Property:chart.print()le t chart: Chart = new Chart({});chart.pr int('chart');chart .appendTo('#chart'); Property:chart.addSerie s()let chart: Chart = new Chart({});chart.ap pendTo('#chart');c hart.addSeries(); Property:chart.removeS eries()let chart: Chart = new Chart({});chart.ap pendTo('#chart');c hart.removeSeries(); ## Events Beh avio API in Essential JS 1 API in Essential ur JS 2 Fires Property:annotationClick on \$("#chart").ejChar ann t({ otati annotationClick: () { }); </pre>
--	--	---

		<p>on click</p> <p>Property:animationComplete</p> <p>Fires Property:animationComplete after let chart: Chart = new Chart({ animationComplete: () => { chart.appendTo('#chart'); } });</p> <p>Fires Property:axisLabelClick\$ on ("#chart").ejChart({ axisLabelClick: () { } });</p> <p>Property:axisLabelRender()</p> <p>Fires Property:axisLabelRendering before ring\$ ("#chart").ejChart({ axisLabelRendering: () => { } });</p> <p>Fires Property:axisLabelMouseMove on Move\$ ("#chart").ejChart({ axisLabelMouseMove: () { } });</p> <p>Fires Property:axisLabelInitialize on ze\$ ("#chart").ejChart({ axisLabelInitialize: () { } });</p> <p>Fires Property:axesRangeCalculated before late\$ ("#chart").ejChart({</p>
--	--	--

		<pre> axis axesRangeCalculate Chart = new rang : () { }); Chart({ e axisRangeCa calc lculated: ulati () => { on }}};chart.a ppendTo('#c hart'); </pre>
		<pre> Fires on Property:axisTitleRenderi axis ng\$("#chart").ejCha title rt({ rend axisTitleRendering erin : () { }); g </pre>
		<pre> Fires on after Property:afterResize\$ (" char #chart").ejChart({ t afterResize: () { resiz }}}; e </pre>
		<pre> Fires on befo Property:beforeResize\$(re "#chart").ejChart(char { beforeResize: () t { }); resiz e </pre>
		<pre> Fires on Property:chartClick\$ ("# char chart").ejChart({ t chartClick: () { click }}}; </pre>
		<pre> Fires Property:chartMouseMove\$ ("#chart").ejChart({ on rt({ char chartMouseMove: () t { }); mou </pre>

		<pre> se mov e </pre>	<pre> chartMouseM ove: () => { }});chart.a ppendTo('#c hart'); </pre>
		<pre> Fires on char t mou 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 re char t dou ble click </pre>	<pre> Property:chartDoubleClick k\$("#chart").ejChar t({ chartDoubleClick: () { }}); </pre> <p>Not applicable</p>
		<pre> Fires on char t mou se up </pre>	<pre> Property:chart mouseUp let chart: Chart = new Chart({ chartmouseU p: () => { }});chart.a ppendTo('#c hart'); </pre> <p>Not Applicable</p>
		<pre> Fires on char t mou se dow n </pre>	<pre> Property:chart mouseDownle t chart: Chart = new Chart({ chartmouseD own: () => { }});chart.a ppendTo('#c hart'); </pre> <p>Not Applicable</p>

		<p>Fires during the calculation of chart area bounds. You can use this even to customize the bounds of chart area</p> <p>Property:chartAreaBoundsCalculate\$("#chart").ejChart({chartAreaBoundsCalculate: () { }});</p> <p>Not applicable</p> <p>Fires when the dragging is started</p> <p>Property:dragStart\$("#chart").ejChart({dragStart: () { }});</p> <p>Not applicable</p> <p>Fires while dragging</p> <p>Property:dragging\$("#chart").ejChart({dragging: () { }});</p> <p>Not applicable</p>
--	--	---

		<p>Fires when the dragging is completed</p> <p>Property:dragComplete</p> <pre>let chart: Chart = new Chart({ dragComplete: () => { }});chart.a ppendTo('#c hart');</pre>
		<p>Fires when character is destroyed completely.</p> <p>Property:destroy\$("#chart").ejChart({destroy: () { }});</p> <p>Not applicable</p>
		<p>Fires after character is created.</p> <p>Property:load</p> <pre>let chart: Chart = new Chart({ loaded: () => { }});chart.a ppendTo('#c hart');</pre>
		<p>Fires before rendering the data labels.</p> <p>Property:textRenderer</p> <pre>let chart: Chart = new Chart({ textRenderer: () => { }});chart.a ppendTo('#c hart');</pre>
		<p>Fires when error bars are rendered.</p> <p>Property:errorBarRendering\$("#chart").ejChart({errorBarRendering: () { }});</p> <p>Not applicable</p>

		<pre> r bar is rend erin g. Fires duri ng the calc Property:legendBoundsC ulati calculate\$("#chart").ej on jChart({ Not applicable of legendBoundsCalcul ate: () { }}); lege nd bou nds. Fires on clicki Property:legendItemClick ng \$("#chart").ejChar the t({ Not applicable lege legendItemClick: nd () { }}); item . Fires whe n movi Property:legendItemMo ng useMove\$("#chart"). mou ejChart({ Not applicable se legendItemMouseMov over e: () { }}); lege nd item Fires Property:legen befo Property:legendItemRen dRenderlet re dering\$("#chart").ej chart: rend Chart({ Chart = new erin legendItemRenderin Chart({ g g: () { }}); legendRende the r: () => { }});chart.a </pre>
--	--	--

		<pre> legend item . Fires before render load Property:load\$("#chart") .ejChart({ load: () { } }); chart. . Fires , when on multi i level label s are render erin g. Fires on click ing a poin t in chart. . Fires when on mouse is move ed over a poin t. </pre>	<pre> appendTo('#chart'); Property:load let chart: Chart = new Chart({ load: () => { }});chart.append To('#chart'); Property:axisM ultiLabelRende r let chart: Chart = new Chart({ axisMultiLa belRender : () => { }});chart.a ppendTo('#c hart'); Property:point Click let chart: Chart = new Chart({ pointClick : () => { }});chart.a ppendTo('#c hart'); Property:point Move let chart: Chart = new Chart({ pointMove : () => { }});chart.a ppendTo('#c hart'); </pre>
--	--	---	---

		<p>Fires before rendering. Property:preRender\$("#chart").ejChart({preRender: () {}});</p> <p>Not applicable</p>
		<p>Fires when point is rendered. Property:pointRender let chart: Chart = new Chart({pointRender: () => {}});chart.appendTo('#chart');</p>
		<p>Fires after selection. Property:rangeSelected\$("#chart").ejChart({rangeSelected: () {}});</p> <p>Not applicable</p>
		<p>Fires after selection. Property:seriesRegionClick\$("#chart").ejChart({seriesRegionClick: () {}});</p> <p>Not applicable</p>
		<p>Fires before rendering. Property:seriesRendering\$("#chart").ejChart({seriesRendering: () {}});</p> <p>Property:seriesRender let chart: Chart = new Chart({seriesRender: () => {}});chart.appendTo('#chart');</p>
		<p>Fires before rendering. Property:symbolRendering\$("#chart").ejChart({symbolRendering: () {}});</p> <p>Not applicable</p>

		<pre> re rt({ rend symbolRendering: erin () { })); g the mar ker sym bols. Fires befo re Property:trendlineRende rend ring\$("#chart").ejCh erin art({ g trendlineRendering the : () { })); tren dline Fires befo re rend Property:titleRendering\$ erin (\$("#chart").ejChart g ({ titleRendering: the () { })); Char t title. Fires befo re rend Property:subTitleRenderi erin ng\$("#chart").ejCha g rt({ the subTitleRendering: Char () { })); t sub title. Fires befo Property:toolTipInitialize re \$("#chart").ejChar rend t({ erin toolTipInitialize: g () { })); </pre>	<p>Not applicable</p> <p>Not applicable</p> <p>Not applicable</p> <p>Property:toolti pRender let chart: Chart = new Chart({ tooltipRend er : () =></p>
--	--	--	---

		<pre> the toolt ip. Fires befo re rend erin g cros shair toolt ip in axis Fires befo re rend erin g trac kball toolt ip. Even t trigg ered when scroll l start s. Even t trigg ered when scroll l </pre>	<pre> { });chart.a ppendTo('#c hart'); Property:trackAxisToolTi p\$("#chart").ejChar t({ trackAxisToolTip: () { }); Property:trackToolTip\$("#chart").ejChart({ trackToolTip: () { }); Property:scroll Start let chart: Chart = new Chart({ scrollStart : () => { });chart.a ppendTo('#c hart'); Property:scroll Endlet chart: Chart = new Chart({ scrollEnd: () => { });chart.a ppendTo('#c hart'); </pre>	<pre> Not applicable Not applicable </pre>
--	--	--	---	---

		<pre> ends . Even t trigg ered Property:scrollChange\$ when "#chart").ejChart(n { scrollChange: () scroll { }}); l chan ges. Fires while e performi ng Property:zoomComplete rect \$("#chart").ejChar angle({ zoomComplete: e () { }}); zoo min g in char t. Behavi API in Essential JS our 1 API in Essential JS 2 Property:selectedDataPointIndexes let \$("#chart").ejChart({ selectedDataPointIndexes: [[{ seriesIndex: 0, pointIndex: 1}]]}); chart: new Chart({ selectedDataPointIndexes: [[{ series: 0, point: 1}]]}); chart.appendTo('#chart'); Property:sideBySideSeriesPlacement let \$("#chart").ejChart({ sideBySideSeriesPlacement: </pre>
--	--	---

		<pre> column n based series iesPlacement} true});chart.ap); pendTo('#chart'); </pre>
		<pre> Property:zoomin g:\$("#chart") .ejChart({ zooming: { enable: true, enableDeferre dZoom: true, enablePinch: true, enableMouseWh eel: true, enableScrollBa r: true, toolBarItems: [], type: 'X' }); </pre>
	ZoomS ettings	<pre> Property:zoomSetti ngslet chart: Chart = new Chart({ zoomSettings: { enable: true, enablePinchZoom ing: true, enableDefferedZ ooming: true enableMouseWhee lZooming: true, enableSelection Zooming: true, enableScrollBar : true });chart.append dTo('#chart'); </pre>
	Backgr ound color of the chart	<pre> Property:backgro und \$("#container ").ejChart({ background: 'transparent' }); </pre>
	URL of the image to be used as chart backgr ound.	<pre> Property:backGro undImageUrl \$("#container ").ejChart({ backGroundIma geUrl : '../images/ch art/wheat.png '}); </pre>
	Custo mizing border of the chart	<pre> Property:border Property:borderlet \$("#container ").ejChart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}); </pre>

		<div><div><div>This</div><div>provide</div><div>Property:exportSettings</div><div>\$("#container").ejChart({</div><div>exportSettings: {</div><div> filename: "chart",</div><div> angle: '45'</div><div>}}</div><div>});</div><div>s</div></div><div><div>Property:export()</div><div>let chart: Chart = new Chart({</div><div> border: {</div><div> width: 2,</div><div> color: '#CCEEFF'}}</div><div>});chart.appendTo('#chart');</div><div>chart.export(type, fileName);</div></div><div><div>Property:chartArea</div><div>let chart: Chart = new Chart({</div><div> chartArea: {</div><div> background: 'transparent',</div><div> border: {</div><div> width: 2,</div><div> color: '#CCEEFF',</div><div> opacity: 0.3,</div><div> color: 'red',</div><div> width: 2}}</div><div>});chart.appendTo('#chart');</div><div>chart.export(type, fileName);</div></div><div><div>ChartArea</div><div>customization</div><div>{ chartArea: {</div><div> background: 'transparent'</div><div>, border: {</div><div> opacity: 0.3,</div><div> color: 'red',</div><div> width: 2}}</div><div>}}});</div></div></div>
--	--	--

```
<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>Alternate grid band</b></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>Axis line cross value</td>``<td>``Property:<i>crossesAt</i>``</br>``</br>``<code>`

```
$("#container").ejChart({  
  primaryXAxis: { crossesAt: 0 }  
});
```

`</code>``</td>``<td>``Property:<i>crossesAt</i>``</br>``</br>``<code>`

```
let chart: Chart = new Chart({  
  primaryXAxis: { crossesAt: 4 }  
});  
chart.appendTo('#chart');
```

`</code>``</td>``</tr>``<tr>``<td>axis name with which the axis line has to be crossed</td>``<td>`

Property: *crossesInAxis*

<code>

```
$("#container").ejChart({  
  primaryXAxis: { crossesInAxis: " }  
});
```

</code>

</td>

<td>

Property: *crossesInAxis*

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { crossesInAxis: " }  
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>**axis elements placed with axis line**</td>

<td>

Property: *showNextToAxisLine*

<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>

Property:<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>
</td>
```

```
<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>
```

<pre><tr> <td>Column index of an axis</td> <td> Property<i>columnIndex</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { columnIndex: 2 } }); </code> </td> <td> Property<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 }</pre>

```

});
</code>
</td>
<td>
<b>Property</b>:<i>span</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { span: 2 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Crosshair label of an axis</b></td>
<td>
<b>Property</b>:<i>crossHairLabel.visible</i>
<br>
<br>
<code>
$("#container").ejChart({
primaryXAxis: { crossHairLabel: { visible: true }}
});
</code>
</td>
<td>
<b>Property</b>:<i>crossHairTooltip.enable</i>
<br>
<br>
<code>
let chart: Chart = new Chart({

```

```
primaryXAxis: { crossHairTooltip: { enable: true }}
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Crosshair label color of an axis</b></td>
```

```
<td>
```

Not applicable

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>crossHairTooltip.fill</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
```

```
primaryXAxis: { crossHairTooltip: { fill: 'red' }}
```

```
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Crosshair label text style</b></td>
```

```
<td>
```

Not applicable

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>crossHairTooltip.textStyle</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```

let chart: Chart = new Chart({
primaryXAxis: { crossHairTooltip: { textStyle: { } } }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Desired interval count for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>desiredIntervals</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { desiredIntervals: 4 }
});
</code>
</td>
<td>
<b>Property</b>:<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>

```


<pre><td> Property:<i>edgeLabelPlacement</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { edgeLabelPlacement: 'none' } }); </code> </td> <td><pre><td> Property:<i>edgeLabelPlacement</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { edgeLabelPlacement: 'Shift' } }); chart.appendTo('#chart'); </code> </td> </pre></td></pre>	<pre><td> Property:<i>edgeLabelPlacement</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { edgeLabelPlacement: 'Shift' } }); chart.appendTo('#chart'); </code> </td> </pre>
<pre></tr> <tr> <td>Enables trim for axis labels</td> <td> Property:<i>enableTrim</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { enableTrim: true } }); </code> </pre>	

</td>

<td>

Property:<i>enableTrim</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { 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({  
  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>Specifies the interval of the axis according to the zoomed data of the chart</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>enableAutoIntervalOnZooming</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
```

```
primaryXAxis: { enableAutoIntervalOnZooming: true }
```

```
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<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>
<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>Interval type for date time axis</td>

<td>

Property:<i>intervalType</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  primaryXAxis: { intervalType: 'Auto' }
});
```

</code>

</td>

<td>

Property:<i>intervalType</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  primaryXAxis: { 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({
primaryXAxis: { isInversed: true }
});
</code>
</td>
<td>
<b>Property</b>:<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>
<b>Property</b>:<i>alignment</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { alignment: 'center' }
});
</code>
```

</td>

<td>

Not Applicable

</td>

</tr>

<tr>

<td>Rotation of axis labels</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { labelRotation: 45 }  
});
```

</code>

</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { labelRotation: 45 }  
});  
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Log base value for logarithmic axis</td>

<td>

Property:<i>logBase</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  primaryXAxis: { logBase: 10 }  
});
```

</code>

</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  primaryXAxis: { logBase: 10 }  
});
```

```
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Major grid line</td>

<td>

Property:<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>Color of MajorGridLines</td>

<td>

Property:<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>

```

```

<tr>

```

```

<td><b>Major Tick line</b></td>

```

```

<td>

```

```

<b>Property</b>:<i>majorTickLines.visible</i>

```

```

</br>

```

```

</br>

```

```

<code>

```

```

$("#container").ejChart({

```

```
primaryXAxis: { 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({  
primaryXAxis: { majorTickLines: { width: 2} }  
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>majorTickLines.width</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({  
primaryXAxis: { majorTickLines: { width: 2} }  
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

<td>Height of MajorTickLines</td>

<td>

Property:<i>majorTickLines.size</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
primaryXAxis: { majorTickLines: { size: 2 } }  
});
```

</code>

</td>

<td>

Property:<i>majorTickLines.height</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
primaryXAxis: { majorTickLines: { height: 2 } }  
});  
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Color of MajorTickLines</td>

<td>

Property:<i>majorTickLines.color</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
primaryXAxis: { majorTickLines: { color: 'black' } }  
});
```


`</code>``</td>``<td>``Property:<i>majorTickLines.color</i>``</br>``</br>``<code>`

```
let chart: Chart = new Chart({
  primaryXAxis: { majorTickLines: { color: 'black' } }
});
```

```
chart.appendTo('#chart');
```

`</code>``</td>``</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>``</td>``<td>``Not Applicable``</td>``</tr>``<tr>``<td>maximum labels of primaryXAxis</td>``<td>`

Property: *maximumLabels*

`<`

```
$("#container").ejChart({
```

```
primaryXAxis: { maximumLabels: 5 }
```

```
});
```

`>`

`<`

Property: *maximumLabels*

`<`

```
let chart: Chart = new Chart({
```

```
primaryXAxis: { maximumLabels: 4 }
```

```
});
```

```
chart.appendTo('#chart');
```

`>`

`<`

`<`**maximum labels width of primaryXAxis to trim**`>`

`<`

Property: *maximumLabelWidth*

`<`

```
$("#container").ejChart({
```

```
primaryXAxis: { maximumLabelWidth: 40 }
```

```
});
```

`>`

<div><td> Property:<i>maximumLabelWidth</i> </br> </br> <code> let chart: Chart = new Chart({ primaryXAxis: { maximumLabelWidth: 4 } }); chart.appendTo('#chart'); </code> </td> </tr><tr><td><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><tr> <td>Width of minorGridLines</td> <td> Property:<i>minorGridLines.width</i> </br></td></tr></td></tr></div>	<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><tr> <td>Width of minorGridLines</td> <td> Property:<i>minorGridLines.width</i> </br></td></tr>	<tr> <td>Width of minorGridLines</td> <td> Property:<i>minorGridLines.width</i> </br>
<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><tr> <td>Width of minorGridLines</td> <td> Property:<i>minorGridLines.width</i> </br></td></tr>	<tr> <td>Width of minorGridLines</td> <td> Property:<i>minorGridLines.width</i> </br>	
<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>

Property:<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>
```

Opacity of minor grid line	
Property : <i>minorGridLines.opacity</i>	
<pre>\$("#container").ejChart({ primaryXAxis: { minorGridLines: { opacity: true} } });</pre>	
Not Applicable	
minor Tick line	
Property : <i>minorTickLines.visible</i>	
<pre>\$("#container").ejChart({ primaryXAxis: { minorTickLines: { visible: true} } });</pre>	
Not Applicable	

<pre><tr> <td>Width of minorTickLines</td> <td> Property<i>minorTickLines.width</i> </br> </br> <code> \$("#container").ejChart({ primaryXAxis: { minorTickLines: { width: 2} } }); </code> </td> <td> Property<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> \$("#container").ejChart({ primaryXAxis: { minorTickLines: { size: 2} }</pre>

```

});
</code>
</td>
<td>
<b>Property</b>:<i>minorTickLines.height</i>
<br>
<br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { minorTickLines: { height: 2} }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Color of minorTickLines</b></td>
<td>
<b>Property</b>:<i>minorTickLines.color</i>
<br>
<br>
<code>
$("#container").ejChart({
primaryXAxis: { minorTickLines: { color: 'black' } }
});
</code>
</td>
<td>
<b>Property</b>:<i>minorTickLines.color</i>
<br>
<br>
<code>
let chart: Chart = new Chart({

```



```
primaryXAxis: { minorTickLines: { color: 'black' } }
});
```

```
chart.appendTo('#chart');
```

```
</code>
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Opacity of minor Tick line</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTickLines.opacity</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
```

```
primaryXAxis: { minorTickLines: { opacity: true } }
```

```
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
Not Applicable
```

```
</td>
```

```
</tr>
```

```
<tr>
```

```
<td><b>Minor ticks per interval of primaryXAxis</b></td>
```

```
<td>
```

```
<b>Property</b>:<i>minorTicksPerInterval</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
$("#container").ejChart({
```

```
primaryXAxis: { minorTicksPerInterval: 4 }
```

```
});
```

```
</code>
```

```
</td>
<td>
<b>Property</b>:<i>minorTickLines.color</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { minorTicksPerInterval: 4 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>name of the primaryXAxis</b></td>
<td>
<b>Property</b>:<i>name</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { name: 'primaryXAxis' }
});
</code>
</td>
<td>
<b>Property</b>:<i>name</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { name: 'primaryXAxis' }
});
```

```
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>Orientation of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>orientation</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { orientation: 'Vertical' }
});
</code>
</td>
<td>
Not Applicable
</td>
</tr>
<tr>
<td><b>Plot offset for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>plotOffset</i>
</br>
</br>
<code>
$( "#container" ).ejChart({
primaryXAxis: { plotOffset: 0 }
});
</code>
</td>
<td>
```

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: *minimum*

```
let chart: Chart = new Chart({  
  primaryXAxis: { minimum: 23 }  
});
```

```
chart.appendTo('#chart');
```

```
</td>
</tr>
<tr>
<td><b>maximum for primaryXAxis</b></td>
<td>
<b>Property</b>:<i>range.maximum</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { range: { maximum: 10 }}
});
</code>
</td>
<td>
<b>Property</b>:<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><b>valueType of primaryXAxis</b></td>
<td>
<b>Property</b>:<i>valueType</i>
</br>
</br>
<code>
$("#container").ejChart({
primaryXAxis: { valueType: 'DateTime' }
});
</code>
</td>
<td>
<b>Property</b>:<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>

```

Property: *visible*

`<`

```
$("#container").ejChart({
```

```
primaryXAxis: { visible: true }
```

```
});
```

`>`

`<`

Property: *visible*

`<`

```
let chart: Chart = new Chart({
```

```
primaryXAxis: { visible: true }
```

```
});
```

```
chart.appendTo('#chart');
```

`>`

`<`

zoomFactor of primaryXAxis

Property: *zoomFactor*

`<`

```
$("#container").ejChart({
```

```
primaryXAxis: { zoomFactor: 0.3 }
```

```
});
```

`>`

```
<td>
<b>Property</b>:<i>zoomFactor</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
primaryXAxis: { zoomFactor: 0.3 }
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>zoomPosition of primaryXAxis</b></td>
<td>
<b>Property</b>:<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>skeleton for an axes</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>

## primaryYAxis		
Behaviour	API in Essential JS 1	API in Essential JS 2
Alternate grid band	Property:alternateGridBand <code>\$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' }}}});</code>	Not applicable
Axis line cross value	Property:crossesAt <code>\$("#container").ejChart({ primaryYAxis: { crossesAt: 0 }});</code>	Property:crossesAt <code>let chart: Chart = new Chart({ primaryYAxis: { crossesAt: 4 }});chart.appendTo('#chart');</code>
axis name with which the axis line has to be crossed	Property:crossesInAxis <code>\$("#container").ejChart({ primaryYAxis: { crossesInAxis: '' }});</code>	Property:crossesInAxis <code>let chart: Chart = new Chart({ primaryYAxis: { crossesInAxis: '' }});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.offset\$("#container").ejChart({ primaryYAxis: { axisLine: { visible : false } } });	Property:visiblelet chart: Chart = new Chart({ primaryYAxis: { visible: false } });chart.appendTo('#chart');
Width of an axis	Property:axisLine.width\$("#container").ejChart({ primaryYAxis: { axisLine: { width : 2 } } });	Property:lineStyle.widthlet chart: Chart = new Chart({ primaryYAxis: { lineStyle: { width: 3 } } });chart.appendTo('#chart');
Column index of an axis	Property:columnIndex\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });	Property:columnIndexlet chart: Chart = new Chart({ primaryYAxis: { columnIndex: 2 } });chart.appendTo('#chart');
span of an axis to place horizontally or vertically	Property:columnSpan\$("#container").ejChart({ primaryYAxis: { columnIndex: 2 } });	Property:spanlet chart: Chart = new Chart({ primaryYAxis: { span: 2 } });chart.appendTo('#chart');
Crosshair label of an axis	Property:crossHairLabel.visible\$("#container").ejChart({ primaryYAxis: { crossHairLabel: { visible: true } } });	Property:crossHairTooltip.enablelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { enable: true } } });chart.appendTo('#chart');
Crosshair label color of an axis	Not applicable	Property:crossHairTooltip.filllet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { fill: 'red' } } });chart.appendTo('#chart');
Crosshair label text style	Not applicable	Property:crossHairTooltip.textStylelet chart: Chart = new Chart({ primaryYAxis: { crossHairTooltip: { textStyle: { } } } });chart.appendTo('#chart');
Desired interval count for primaryYAxis	Property:desiredIntervals\$("#container").ejChart({ primaryYAxis: { desiredIntervals: 4 } });	Property:desiredIntervalslet chart: Chart = new Chart({ primaryYAxis: { desiredIntervals: 4 } });

		<code>});chart.appendTo('#chart');</code>
Edges primaryYAxis	<code>Property:edgeLabelPlacement\$("#container").ejChart({ primaryYAxis: { edgeLabelPlacement: 'none' }});</code>	Property:edgeLabelPlacement <code>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>	Property:enableTrim <code>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>	Property:enableAutoIntervalOnZooming <code>let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: 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>	Property:enableAutoIntervalOnZooming <code>let 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: 'normal', opacity: 0.5, size: 12 } }});</code>	Property:titleStyle <code>let chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } }});chart.appendTo('#chart');</code>
Indexed for category axis	<code>Property:isIndexed\$("#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	<code>Property:intervalType\$("#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\$("#container").ejChart({ primaryYAxis: { isInversed: true }});	Property:isInversedlet chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendTo('#chart');
Custom label format	Property:labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }});	Property:labelFormatlet chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');
labelIntersect Action	Property:labelIntersectAction\$("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }});	Property:labelIntersectActionlet chart: Chart = new Chart({ primaryYAxis: { labelIntersectAction: 'Trim' }});chart.appendTo('#chart');
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.width let 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.color let 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.dashArray let 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.width let 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.height let 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.color let chart: Chart = new Chart({ primaryYAxis: { majorTickLines: { color: 'black' } }});chart.appendTo('#chart');

		'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:maximumLabelslet 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:maximumLabelWidthlet 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.widthlet 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.colorlet 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.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.color let chart: Chart = new Chart({ primaryYAxis: { minorTicksPerInterval: 4 } });chart.appendTo('#chart');
name of the primaryYAxis	Property:name\$("#container").ejChart({ primaryYAxis: { name: 'primaryYAxis' } });	Property:name let 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:plotOffset let 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('#chart');
ScrollBar settings of primaryYAxis	Property:scrollbarSettings\$("#container").ejChart({ primaryYAxis: { scrollbarSettings : { } } });	Not Applicable
TickPosition in primaryYAxis	Property:tickLinesPosition\$("#container").ejChart({ primaryYAxis: { tickLinesPosition: 'Inside' } });	Property:tickPositionlet chart: Chart = new Chart({ primaryYAxis: { tickPosition: 'Inside' } });chart.appendTo('#chart');
valueType of primaryYAxis	Property:valueType\$("#container").ejChart({ primaryYAxis: { valueType: 'DateTime' } });	Property:valueTypelet chart: Chart = new Chart({ primaryYAxis: { valueType: 'DateTime' } });chart.appendTo('#chart');
visible of primaryYAxis	Property:visible\$("#container").ejChart({ primaryYAxis: { visible: true } });	Property:visiblelet chart: Chart = new Chart({ primaryYAxis: { visible:

		<pre>true });chart.appendTo('#chart');</pre>
zoomFactor of primaryYAxis	<pre>Property:zoomFactor\$("#container").ejChart({ primaryYAxis: { zoomFactor: 0.3 }});</pre>	<pre>Property:zoomFactorlet chart: Chart = new Chart({ primaryYAxis: { zoomFactor: 0.3 }});chart.appendTo('#chart');</pre>
zoomPosition of primaryYAxis	<pre>Property:zoomPosition\$("#container").ejChart ({ primaryYAxis: { zoomPosition: 0.3 }});</pre>	<pre>Property:zoomPositionlet chart: Chart = new Chart({ primaryYAxis: { zoomPosition: 0.3 }});chart.appendTo('#chart');</pre>
labelBorder of primaryYAxis	<pre>Property:labelBorder\$("#container").ejChart({ primaryYAxis: { labelBorder: { color: 'red', width: 2} }});</pre>	<pre>Property:borderlet chart: Chart = new Chart({ primaryYAxis: { border: { color: 'red', width: 3 } }});chart.appendTo('#chart');</pre>
title of primaryYAxis	<pre>Property:title.text\$("#container").ejChart({ primaryYAxis: { title: { text: 'Chart title' } }});</pre>	<pre>Property:titlelet chart: Chart = new Chart({ primaryYAxis: { title: 'Chart title' }});chart.appendTo('#chart');</pre>
StripLine of primaryYAxis	<pre>Property:stripLine\$("#container").ejChart({ primaryYAxis: { stripLine: [] }});</pre>	<pre>Property:stripLineslet chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.appendTo('#chart');</pre>
Multilevel labels of primaryYAxis	<pre>Property:multiLevelLabels\$("#container").ejCha rt({ primaryYAxis: { multiLevelLabels: [] }});</pre>	<pre>Property:multiLevelLabelslet chart: Chart = new Chart({ primaryYAxis: { stripLines: [] }});chart.appendTo('#chart');</pre>
skeleton for an axes	Not Applicable	<pre>Property:skeletonlet chart: Chart = new Chart({ axes: [{ skeleton: 'yMd' }]});chart.appendTo('#chart');</pre>
skeleton type for an axes	Not Applicable	<pre>Property:skeletonTypelet chart: Chart = new Chart({ axes: [{</pre>

		<code>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 <code>\$("#container").ejChart({ axes: [{ axisLine: { color: 'red' } }] });</code>	Property:lineStyle.color <code>let chart: Chart = new Chart({ axes: [{ lineStyle: { color: 'black' } }] }); chart.appendTo('#chart');</code>
axis line dashArray	Property:axisLine.dashArray <code>\$("#container").ejChart({ axes: [{ axisLine: { dashArray: '10, 5' } }] });</code>	Property:lineStyle.dashArray <code>let chart: Chart = new Chart({ axes: [{ lineStyle: { dashArray: '10, 5' } }] }); chart.appendTo('#chart');</code>

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 horizontally or vertically	Property:columnSpan\$("#container").ejChart({ axes: [{ columnIndex: 2 }] });	Property:spanlet chart: Chart = new Chart({ axes: [{ span: 2 }] });chart.appendTo('#chart');
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');
Enable 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 of the chart	Property:enableAutoIntervalOnZooming\$("#container").ejChart({ axes: [{ enableAutoIntervalOnZooming: true }] });	Property:enableAutoIntervalOnZoominglet chart: Chart = new Chart({ axes: [{ enableAutoIntervalOnZooming: 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: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',	Property:titleStylelet chart: Chart = new Chart({ axes: [{ titleStyle: { } }] });chart.appendTo('#chart');

	<code>fontWeight: '', opacity: 0.5, size: 12} }]]);</code>	
Indexed for category axis	<code>Property:isIndexed\$("#container").ejChart({ axes: [{ isIndexed: true }]});</code>	<code>Property:isIndexedlet chart: Chart = new Chart({ axes: [{ isIndexed: true }]});chart.appendTo('#chart');</code>
Interval type for date time axis	<code>Property:intervalType\$("#container").ejChart({ axes: [{ intervalType: 'Auto' }]});</code>	<code>Property:intervalTypelet chart: Chart = new Chart({ axes: [{ intervalType: 'Auto }]});chart.appendTo('#chart');</code>
Inversed axis	<code>Property:isInversed\$("#container").ejChart({ axes: [{ isInversed: true }]});</code>	<code>Property:isInversedlet chart: Chart = new Chart({ axes: [{ isInversed: true }]});chart.appendTo('#chart');</code>
Custom label format	<code>Property:labelFormat\$("#container").ejChart({ axes: [{ labelFormat: '{value}K' }]});</code>	<code>Property:labelFormatlet chart: Chart = new Chart({ axes: [{ labelFormat: '{value}K' }]});chart.appendTo('#chart');</code>
labelIntersectAction	<code>Property:labelIntersectAction\$("#container").ejChart({ axes: [{ labelIntersectAction: 'trim' }]});</code>	<code>Property:labelIntersectActionlet chart: Chart = new Chart({ axes: [{ labelIntersectAction: 'Trim' }]});chart.appendTo('#chart');</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	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 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 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 Axis	Property: maximumLabels\$("#container").ejChart({ axes: [{ maximumLabels: 5 }] });	Property: maximumLabelslet chart: Chart = new Chart({ axes: [{ maximumLabels: 4 }] });chart.appendTo('#chart');
maximum labels width of primary Axis 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 minorG	Property: minorGridLines.width\$("#container").ejChart({ axes: [{	Property: minorGridLines.widthlet chart: Chart = new Chart({ axes: [{ minorGridLines: { width: 2 } }] });chart.appendTo('#chart');

ridLines	minorGridLines: { width: 2} }]]]);	
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 minor	Property:minorTickLines.opacity\$("#container").ejChart({ axes: [{ minorTickLines: { opacity: true} }]]});	Not Applicable

Tick line		
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 primary YAxis	Property:range.interval\$("#container").ejChart({ axes: [{ range: { interval: 1 } }] });	Property:intervallet chart: Chart = new Chart({ axes: [{ interval: 2 }] });chart.appendTo('#chart');
RangePadding for primary YAxis	Property:rangePadding\$("#container").ejChart({ axes: [{ rangePadding: 'None' }] });	Property:rangePaddinglet chart: Chart = new Chart({ axes: [{ rangePadding: 'None' }] });chart.appendTo('#chart');

Rounding Places in primary YAxis	Property:roundingPlaces \$("#container").ejChart({ axes: [{ roundingPlaces: 3 }] });	Property:labelFormat let chart: Chart = new Chart({ axes: [{ labelFormat: 'n3' }] });chart.appendTo('#chart');
Scrollbar settings of primary YAxis	Property:scrollbarSettings \$("#container").ejChart({ axes: [{ scrollbarSettings : { } }] });	Not Applicable
TickPosition in primary YAxis	Property:tickLinesPosition \$("#container").ejChart({ axes: [{ tickLinesPosition: 'Inside' }] });	Property:tickPosition let chart: Chart = new Chart({ axes: [{ tickPosition: 'Inside' }] });chart.appendTo('#chart');
valueType of primary YAxis	Property:valueType \$("#container").ejChart({ axes: [{ valueType: 'DateTime' }] });	Property:valueType let chart: Chart = new Chart({ axes: [{ valueType: 'DateTime' }] });chart.appendTo('#chart');
visible of primary YAxis	Property:visible \$("#container").ejChart({ axes: [{ visible: true }] });	Property:visible let chart: Chart = new Chart({ axes: [{ visible: true }] });chart.appendTo('#chart');
zoomFactor of primary YAxis	Property:zoomFactor \$("#container").ejChart({ axes: [{ zoomFactor: 0.3 }] });	Property:zoomFactor let chart: Chart = new Chart({ axes: [{ zoomFactor: 0.3 }] });chart.appendTo('#chart');
zoomPosition of primary YAxis	Property:zoomPosition \$("#container").ejChart({ axes: [{ zoomPosition: 0.3 }] });	Property:zoomPosition let chart: Chart = new Chart({ axes: [{ zoomPosition: 0.3 }] });chart.appendTo('#chart');
labelBorder of primary YAxis	Property:labelBorder \$("#container").ejChart({ axes: [{ labelBorder: { color: 'red', width: 2 } }] });	Property:border let chart: Chart = new Chart({ axes: [{ border: { color: 'red', width: 3 } }] });chart.appendTo('#chart');

title of primary YAxis	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 primary YAxis	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> <tr> <td>Line customization</td> <td>Property: lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%',</td> <td>Property: borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color:</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');	Line customization	Property: lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%',	Property: borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color:
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');															
Line customization	Property: lineColor, lineWidth\$("#chart").ejChart({ rowDefinitions: [{ rowHeight: '50%',	Property: borderlet chart: Chart = new Chart({ rows: [{ height: '300', border: { width: 2, color:															

		<pre> lineColor: 'brown', 'brown'}}]);});chart. lineWidth: 2});}); appendTo('#chart'); </pre>
		<pre> ## Series </pre>
	Behavior	<pre> API in Essential JS 1 </pre>
		<pre> API in Essential JS 2 </pre>
	bearFillColor	<pre> Property:bearFillColor\$("#chart").ejChart({ series: [{{bearFillColor: 'red' }}]);}); </pre>
		<pre> Property:bearFillColor let chart: Chart = new Chart({ series: [{{bearFillColor: 'red' }}]);});chart.append To('#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.append To('#chart'); </pre>
	BoxPlot Mode	<pre> Property:boxPlotMode\$("#chart").ejChart({ series: [{{ boxPlotMode: 'inclusive' }}]);}); </pre>
		<pre> Property:rowslet chart: Chart = new Chart({ series: [{ boxPlotMode: 'Inclusive' }}]);chart.append To('#chart'); </pre>
	Minimum radius of Bubble series	<pre> Property:bubbleOptions.minRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { minRadius: 2} }]);}); </pre>
		<pre> Property:minRadius1 let chart: Chart = new Chart({ series: [{ minRadius: 2 }}]);chart.append To('#chart'); </pre>
	Maximum radius of Bubble series	<pre> Property:bubbleOptions.maxRadius\$("#chart").ejChart({ series: [{ bubbleOptions: { maxRadius: 10} }]);}); </pre>
		<pre> Property:maxRadius1 let chart: Chart = new Chart({ series: [{ maxRadius: 2 }}]);chart.append To('#chart'); </pre>
	bullFillColor	<pre> Property:bullFillColor\$("#chart").ejChart({ series: [{{bullFillColor: 'red' }}]);}); </pre>
		<pre> Property:bullFillColor let chart: Chart = new Chart({ series: </pre>

			<pre> [{} bullFillColor: 'red' });});chart.appe ndTo('#chart'); </pre>
	Cardinal spline tension for spline series	<pre> Property:cardinalSplineTension\$ ("#chart").ejChart({ series: [{ cardinalSplineTension: 0.5 }]);}); </pre>	<pre> Property:cardinalSplineTensionlet chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]);});chart.appe ndTo('#chart'); </pre>
	Column Width for rectangle series	<pre> Property:columnWidth\$ ("#chart").ejChart({ series: [{} columnWidth: 0.5 }]);}); </pre>	<pre> Property:columnWidthlet chart: Chart = new Chart({ series: [{} columnWidth: 0.5 }]);});chart.appe ndTo('#chart'); </pre>
	Column spacing for rectangle series	<pre> Property:columnSpacing\$ ("#chart").ejChart({ series: [{} columnSpacing: 0.5 }]);}); </pre>	<pre> Property:columnSpacinglet chart: Chart = new Chart({ series: [{} columnSpacing: 0.5 }]);});chart.appe ndTo('#chart'); </pre>
	Topleft radius for rectangle series	<pre> Property:cornerRadius.topLeft\$ ("#chart").ejChart({ series: [{ topLeft: 0 }]);}); </pre>	<pre> Property:cornerRadius.topLeftlet chart: Chart = new Chart({ series: [{ topLeft: 0 }]);});chart.appe ndTo('#chart'); </pre>
	topRight radius for rectangle series	<pre> Property:cornerRadius.topRight\$ ("#chart").ejChart({ series: [{ topRight: 0 }]);}); </pre>	<pre> Property:cornerRadius.topRightlet chart: Chart = new Chart({ series: [{ topRight: 0 }]);});chart.appe ndTo('#chart'); </pre>
	bottomRight radius	<pre> Property:cornerRadius.bottomRight\$ ("#chart").ejChart({ </pre>	<pre> Property:cornerRadius.bottomRightlet </pre>

		<p>radius for rectangle series</p> <pre>series: [{ bottomRight: 0 }];});</pre>	<pre>chart: Chart = new Chart({ series: [{ bottomRight: 0 }];});chart.appendTo('#chart');</pre>
		<p>bottomLeft radius for rectangle series</p> <pre>Property:cornerRadius.bottomLeftlet chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];});chart.appendTo('#chart');</pre>	<pre>Property:cornerRadius.bottomLeftlet chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];});chart.appendTo('#chart');</pre>
		<p>DashArray property</p> <pre>Property:dashArray\$("#chart").ejChart({ series: [{ dashArray: '10, 5' }];});</pre>	<pre>Property:dashArray1let chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];});chart.appendTo('#chart');</pre>
		<p>DataSource for series</p> <pre>Property:dataSource\$("#chart").ejChart({ series: [{ dataSource: [] }];});</pre>	<pre>Property:dataSource1let chart: Chart = new Chart({ series: [{ dataSource: [] }];});chart.appendTo('#chart');</pre>
		<p>Draw type for Polar series</p> <pre>Property:drawType\$("#chart").ejChart({ series: [{ drawType: 'Line' }];});</pre>	<pre>Property:drawType1let chart: Chart = new Chart({ series: [{ drawType: 'Line' }];});chart.appendTo('#chart');</pre>
		<p>EmptyPointSettings for series</p> <pre>Property:emptyPointSettings.visible\$("#chart").ejChart({ series: [{ emptyPointSettings: { visible: false } }];});</pre>	<p>Not Applicable</p>
		<p>Empty Point Display mode</p> <pre>Property:emptyPointSettings.displayMode\$("#chart").ejChart({ series: [{ displayMode: 'gap' }];});</pre>	<pre>Property:emptyPointSettings.displayModelet chart: Chart = new Chart({ series: [{ displayMode: 'Average' }];});</pre>

			<pre> });});chart.appendTo('#chart'); Property:emptyPointSettings.filllet chart: Chart = new Chart({ series: [{ fill: 'red' }]);});chart.appendTo('#chart'); Property:filllet chart: Chart = new Chart({ series: [{ emptyPointSettings: { color: 'red', width: 2 }]);});chart.appendTo('#chart'); Property:animation.enablelet chart: Chart = new Chart({ series: [animation: { enable: false }]);});chart.appendTo('#chart'); Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { duration: 1000 }]);});chart.appendTo('#chart'); Property:animation.durationlet chart: Chart = new Chart({ series: [animation: { delay: 100 }]);});chart.appendTo('#chart'); Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }]);}); </pre>
	Empty Point color	Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }]);});	
	Empty Point Border	Property:emptyPointSettings.border\$("#chart").ejChart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]);});	
	Enable animation for series	Property:enableAnimation\$("#chart").ejChart({ series: [{ enableAnimation: true }]});});	
	Animation duration for series	Property:animationDuration\$("#chart").ejChart({ series: [{ animationDuration: 1000 }]});});	
	Animation delay for series	Not Applicable	
	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: pointColor Mappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }];});chart.appendTo('#chart');

		<p>Positive fill for waterfall series</p> <p>Property:positiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }]});</p>	<p>Property:pointColor Mappinglet</p> <pre>chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#chart');</pre>
		<p>Show average value in box and whisker series</p> <p>Property:showMedian\$("#chart").ejChart({ series: [{ showMedian: true }]});</p>	<p>Property:pointColor Mappinglet</p> <pre>chart: Chart = new Chart({ series: [{ showMean: false }]});chart.appendTo('#chart');</pre>
		<p>To group the series of stacking collection.</p> <p>Property:stackingGroup\$("#chart").ejChart({ series: [{ stackingGroup: 'group' }]});</p>	<p>Property:stackingGrouplet</p> <pre>chart: Chart = new Chart({ series: [{ stackingGroup: 'group' }]});chart.appendTo('#chart');</pre>
		<p>Specifies the type of the series to render in chart.</p> <p>Property:type\$("#chart").ejChart({ series: [{ type: 'Line' }]});</p>	<p>Property:typetlet</p> <pre>chart: Chart = new Chart({ series: [{ type: 'Line' }]});chart.appendTo('#chart');</pre>
		<p>Defines the visibility of the series.</p> <p>Property:visibility\$("#chart").ejChart({ series: [{ visibility: true }]});</p>	<p>Property:visiblelet</p> <pre>chart: Chart = new Chart({ series: [{ visible: true }]});chart.appendTo('#chart');</pre>
		<p>Enables or disables the visibility of legend item.</p> <p>Property:visibleOnLegend</p> <pre>\$("#chart").ejChart({ series: [{ visibleOnLegend : true }]});</pre>	<p>Property:toggleVisibilitylet</p> <pre>chart: Chart = new Chart({ legendSettings: [{ toggleVisibility : true }</pre>

		<p>}}});chart.appendTo('#chart');</p> <p>Property:splineType et chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }]);chart.appendTo('#chart');</p> <p>Specifies the different types of spline curve.</p> <p>Property:splineType \$("#chart").ejChart({ series: [{ splineType : 'Natural' }]);});</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>Property:xAxisName let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }]);chart.appendTo('#chart');</p> <p>Property:xAxisName \$("#chart").ejChart({ series: [{ xAxisName : 'secondaryXAxis' }]);});</p>
		<p>Name of the property in the datasource that contains x value for the series.</p> <p>Property:xName let chart: Chart = new Chart({ series: [{ xName: 'x' }]);chart.appendTo('#chart');</p> <p>Property:xName \$("#chart").ejChart({ series: [{ xName : 'x' }]);});</p>
		<p>Specifies the name of the y-axis that has to</p> <p>Property:yAxisName let chart: Chart = new Chart({ series: [{ yAxisName: 'secondaryYAxis' }]);chart.appendTo('#chart');</p> <p>Property:yAxisName \$("#chart").ejChart({ series: [{ yAxisName : 'secondaryYAxis' }]);});</p>

		<p>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 datasource that contains close value for the series.</p> <p>Name of the property in the datasource 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</p> <pre>let chart: Chart = new Chart({ series: [{ close: 'y' }] }); chart.appendTo('#chart');</pre> <p>Property:open</p> <pre>let chart: Chart = new Chart({ series: [{ open: 'y' }] }); chart.appendTo('#chart');</pre> <p>Property:trendLines</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines : [{}] }] }); chart.appendTo('#chart');</pre> <p>Property:highlightSettings</p> <pre>let chart: Chart = new Chart({ series: [{ highlightSettings : {} }] }); chart.appendTo('#chart');</pre> <p>Not applicable.</p>
--	--	---	--

		Options for customizing the appearance of the series/data point on selection . ## marker	Property:selectionSettings \$("#chart").ejChart({ series: [{ selectionSettings : {} }];});	Not applicable.
		visibility of marker	Property:visible \$("#chart").ejChart({ series: [{ marker: { visible: true } }];});	Property:visible let chart: Chart = new Chart({ series: [{ marker: { visible: false } }];});chart.appendTo('#chart');
		Fill for marker	Property:fill \$("#chart").ejChart({ series: [{ marker: { fill : 'red' } }];});	Property:visible let chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }];});chart.appendTo('#chart');
		Opacity for marker	Property:opacity \$("#chart").ejChart({ series: [{ marker: { opacity : 0.5 } }];});	Property:opacity let chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }];});chart.appendTo('#chart');
		Shape of marker	Property:shape \$("#chart").ejChart({ series: [{ marker: { shape : 'Circle' } }];});	Property:shape let chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }];});chart.appendTo('#chart');

		<p>Image Url of marker</p> <p>Property:imageUrl\$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }];});</p>	<p>Property:imageUrl1 et chart: Chart = new Chart({ series: [{ marker: { imageUrl : '' } }];});chart.appendTo('#chart');</p>
		<p>Border of marker</p> <p>Property:border\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }];});</p>	<p>Property:shapelet chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }];});chart.appendTo('#chart');</p>
		<p>Height of marker</p> <p>Property:size.height\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }];});</p>	<p>Property:heightlet chart: Chart = new Chart({ series: [{ marker: { height: 25 } }];});chart.appendTo('#chart');</p>
		<p>Width of marker</p> <p>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }];});</p>	<p>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }];});chart.appendTo('#chart');</p>
		<p>Width of marker</p> <p>Property:size.width\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }];});</p>	<p>Property:widthlet chart: Chart = new Chart({ series: [{ marker: { width: 25 } }];});chart.appendTo('#chart');</p>
		<p>Data Labels of marker</p> <p>Property:marker.dataLabel\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }];});</p>	<p>Property:marker.dataLabellet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }];});</p>

		<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>	
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>
Alignme nt of data label	Property:dataLabel.verticalAlignm ent\$("#chart").ejChart({ series: [{ marker: {dataLabel: { verticalAlignment: 'near' } } }];});	Property:dataLabel. alignmentlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { alignment: 'Near' } } }];});chart.append endTo('#chart');
Border of data label	Property:dataLabel.border\$("#ch art").ejChart({ series: [{ marker: {dataLabel: { border: { color: 'blue', width: 2, opacity: 0.4 } } } }];});	Property:dataLabel. alignmentlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { border: { color: 'blue', width: 2 } } } }];});chart.append endTo('#chart');
Offset for data label	Property:dataLabel.offset\$("#ch art").ejChart({ series: [{ marker: {dataLabel: { offset: { x: 5, y: 6 } } } }];});	Not Applicable
Margin of data label	Property:dataLabel.border\$("#ch art").ejChart({ series: [{ marker: {dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10} } }];});	Property:dataLabel. marginlet chart: Chart = new Chart({ series: [{ marker: { dataLabel: { margin: { top: 10, bottom: 10, left: 10, right: 10 } } } }];});chart.append endTo('#chart');
Font of data label	Property:dataLabel.border\$("#ch art").ejChart({ series: [{ marker: {dataLabel: { font: { fontFamily: 'SegoeUI', fontStyle: 'italic', fontWeight:	Property:dataLabel. marginlet chart: Chart = new Chart({ series: [{ marker:

		<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 ' } } }]]});	Property:dataLabel. templatelet chart: 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	Property:dataLabel. 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	Property:dataLabel. 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	Not Applicable

		<p>Enable wrapping of text for data label</p> <p>Property: <code>dataLabel.enableWrap\$</code></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: <code>dataLabel.showContrastColor\$</code></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: <code>dataLabel.showEdgeLabels\$</code></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>Trendline settings</p> <p>Property: <code>series.trendLines\$</code></p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [] }] }); chart.appendTo('#chart');</pre> <p>Visibility of trendline</p> <p>Property: <code>trendLines.visibility\$</code></p> <pre>("#chart").ejChart({ series: [{ trendLines: [{ visibility: true }] }] });</pre> <p>Not applicable</p> <p>Type of trendline</p> <p>Property: <code>trendLines.type\$</code></p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [{ type: 'Polynomial' }] }] }); chart.appendTo('#chart');</pre> <p>Name of trendline</p> <p>Property: <code>trendLines.name\$</code></p> <pre>("#chart").ejChart({ series: [{ trendLines: [{ name: 'trendLine' }] }] });</pre> <p>Property: <code>trendLines.name\$</code></p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [{ name: 'trendLine' }] }] }); chart.appendTo('#chart');</pre>
--	--	--

		<pre>rt.appendTo('#chart'));</pre>	
Period of trendLine	Property:trendLines.period\$ (" #chart").ejChart({ series: [{ trendLines: [period: 45] }]});	Property:trendLines.period <pre>let chart: Chart = new Chart({ series: [{ trendLines: [period: 45] }] }); chart.append To('#chart');</pre>	
Polynomial order for Polynomial trendLines	Property:trendLines.polynomialOrder\$ (" #chart").ejChart({ series: [{ trendLines: [polynomialOrder: 3] }]});	Property:trendLines.polynomialOrder <pre>let chart: Chart = new Chart({ series: [{ trendLines: [polynomialOrder: 3] }] }); chart.appendT o('#chart');</pre>	
Backward forecast trendLines	Property:trendLines.backwardforecast\$ (" #chart").ejChart({ series: [{ trendLines: [backwardforecast: 3] }]});	Property:trendLines.backwardforecast <pre>let chart: Chart = new Chart({ series: [{ trendLines: [backwardforecast: 3] }] }); chart.appendT o('#chart');</pre>	
Forward forecast trendLines	Property:trendLines.forwardForecast\$ (" #chart").ejChart({ series: [{ trendLines: [forwardForecast: 3] }]});	Property:trendLines.forwardForecast <pre>let chart: Chart = new Chart({ series: [{ trendLines: [forwardForecast: 3] }] }); chart.appendT o('#chart');</pre>	
Fill trendLines	Property:trendLines.fill\$ (" #chart").ejChart({ series: [{ trendLines: [fill: 'EEFFCC'] }]});	Property:trendLines.fill <pre>let chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEFFCC'] }] }); chart. appendTo('#chart');</pre>	
Width for	Property:trendLines.width\$ (" #chart").ejChart({	Property:trendLines.width <pre>let chart: Chart =</pre>	

		<p>trendLines</p> <pre>series: [{ trendLines: [width: 2]}]];</pre>	<pre>new Chart({ series: [{ trendLines: [width: 2]}]});chart.appendTo('#chart');</pre>
		<p>Intercept value for trendLines</p> <pre>Property:trendLines.intercept\$ ("#chart").ejChart({ series: [{ trendLines: [intercept: 2]}]]);</pre>	<p>Property:trendLines.intercept</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [intercept: 2]}]});chart.appendTo('#chart');</pre>
		<p>Legend shape for trendLines</p> <p>Not Applicable</p>	<p>Property:trendLines.legendShape</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [legendShape: 'Rectangle']}]});chart.appendTo('#chart');</pre>
		<p>Animation settings for trendLines</p> <p>Not Applicable</p>	<p>Property:trendLines.animation</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [animation: { enable: true }]}]});chart.appendTo('#chart');</pre>
		<p>Marker settings for trendLines</p> <p>Not Applicable</p>	<p>Property:trendLines.marker</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [{marker: { visible: true } }]}]});chart.appendTo('#chart');</pre>
		<p>Tooltip for trendLines</p> <pre>Property:trendLines.tooltip\$ ("#chart").ejChart({ series: [{ trendLines: [{ tooltip: { } }]}]]);</pre>	<p>Property:trendLines.enableTooltip</p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [{enableTooltip: true }]}]});chart.appendTo('#chart');</pre>

		<p>Dash Array for trendLines es</p> <p>Property: trendLines.dashArray <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ dashArray: '10, 5' }] }]});</pre></p> <p>Visible on legend end for trendLines es</p> <p>Property: trendLines.visibleOnLegend <pre>\$("#chart").ejChart({ series: [{ trendLines: [{ visibleOnLegend: true }] }]});</pre></p> <p>## Striplines</p> <p>Behaviour Default behaviour for striplines border for stripline Background color for stripline Start value for</p> <p>API in Essential JS 1 Property: primaryXAxis.striplines <pre>\$("#chart").ejChart({ primaryXAxis: { striplines: [{ visible: true }] } });</pre></p> <p>API in Essential JS 2 Property: primaryXAxis.striplines <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ visible: true }] } }); chart.appendTo('#chart');</pre></p> <p>Property: striplines.borderColor <pre>\$("#chart").ejChart({ primaryXAxis: { striplines: [{ border: { color: 'red', width: 2 } }] } }); chart.appendTo('#chart');</pre></p> <p>Property: striplines.borderColor <pre>let chart: Chart = new Chart({ primaryXAxis: { striplines: [{ color: 'red' }] } }); chart.appendTo('#chart');</pre></p> <p>Property: striplines.start <pre>\$("#chart").ejChart({ primaryXAxis: {</pre></p> <p>Not Applicable. Not Applicable.</p>
--	--	---

		<pre> size: 10 }}});chart.appendTo ('#chart'); </pre>
ZIndex of stripLine	Property:stripLines.zIndex\$ <pre> ("#chart").ejChart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }] } } } } } </pre>	Property:stripLines.size <pre> let chart: Chart = new Chart({ primaryXAxis: { stripLines: [{ zIndex: 'Behind' }] } } } } } chart.appendTo ('#chart'); </pre>
Font style of stripLine	Property:stripLines.fontStyle\$ <pre> ("#chart").ejChart({ primaryXAxis: { stripLines: [{ fontStyle: {} }] } } } } } </pre>	Property:stripLines.textStyle <pre> let 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	Property:primaryXAxis.multilevelLabels\$ <pre> ("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }] } } } } } </pre>	Property:primaryXAxis.multilevelLabels <pre> let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }] } } } } } chart.appendTo ('#chart'); </pre>
Default behavior for multilevels	Property:primaryXAxis.multilevelLabels\$ <pre> ("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ visible: true }] } } } } } </pre>	Property:primaryXAxis.multilevelLabels <pre> let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ }] } } } } } chart.appendTo ('#chart'); </pre>
Text alignment for multil	Property:multiLevelLabels.textAlignment\$ <pre> ("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ textAlignment: 'Near' }] } } } } } </pre>	Property:multilevelLabels.alignment <pre> let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{alignment: 'Near' </pre>

		<pre> evelLa bels }}}});chart.appendT o('#chart'); </pre>	
Text overfl ow for multil evelLa bels	<pre> Property:multiLevelLabels.text s.overflowlet chart: OverFlow\$("#chart").ejCh art({ primaryXAxis: { multilevelLabels: [{ textOverFlow: 'Trim' }}}}); </pre>	<pre> Property:multiLevelLabel s.overflowlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{overflow: 'Trim' }}}});chart.appendT o('#chart'); </pre>	
Borde r for multil evelLa bels	<pre> Property:multiLevelLabels.bor der\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } } } } }); </pre>	<pre> Property:multiLevelLabel s.borderlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ border: { width: 2, color: 'red', type: 'brace' } }}}});chart.appendT o('#chart'); </pre>	
Start value for label	<pre> Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ start: 45 } } } }); </pre>	<pre> Property:multiLevelLabel s.categories.startlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ start: 45}] } }}}});chart.appendT o('#chart'); </pre>	
End value for label	<pre> Property:multiLevelLabels.start \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ end: 45 } } } }); </pre>	<pre> Property:multiLevelLabel s.categories.endlet chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ end: 45}] } }}}});chart.appendT o('#chart'); </pre>	
Text for label	<pre> Property:multiLevelLabels.text \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ text: 'Start' } } } }); </pre>	<pre> 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 let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 20 }] } }] } });chart.appendTo('#chart'); </pre>
level of labels	<pre> Property:multiLevelLabels.level \$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } }); </pre>	<pre> Property:multiLevelLabels.level let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } }] } });chart.appendTo('#chart'); </pre> <p>Not applicable. Categories are used</p>
## 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: () { } }); </pre> <p>Not applicable</p>
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('ch art');chart.appendTo('#ch art'); </pre>
AddSeries	Not Applicable	<pre> Property:chart.addSeries()let chart: Chart = new Chart({});chart.appendTo(</pre>

		<pre> '#chart');chart.addSeries (); Property:chart.removeSeries()le t chart: Chart = new Chart({});chart.appendTo('#chart');chart.removeSer ies(); </pre>
	Remo veSeri es	Not Applicable
	## Events	
	Behavi our	API in Essential JS 1 API in Essential JS 2
	Fires on annot ation click	<pre> Property:annotationClick\$("#cha rt").ejChart({ annotationClick: () { }}); </pre> Not applicable
	Fires after anima tion	<pre> Property:animationComplete\$("# chart").ejChart({ animationComplete: () { }}); </pre> Property:animationC omplete()let chart: Chart = new Chart({ animationComple te: () => { }});chart.append To('#chart');
	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> Property:axisLabelRe nder()let chart: Chart = new Chart({ axisLabelRender: () => { }});chart.append To('#chart');
	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</p> <p>on Property:chartMouseLeave\$("#chart").ejChart({ chart chartMouseLeave: () { }}); mouse leave</p> <p>Fires</p> <p>on Property:chartDoubleClick\$("#chart").ejChart({ before chartDoubleClick: () { chart }}); doubl e click</p> <p>Fires</p> <p>on Not Applicable chart mouse up</p> <p>Fires</p> <p>on Not Applicable chart mouse down</p> <p>Fires</p> <p>during the calculation of chart area bounds. You can use this event to customize the</p>	<p>Property:chartMouseLeave let chart: Chart = new Chart({ chartMouseLeave: () => { }});chart.appendTo('#chart');</p> <p>Not applicable</p> <p>Property:chartmouseUp let chart: Chart = new Chart({ chartmouseUp: () => { }});chart.appendTo('#chart');</p> <p>Property:chartmouseDown let chart: Chart = new Chart({ chartmouseDown: () => { }});chart.appendTo('#chart');</p> <p>Not applicable</p>
--	--	--	---

		<p>bound s of chart area</p> <p>Fires when the dragging is started</p> <p>Fires while dragging</p> <p>Fires when the dragging is completed</p> <p>Fires when chart is destroyed completely.</p> <p>Fires after chart is created.</p> <p>Fires before rendering the</p>	<p>Property:dragStart\$("#chart").ejChart({ dragStart: () { }});</p> <p>Property:dragging\$("#chart").ejChart({ dragging: () { }});</p> <p>Property:dragComplete\$("#chart").ejChart({ dragEnd: () { }});</p> <p>Property:destroy\$("#chart").ejChart({ destroy: () { }});</p> <p>Property:create\$("#chart").ejChart({ create: () { }});</p> <p>Property:displayTextRendering\$("#chart").ejChart({ displayTextRendering: () { }});</p>	<p>Not applicable</p> <p>Not applicable</p> <p>Property:dragComplete\$("#chart").ejChart({ dragComplete: () => { }});chart.appendTo('#chart');</p> <p>Not applicable</p> <p>Property:loaded\$("#chart").ejChart({ loaded: () => { }});chart.appendTo('#chart');</p> <p>Property:textRendering\$("#chart").ejChart({ textRender: () => { }});chart.appendTo('#chart');</p>
--	--	--	--	--

		<p>data labels.</p> <p>Fires when error bar is rendering.</p> <p>Property: <code>errorBarRendering\$</code> (<code>"#chart").ejChart({ errorBarRendering: () { }}</code>);</p> <p>Not applicable</p> <p>Fires during the calculation of legend bounds.</p> <p>Property: <code>legendBoundsCalculate\$</code> (<code>"#chart").ejChart({ legendBoundsCalculate: () { }}</code>);</p> <p>Not applicable</p> <p>Fires on clicking the legend item.</p> <p>Property: <code>legendItemClick\$</code> (<code>"#chart").ejChart({ legendItemClick: () { }}</code>);</p> <p>Not applicable</p> <p>Fires when moving mouse over legend item.</p> <p>Property: <code>legendItemMouseMove\$</code> (<code>"#chart").ejChart({ legendItemMouseMove: () { }}</code>);</p> <p>Not applicable</p> <p>Fires before rendering the legend item.</p> <p>Property: <code>legendItemRendering\$</code> (<code>"#chart").ejChart({ legendItemRendering: () { }}</code>);</p> <p>Property: <code>legendRender</code></p> <pre>let chart; Chart = new Chart({ legendRender: () => { chart.append('chart'); } });</pre> <p>Fires before loading the chart.</p> <p>Property: <code>load\$</code> (<code>"#chart").ejChart({ load: () { }}</code>);</p> <p>Property: <code>load</code></p> <pre>let chart; Chart = new Chart({ load: () => { chart.append('chart'); } });</pre>
--	--	--

		<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\$("#chart").ejChart({ <pre>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\$("#chart").ejChart({ <pre>preRender: () { }});</pre> </p> <p>Fires when point render .</p> <p>Not Applicable</p> <p>Fires after selected the data in chart.</p> <p>Property:rangeSelected\$("#chart").ejChart({ <pre>rangeSelected: () { }});</pre> </p> <p>Fires after selecti</p> <p>Property:seriesRegionClick\$("#chart").ejChart({</p>	<p>Property:axisMultiLabelRender let <pre>chart: Chart = new Chart({ axisMultiLabelRender : () => { }});chart.appendTo('#chart');</pre> </p> <p>Property:pointClick <pre>let chart: Chart = new Chart({ pointClick : () => { }});chart.appendTo('#chart');</pre> </p> <p>Property:pointMove <pre>let chart: Chart = new Chart({ pointMove : () => { }});chart.appendTo('#chart');</pre> </p> <p>Not applicable</p> <p>Property:pointRender let <pre>chart: Chart = new Chart({ pointRender : () => { }});chart.appendTo('#chart');</pre> </p> <p>Not applicable</p> <p>Not applicable</p>
--	--	--	---

		<pre> ng a seriesRegionClick: () { series. }}}; </pre>	
		<pre> Fires before Property:seriesRendering\$ ("#cha 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\$ ("#ch the art").ejChart({ marke symbolRendering: () { }}}; r symbo ls. </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:trendlineRendering\$ ("# ing chart").ejChart({ the trendlineRendering: () { trendli }}}; ne </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:titleRendering\$ ("#char ing t").ejChart({ the titleRendering: () { }}}; Chart title. </pre>	<p>Not applicable</p>
		<pre> Fires before render Property:subTitleRendering\$ ("#c ing hart").ejChart({ the subTitleRendering: () { Chart }}}; sub title. </pre>	<p>Not applicable</p>
		<pre> Fires before Property:toolTipInitialize\$ ("#cha render rt").ejChart({ </pre>	<p>Property:tooltipRend</p> <pre> er let chart: Chart = new </pre>

		<p>ing the tooltip .</p> <p>Fires before rendering crosshair tooltip in axis</p> <p>Fires before rendering trackball tooltip .</p> <p>Event triggered when scroll starts.</p> <p>Event triggered when scroll ends.</p> <p>Event triggered when scroll changes.</p> <p>Fires while performing</p>	<pre> toolTipInitialize: () { Chart({ tooltipRender : () => { }});chart.append To('#chart'); Property:trackAxisToolTip\$ ("#chart").ejChart({ trackAxisToolTip: () { }}); Property:trackToolTip\$ ("#chart").ejChart({ trackToolTip: () { }}); Property:scrollStart let chart: Chart = new Chart({ .ejChart({ scrollStart: () scrollStart : () => { }});chart.append To('#chart'); Property:scrollEndle t chart: Chart = new Chart({ scrollEnd: () => { }});chart.append To('#chart'); Property:scrollChang e let chart: Chart = new Chart({ scrollChange: () => { }});chart.append To('#chart'); Property:zoomComple te let chart: Chart = new Chart({ </pre>
--	--	--	--

		<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: { 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 Not Applicable </pre>
--	--	---	--

		<pre> chart : background. ../images/chart/w heat.png'}}); </pre>
Customizing border of the chart	Property: border <pre> \$("#container").ejChart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}}); </pre>	Property: border <pre> let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.ap pendTo('#chart'); </pre>
This provides options for customizing export settings	Property: exportSettings <pre> \$("#container").ejChart({ exportSettings: { filename : "chart", angle: '45' }}); </pre>	Property: export() <pre> let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.ap pendTo('#chart');char t.export(type, fileName); </pre>
ChartArea customization	Property: chartArea <pre> \$("#container").ejChart({ chartArea: { background: 'transparent', border: { opacity: 0.3, color: 'red', width: 2}}}); </pre>	Property: chartArea <pre> 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	Property: alternateGridBand <pre> \$("#container").ejChart({ primaryYAxis: { alternateGridBand: { even: { fill: 'red' }}}}); </pre>	Not applicable
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.appendTo('#ch art'); </pre>
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: '' </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	Property: <code>enableAutoIntervalOnZooming\$("#container").ejChart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});</code>	Property: <code>enableAutoIntervalOnZooming</code> <code>let chart: Chart = new Chart({ primaryYAxis: { enableAutoIntervalOnZooming: true }});chart.appendTo('#chart');</code>
Font style for primaryYAxis	Property: <code>font\$("#container").ejChart({ primaryYAxis: { font: { fontFamily: 'Calibri', fontStyle: 'italic', fontWeight: '', opacity: 0.5, size: 12 } }});</code>	Property: <code>titleStyle</code> <code>let chart: Chart = new Chart({ primaryYAxis: { titleStyle: { } }});chart.appendTo('#chart');</code>
Indexed for category axis	Property: <code>isIndexed\$("#container").ejChart({ primaryYAxis: { isIndexed: true }});</code>	Property: <code>isIndexed</code> <code>let chart: Chart = new Chart({ primaryYAxis: { isIndexed: true }});chart.appendTo('#chart');</code>
Interval type for date time axis	Property: <code>intervalType\$("#container").ejChart({ primaryYAxis: { intervalType: 'Auto' }});</code>	Property: <code>intervalType</code> <code>let chart: Chart = new Chart({ primaryYAxis: { intervalType: 'Auto' }});chart.appendTo('#chart');</code>
Inversed axis	Property: <code>isInversed\$("#container").ejChart({ primaryYAxis: { isInversed: true }});</code>	Property: <code>isInversed</code> <code>let chart: Chart = new Chart({ primaryYAxis: { isInversed: true }});chart.appendTo('#chart');</code>
Custom label format	Property: <code>labelFormat\$("#container").ejChart({ primaryYAxis: { labelFormat: '{value}K' }});</code>	Property: <code>labelFormat</code> <code>let chart: Chart = new Chart({ primaryYAxis: { labelFormat: '{value}K' }});chart.appendTo('#chart');</code>
labelIntersect Action	Property: <code>labelIntersectAction\$("#container").ejChart({ primaryYAxis: { labelIntersectAction: 'trim' }});</code>	Property: <code>labelIntersectAction</code> <code>let 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 <code>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	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: 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 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 primary YAxis	Property:zoomPosition\$("#container").ejChart({ axes: [{ zoomPosition: 0.3 }] });	Property:zoomPositionlet chart: Chart = new Chart({ axes: [{ zoomPosition: 0.3 }] });chart.appendTo('#chart');									
labelBorder of primary YAxis	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 primary YAxis	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 primary YAxis	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 customization	<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>

		<p>Maximum radius of Bubble series</p> <p>Property: <code>bubbleOptions.maxRadius</code></p> <pre>let chart: Chart = new Chart({ series: [{ bubbleOptions: { maxRadius: 10 } }]});</pre>	<p>Property: <code>maxRadius</code></p> <pre>let chart: Chart = new Chart({ series: [{ maxRadius: 2 }]});chart.appendTo('#chart');</pre>
		<p>bullFillColor</p> <p>Property: <code>bullFillColor</code></p> <pre>let chart: Chart = new Chart({ series: [{ bullFillColor: 'red' }]});</pre>	<p>Property: <code>bullFillColor</code></p> <pre>let chart: Chart = new Chart({ series: [{ bullFillColor: 'red' }]});chart.appendTo('#chart');</pre>
		<p>Cardinal spline tension for spline series</p> <p>Property: <code>cardinalSplineTension</code></p> <pre>let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]});</pre>	<p>Property: <code>cardinalSplineTension</code></p> <pre>let chart: Chart = new Chart({ series: [{ cardinalSplineTension: 0.5 }]});chart.appendTo('#chart');</pre>
		<p>Column Width for rectangle series</p> <p>Property: <code>columnWidth</code></p> <pre>let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]});</pre>	<p>Property: <code>columnWidth</code></p> <pre>let chart: Chart = new Chart({ series: [{ columnWidth: 0.5 }]});chart.appendTo('#chart');</pre>
		<p>Column spacing for rectangle series</p> <p>Property: <code>columnSpacing</code></p> <pre>let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]});</pre>	<p>Property: <code>columnSpacing</code></p> <pre>let chart: Chart = new Chart({ series: [{ columnSpacing: 0.5 }]});chart.appendTo('#chart');</pre>
		<p>Top left radius for rectangle series</p> <p>Property: <code>cornerRadius.topLeft</code></p> <pre>let chart: Chart = new Chart({ series: [{ topLeft: 0 }]});</pre>	<p>Property: <code>cornerRadius.topLeft</code></p> <pre>let chart: Chart = new Chart({ series: [{ topLeft: 0 }]});chart.appendTo('#chart');</pre>

		<p>topRight radius for rectangle series</p> <p>Property:cornerRadius.topRight\$ ("#chart").ejChart({ series: [{ topRight: 0 }];});</p>	<p>Property:cornerRadius.topRight\$ let chart: Chart = new Chart({ series: [{ topRight: 0 }];});chart.append To('#chart');</p>
		<p>bottomRight radius for rectangle series</p> <p>Property:cornerRadius.bottomRight\$ ("#chart").ejChart({ series: [{ bottomRight: 0 }];});</p>	<p>Property:cornerRadius.bottomRight\$ let chart: Chart = new Chart({ series: [{ bottomRight: 0 }];});chart.append To('#chart');</p>
		<p>bottomLeft radius for rectangle series</p> <p>Property:cornerRadius.bottomLeft\$ ("#chart").ejChart({ series: [{ bottomLeft: 0 }];});</p>	<p>Property:cornerRadius.bottomLeft\$ let chart: Chart = new Chart({ series: [{ bottomLeft: 0 }];});chart.append To('#chart');</p>
		<p>DashArray property</p> <p>Property:dashArray\$("#chart") .ejChart({ series: [{ dashArray: '10, 5' }];});</p>	<p>Property:dashArray\$ let chart: Chart = new Chart({ series: [{ dashArray: '10, 5' }];});chart.append To('#chart');</p>
		<p>DataSource for series</p> <p>Property:dataSource\$("#chart") .ejChart({ series: [{ dataSource: [] }];});</p>	<p>Property:dashArray\$ let chart: Chart = new Chart({ series: [{ dataSource: [] }];});chart.append To('#chart');</p>
		<p>Draw type for Polar series</p> <p>Property:drawType\$("#chart") .ejChart({ series: [{ drawType: 'Line' }];});</p>	<p>Property:drawType\$ let chart: Chart = new Chart({ series: [{ drawType: 'Line' }];});chart.append To('#chart');</p>
		<p>EmptyPointSettings</p> <p>Property:emptyPointSettings.visible\$ ("#chart").ejChart({ series: [{</p>	<p>Not Applicable</p>

		gs for series <pre>emptyPointSettings: { visible: false } }]);</pre>	
	Empty Point Display mode	Property:emptyPointSettings.displayMode <pre>let chart: Chart = new Chart({ series: [{ displayMode: 'gap' }]); chart.appendTo('#chart');</pre>	
	Empty Point color	Property:emptyPointSettings.fill <pre>let chart: Chart = new Chart({ series: [{ fill: 'red' }]); chart.appendTo('#chart');</pre>	
	Empty Point Border	Property:emptyPointSettings.border <pre>let chart: Chart = new Chart({ series: [{ emptyPointSettings: { color: 'red', width: 2 } }]); chart.appendTo('#chart');</pre>	
	Enable animation for series	Property:animation.enable <pre>let chart: Chart = new Chart({ series: [{ animation: { enable: false } }]); chart.appendTo('#chart');</pre>	
	Animation duration for series	Property:animation.duration <pre>let chart: Chart = new Chart({ series: [{ animation: { duration: 1000 } }]); chart.appendTo('#chart');</pre>	
	Animation delay for series	Property:animation.duration <pre>let chart: Chart = new Chart({ series: [{ animation: { duration: 1000 } }]); chart.appendTo('#chart');</pre>	

		<pre> delay: 100 }}});chart.appe ndTo('#chart'); </pre>
Drag settings for series	<pre> Property:dragSettings\$("#chart").ejChart({ series: [{ dragSettings: { mode: 'X' } }]}); </pre>	Not Applicable
Errorbar settings for series	<pre> Property:errorBarSettings\$("#chart").ejChart({ series: [{ errorBarSettings: { } }]}); </pre>	<pre> Property:errorBarSettingslet chart: Chart = new Chart({ series: [{ errorBarSettings: { } }]}); </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:errorBarSettingslet chart: Chart = new Chart({ series: [{ opacity: 0.7 }]}); </pre>
Outlier settings of series	<pre> Property:outLierSettings\$("#chart").ejChart({ series: [{ outLierSettings: { shape: 'rectangle', size: { height: 30, width: 20 } } }]}); </pre>	Not Applicable
Palette	<pre> Property:palette\$("#chart").ejChart({ series: [{ </pre>	<pre> Property:pointColorMappinglet 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> <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-</p>	<pre> visibleOnLegend : true }]);}); Property:splineType let chart: Chart = new Chart({ legendSettings: [{ splineType: 'Natural' }];});chart.append To('#chart'); Property:xAxisName1 let chart: Chart = new Chart({ series: [{ xAxisName: 'secondaryXAxis' }];});chart.append To('#chart'); Property:xName1 let chart: Chart = new Chart({ series: [{ xName: 'x' }];});chart.append To('#chart'); Property:yAxisName1 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.appendTo('#chart'); Property:openlet chart: Chart = new Chart({ series: [{ open: 'y' }];});chart.appendTo('#chart'); Property:trendLines1 et chart: Chart = new Chart({ series: [{ trendLines : [{}]}];});chart.appendTo('#chart'); Property:highlightSettings\$("#chart").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</p> <pre>\$("#chart").ejChart({ series: [{ selectionSettings : {} }];});</pre> <p>Not applicable.</p> <p>Property:visible</p> <pre>let chart: Chart = new Chart({ series: [{ marker: { visible: false } }];});chart.append('#chart');</pre> <p>Property:visible</p> <pre>let chart: Chart = new Chart({ series: [{ marker: { fill : 'red' } }];});chart.append('#chart');</pre> <p>Property:opacity</p> <pre>let chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }];});chart.append('#chart');</pre> <p>Property:shape</p> <pre>let chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }];});chart.append('#chart');</pre>
--	--	---	---

		<p>Image Url of marker</p> <p>Property:imageUrl <code>ejChart({ series: [{ marker: { imageUrl : '' } }];});</code></p> <p>Property:imageUrl <code>let chart: Chart = new Chart({ series: [{ marker: { imageUrl : '' } }];});chart.appendTo('#chart');</code></p> <p>Border of marker</p> <p>Property:border <code>ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }];});</code></p> <p>Property:shape <code>let chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }];});chart.appendTo('#chart');</code></p> <p>Height of marker</p> <p>Property:size.height <code>ejChart({ series: [{ marker: { size: { height: 30 } } }];});</code></p> <p>Property:height <code>let chart: Chart = new Chart({ series: [{ marker: { height: 25 } }];});chart.appendTo('#chart');</code></p> <p>Width of marker</p> <p>Property:size.width <code>ejChart({ series: [{ marker: { size: { width: 30 } } }];});</code></p> <p>Property:width <code>let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }];});chart.appendTo('#chart');</code></p> <p>Width of marker</p> <p>Property:size.width <code>ejChart({ series: [{ marker: { size: { width: 30 } } }];});</code></p> <p>Property:width <code>let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }];});chart.appendTo('#chart');</code></p> <p>Data Label Setting of marker</p> <p>Property:marker.dataLabel <code>ejChart({ series: [{ marker: { dataLabel: { } } }];});</code></p> <p>Property:marker.dataLabel <code>let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }];});</code></p>
--	--	---

		<pre>]});});chart.appendTo('#chart'); </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('#chart'); </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('#chart'); </pre>
Fill color of data label	<pre> Property:dataLabel.fill\$("#chart").ejChart({ series: [{ marker: {dataLabel: {fill: 'pink' } } }]});}); </pre>	<pre> Property:dataLabel.filllet chart:Chart = new Chart({ series: [{ marker: {dataLabel: {fill: 'pink' } } }]});});chart.appendTo('#chart'); </pre>
Opacity of data label	<pre> Property:dataLabel.opacity\$("#chart").ejChart({ series: [{ marker: {dataLabel: {opacity: 0.6 } } }]});}); </pre>	<pre> Property:dataLabel.opacitylet chart:Chart = new Chart({ series: [{ marker: {dataLabel: {opacity: 0.4 } } }]});});chart.appendTo('#chart'); </pre>
Text position of data label	<pre> Property:dataLabel.textPosition\$("#chart").ejChart({ series: [{ marker: {dataLabel: {textPosition: 'middle' } } }]});}); </pre>	<pre> Property:dataLabel.positionlet chart:Chart = new Chart({ series: [{ marker: {dataLabel: {position: 'Top' } } }]});});chart.appendTo('#chart'); </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</p> <p>Property: <code>dataLabel.enableWrap\$</code> (<code>"#chart").ejChart({ series: [{ marker: {dataLabel: { enableWrap: true }} }]}]</code>);</p> <p>Not Applicable</p> <p>To show contrast color for data label</p> <p>Property: <code>dataLabel.showContrastColor\$</code> (<code>"#chart").ejChart({ series: [{ marker: {dataLabel: { showContrastColor: true }} }]}]</code>);</p> <p>Not Applicable</p> <p>To show edge label for data label</p> <p>Property: <code>dataLabel.showEdgeLabels\$</code> (<code>"#chart").ejChart({ series: [{ marker: {dataLabel: { showEdgeLabels: true }} }]}]</code>);</p> <p>Not Applicable</p> <p>## TrendLines</p> <p>Behaviour</p> <p>API in Essential JS 1 API in Essential JS 2</p> <p>Trend lines settings</p> <p>Property: <code>series.trendLines\$</code> (<code>"#chart").ejChart({ series: [{ trendLines: []}]</code>);</p> <p>Property: <code>series.trendLines</code></p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: []}] });chart.appendTo('#chart');</pre> <p>Visibility of trend line</p> <p>Property: <code>trendLines.visibility\$</code> (<code>"#chart").ejChart({ series: [{ trendLines: [visibility: true]}]</code>);</p> <p>Not applicable</p> <p>Type of trend line</p> <p>Property: <code>trendLines.type\$</code> (<code>"#chart").ejChart({ series: [{ trendLines: [type: 'linear']}]</code>);</p> <p>Property: <code>trendLines.type</code></p> <pre>let chart: Chart = new Chart({ series: [{ trendLines: [type: 'Polynomial']}] });chart.appendTo('#chart');</pre> <p>Name of trend line</p> <p>Property: <code>trendLines.name\$</code> (<code>"#chart").ejChart({ series: [{ trendLines: [name: 'trendLine']}]</code>);</p> <p>Property: <code>trendLines.name</code></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.polynomial Order\$("#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 for trendLines	<pre> Property:trendLines.backwardf orecast\$("#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 for trendLines	<pre> Property:trendLines.forwardFo recast\$("#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 for 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']}]})];chart. appendTo('#chart'); </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\$ (" #chart").ejChart({ 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>maxim um text width for label</p> <pre>Property:multiLevelLabels.maximumTextWidth\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ maximumTextWidth: 10 }] } });</pre> <p>level of labels</p> <pre>Property:multiLevelLabels.level\$("#chart").ejChart({ primaryXAxis: { multilevelLabels: [{ level: 2 }] } });</pre> <p>## Methods</p> <p>Behav iour</p> <p>API in Essential JS 1 API in Essential JS 2</p> <p>anima tion for series</p> <pre>Property:chart.animate\$("#chart").ejChart({ animate: () { { } });</pre> <p>Redra w for chart</p> <pre>Property:chart.redraw\$("#chart").ejChart({ redraw: () { { } });</pre> <p>Expor t</p> <pre>Property:chart.export()\$("#chart").ejChart({ export: () { { } });</pre> <p>Print</p> <pre>Property:chart.print()\$("#chart").ejChart({ print: () { { } });</pre> <p>AddSe ries</p> <pre>Property:chart.addSeries()let chart: Chart = new Chart({});chart.appendTo('#chart');chart.addSeries ();</pre> <p>Property:multiLevelLabels.categories.maximumTextWidth</p> <pre>let chart: Chart = new Chart({ primaryXAxis: { multilevelLabels: [{ categories: [{ maximumTextWidth: 20 }] } } });chart.appendTo('#chart');</pre> <p>Not applicable. Categories are used</p>
--	--	--

		<p>Removes series from the chart. Not applicable in EJ2.</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 axis range calculation</p> <p>Property:axesRangeCalculate\$("#chart").ejChart({ axesRangeCalculate: () { }});</p> <p>Property:axisRangeCalculated()let chart: Chart = new Chart({ axisRangeCalculated: () => { }});chart.appendTo('#chart');</p>
		<p>Fires on axis title rendering</p> <p>Property:axisTitleRendering\$("#chart").ejChart({ axisTitleRendering: () { }});</p> <p>Not applicable</p>
		<p>Fires on after chart resize</p> <p>Property:afterResize\$("#chart").ejChart({ afterResize: () { }});</p> <p>Not applicable</p>
		<p>Fires on before chart resize</p> <p>Property:beforeResize\$("#chart").ejChart({ beforeResize: () { }});</p> <p>Property:resizedlet chart: Chart = new Chart({ resized: () => { }});chart.appendTo('#chart');</p>
		<p>Fires on chart click</p> <p>Property:chartClick\$("#chart").ejChart({ chartClick: () { }});</p> <p>Property:chartMouseClicklet chart: Chart = new Chart({ chartMouseClick: () => { }});chart.appendTo('#chart');</p>
		<p>Fires on chart mouse move</p> <p>Property:chartMouseMove\$("#chart").ejChart({ chartMouseMove: () { }});</p> <p>Property:chartMouseMoveMovelet chart: Chart = new Chart({ chartMouseMove: () => { }});chart.appendTo('#chart');</p>
		<p>Fires on chart mouse leave</p> <p>Property:chartMouseLeave\$("#chart").ejChart({ chartMouseLeave: () { }});</p> <p>Property:chartMouseLeavelet chart: Chart = new Chart({ chartMouseLeave: () => { }}</p>

		<pre> });chart.append To('#chart'); Fires on Property:chartDoubleClick\$("#cha before rt").ejChart({ chart chartDoubleClick: () { Not applicable double }}); click Fires on Not Applicable chart mouse up Fires on Not Applicable chart mouse down Fires during the calcula tion of chart area bound s. You can use this event to custo mize the bound s of chart area Property:chartAreaBoundsCalculat e\$("#chart").ejChart({ chartAreaBoundsCalculate: () { }}); Not applicable </pre>
--	--	---

		<p>Fires when the dragging is started</p> <p>Property:dragStart\$("#chart").ejChart({ dragStart: () { } });</p> <p>Not applicable</p>
		<p>Fires while dragging</p> <p>Property:dragging\$("#chart").ejChart({ dragging: () { } });</p> <p>Not applicable</p>
		<p>Fires when the dragging is completed</p> <p>Property:dragComplete\$("#chart").ejChart({ dragEnd: () { } });</p> <p>Property:dragComplete\$("#chart").ejChart({ dragComplete: () => { } });chart.appendTo("#chart");</p>
		<p>Fires when chart is destroyed.</p> <p>Property:destroy\$("#chart").ejChart({ destroy: () { } });</p> <p>Not applicable</p>
		<p>Fires after chart is created.</p> <p>Property:create\$("#chart").ejChart({ create: () { } });</p> <p>Property:loaded\$("#chart").ejChart({ loaded: () => { } });chart.appendTo("#chart");</p>
		<p>Fires before rendering the data labels.</p> <p>Property:displayTextRendering\$("#chart").ejChart({ displayTextRendering: () { } });</p> <p>Property:textRender\$("#chart").ejChart({ textRender: () => { } });chart.appendTo("#chart");</p>
		<p>Fires when error bar is</p> <p>Property:errorBarRendering\$("#chart").ejChart({ errorBarRendering: () { } });</p> <p>Not applicable</p>

		<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. chart.appendTo('#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 . Not Applicable Fires after selecting 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.appendTo('#chart'); Property:pointClick let chart: Chart = new Chart({ pointClick: () => { // ... } });chart.appendTo('#chart'); Property:pointMove let chart: Chart = new Chart({ pointMove: () => { // ... } });chart.appendTo('#chart'); Not applicable Property:pointRender let chart: Chart = new Chart({ pointRender: () => { // ... } });chart.appendTo('#chart'); Not applicable Not applicable Property:seriesRender let chart: Chart = new Chart({ seriesRender: () => { // ... } });chart.appendTo('#chart');</pre>
--	--	--	--

		<p>ing a series.</p> <pre> seriesRender : () => { }});chart.append To('#chart');</pre>	
	<p>Fires before rendering the marker symbol.</p>	<pre> Property:symbolRendering\$("#chart").ejChart({ symbolRendering: () { }});</pre>	Not applicable
	<p>Fires before rendering the trendline.</p>	<pre> Property:trendlineRendering\$("#chart").ejChart({ trendlineRendering: () { }});</pre>	Not applicable
	<p>Fires before rendering the Chart title.</p>	<pre> Property:titleRendering\$("#chart").ejChart({ titleRendering: () { }});</pre>	Not applicable
	<p>Fires before rendering the Chart subtitle.</p>	<pre> Property:subTitleRendering\$("#chart").ejChart({ subTitleRendering: () { }});</pre>	Not applicable
	<p>Fires before rendering the tooltip.</p>	<pre> Property:tooltipInitialize\$("#chart").ejChart({ tooltipInitialize: () { }});</pre>	<pre> Property:tooltipRender er let chart: Chart = new Chart({ tooltipRender : () => { }});chart.append To('#chart');</pre>
	<p>Fires before rendering the crosshair.</p>	<pre> Property:trackAxisToolTip\$("#chart").ejChart({ trackAxisToolTip: () { }});</pre>	Not applicable

		<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: () { }});</p> <p>Property:scrollStart\$("#chart"). ejChart({ scrollStart: () { }});</p> <p>Property:scrollEnd\$("#chart"). ejChart({ scrollEnd: () { }});</p> <p>Property:scrollChange\$("#chart") .ejChart({ scrollChange: () { }});</p> <p>Property:zoomComplete\$("#chart") .ejChart({ zoomComplete: () { }});</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>API in Essential JS 2</p> <p>Property:selectedDataPo intIndexes\$("#chart") .ejChart({</p> <p>Property:selectedDataIndex eslet chart: Chart = new</p> <p>Not applicable</p>
--	--	---	---

		<pre>selectedDataPointIndexes: [{ seriesIndex: 0, pointIndex: 1}]]);</pre>	<pre>Chart({selectedDataIndexes: [{ series: 0, point: 1}]]});chart.appendTo('#chart');</pre>
	sideBySideSeriesPlacement for column based series	Property:sideBySideSeriesPlacement <pre>\$("#chart").ejChart({ sideBySideSeriesPlacement});</pre>	Property:sideBySidePlacement <pre>let chart: Chart = new Chart({ sideBySidePlacement: true});chart.appendTo('#chart');</pre>
	ZoomSettings	Property:zooming <pre>\$("#chart").ejChart({ zooming: { enable: true, enableDeferredZooming: true, enablePinch: true, enableMouseWheel: true, enableScrollBar: true, toolbarItems: [], type: 'X' }});</pre>	Property:zoomSettings <pre>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	Property:background <pre>\$("#container").ej Chart({ background: 'transparent'});</pre>	Property:background <pre>let chart: Chart = new Chart({ background: '#EEFFCC'});chart.app endTo('#chart');</pre>
	URL of the image to be used as chart background.	Property:backGroundImageUrl <pre>\$("#container").ej Chart({ backGroundImageUrl : '../images/chart/w heat.png'});</pre>	Not Applicable
	Customizing border of the chart	Property:border <pre>\$("#container").ej Chart({ border: { width: 2, color: '#CCEEFF', opacity: 0.5}});</pre>	Property:border <pre>let chart: Chart = new Chart({ border: { width: 2, color: '#CCEEFF'}});chart.ap pendTo('#chart');</pre>
	This provides	Property:exportSettings <pre>\$("#container").ej Chart({</pre>	Property:export() <pre>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>

<pre><tr> <td>Axis line cross value</td> <td> Property<i>crossesAt</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ crossesAt: 0 }] }); </code> </td> <td> Property<i>crossesAt</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ crossesAt: 4}] }); chart.appendTo('#chart'); </code> </td> </tr> <tr> <td>axis name with which the axis line has to be crossed</td> <td> Property<i>crossesInAxis</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ crossesInAxis: " " }]</pre>	
--	--

```
});
</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>
</tr>
```


<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'); |

```
</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><b>Crosshair label of an axis</b></td>
<td>
<b>Property</b>:<i>crossHairLabel.visible</i>
</br>
</br>
<code>
$("#container").ejChart({
axes: [ { crossHairLabel: { visible: true } } ]
});
</code>
</td>
<td>
<b>Property</b>:<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>
```

```
<b>Property</b>:<i>isInversed</i>
```

```
</br>
```

```
</br>
```

```
<code>
```

```
let chart: Chart = new Chart({
  axes: [ { 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({
  axes: [ { labelFormat: '{value}K' } ]
});
```

```
</code>
```

```
</td>
```

```
<td>
```

```
<b>Property</b>:<i>labelFormat</i>
```

```
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { 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({
axes: [ { labelIntersectAction: 'trim' } ]
});
</code>
</td>
<td>
<b>Property</b>:<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>

```
$("#container").ejChart({  
  axes: [ { labelRotation: 45 } ]  
});
```

</code>

</td>

<td>

Property:<i>labelRotation</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({  
  axes: [ { labelRotation: 45 } ]  
});  
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Log base value for logarithmic axis</td>

<td>

Property:<i>logBase</i>

</br>

</br>

<code>

```
$("#container").ejChart({  
  axes: [ { logBase: 10 } ]  
});
```


`</code>``</td>``<td>``Property:<i>labelRotation</i>``</br>``</br>``<code>``let chart: Chart = new Chart({``axes: [{ logBase: 10 }]``});``chart.appendTo('#chart');``</code>``</td>``</tr>``<tr>``<td>Major grid line</td>``<td>``Property:<i>majorGridLines.visible</i>``</br>``</br>``<code>``$("#container").ejChart({``axes: [{ majorGridLines: { visible: true } }]``});``</code>``</td>``<td>``Not Applicable``</td>``</tr>``<tr>``<td>Width of MajorGridLines</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' } } ]  
});
```

<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>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><b>Opacity of major grid line</b></td>
<td>
<b>Property</b>:<i>majorGridLines.opacity</i>
<br>
<br>
<code>
$( "#container").ejChart({
axes: [ { majorGridLines: { opacity: true} }}
]);
</code>
</td>
<td>
Not Applicable
</td>
</tr>
<tr>
<td><b>Major Tick line</b></td>
<td>
<b>Property</b>:<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>
</tr>
<tr>
<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>name of the primaryYAxis</td>

<td>

Property<i>name</i>

</br>

</br>

<code>

```
$("#container").ejChart({
  axes: [ { name: 'primaryYAxis' } ]
});
```

</code>

</td>

<td>

Property<i>name</i>

</br>

</br>

<code>

```
let chart: Chart = new Chart({
  axes: [ { name: 'primaryYAxis' } ]
});
chart.appendTo('#chart');
```

</code>

</td>

</tr>

<tr>

<td>Orientation of primaryYAxis</td>

<div><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 }] </div>
--

```

});
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>
<b>Property</b>:<i>maximum</i>
</br>
</br>
<code>
let chart: Chart = new Chart({
axes: [ { maximum: 23 } ]
});
chart.appendTo('#chart');
</code>
</td>
</tr>
<tr>
<td><b>interval for primaryYAxis</b></td>
<td>
<b>Property</b>:<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>
```

<pre><td> Property:<i>title</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ title: 'Chart title' }] }); chart.appendTo('#chart'); </code> </td> </tr> <tr> <td>StripLine of primaryYAxis</td> <td> Property:<i>stripLine</i> </br> </br> <code> \$("#container").ejChart({ axes: [{ stripLine: [] }] }); </code> </td> <td> Property:<i>stripLines</i> </br> </br> <code> let chart: Chart = new Chart({ axes: [{ stripLines: [] }] }); chart.appendTo('#chart');</pre>
--

```
</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>	Property:dashArray <code>let chart: Chart = new Chart({ series: [{ dataSource: [] }]});chart.appendTo('#chart');</code>
Draw type for Polar series	<code>Property:drawType\$("#chart").ejChart({ series: [{ drawType: 'Line' }]});</code>	Property:drawType <code>let 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>	Property:emptyPointSettings.displayMode <code>let chart: Chart = new Chart({ series: [{ displayMode: 'Average' }]});chart.appendTo('#chart');</code>
Empty Point color	<code>Property:emptyPointSettings.color\$("#chart").ejChart({ series: [{ color: 'red' }]});</code>	Property:emptyPointSettings.fill <code>let 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>	Property:fill <code>let 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>	Property:animation.enable <code>let 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>	Property:animation.duration <code>let 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:opacity let chart: Chart = new Chart({ series: [{ marker: { opacity : 0.5 } }]});chart.appendTo('#chart');
Shape of marker	Property:shape \$("#chart").ejChart({ series: [{ marker: { shape : 'Circle' } }]});	Property:shape let chart: Chart = new Chart({ series: [{ marker: { shape : 'Triangle' } }]});chart.appendTo('#chart');
ImageUrl of marker	Property:imageUrl \$("#chart").ejChart({ series: [{ marker: { imageUrl : '' } }]});	Property:imageUrl let 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:shape let 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:height let 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:width let 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:width let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});chart.appendTo('#chart');
DataLabelSettings of marker	Property:marker.dataLabel \$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]});	Property:marker.dataLabel let 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: <code>dataLabel.maximumLabelWidth\$("#chart").ejChart({ series: [{ marker: { dataLabel: { maximumLabelWidth: 20 } } }]});</code>	Not Applicable
Enable wrapping of text for data label	Property: <code>dataLabel.enableWrap\$("#chart").ejChart({ series: [{ marker: { dataLabel: { enableWrap: true } } }]});</code>	Not Applicable
To show contrast color for data label	Property: <code>dataLabel.showContrastColor\$("#chart").ejChart({ series: [{ marker: { dataLabel: { showContrastColor: true } } }]});</code>	Not Applicable
To show edge label for data label	Property: <code>dataLabel.showEdgeLabels\$("#chart").ejChart({ series: [{ marker: { dataLabel: { showEdgeLabels: true } } }]});</code>	Not Applicable
## TrendLines		
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	Property: <code>trendLines.polynomialOrder\$("#chart").ejChart({ series: [{ trendLines: [polynomialOrder: 3] }]});</code>	Property: <code>trendLines.polynomialOrder</code> let 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: 'EEEEFFCC']}]]);	Property:trendLines.filllet chart: Chart = new Chart({ series: [{ trendLines: [fill: 'EEEEFFCC']}]]);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: [{

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.alignment1let 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.overFlow1let 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.border1let 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.start1let 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.end1let 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.text1let 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:scrollEnd let chart: Chart = new Chart({ scrollEnd: () => { } }); chart.appendTo('#chart');
Event triggered when scroll changes.	Property:scrollChange\$("#chart").ejChart({ scrollChange: () { } });	Property:scrollChange let chart: Chart = new Chart({ scrollChange: () => { } }); chart.appendTo('#chart');
Fires while performing rectangle zooming in chart.	Property:zoomComplete\$("#chart").ejChart({ zoomComplete: () { } });	Property:zoomComplete let 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:selectedDataIndexes let chart: Chart = new Chart({ selectedDataIndexes: [{ series: 0, point: 1 }] }); chart.appendTo('#chart');
sideBySideSeries Placement for column based series	Property:sideBySideSeriesPlacement\$("#chart").ejChart({ sideBySideSeriesPlacement });	Property:sideBySidePlacement let 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:zoomSettings let 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:background let 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:lineCap\$("#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('#cha rt');
Positive fill for waterfall series	Property:positiveFill\$("#chart").ejChart({ series: [{ positiveFill: "red" }]});	Property:pointColorMappinglet chart: Chart = new Chart({ series: [{ pointColorMapping: 'color' }]});chart.appendTo('#cha rt');
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('#cha rt');
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('#cha rt');
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('#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: {

		<pre>imageUrl : '' } });});chart.appendTo('#c hart);</pre>
Border of marker	Property: <code>border</code> <pre>\$("#chart").ejChart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});});</pre>	Property: <code>shapelet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { border : { width: 2, color: 'red' } } }]});});chart.appendTo('#c hart);</pre>
Height of marker	Property: <code>size.height</code> <pre>\$("#chart").ejChart({ series: [{ marker: { size: { height: 30 } } }]});});</pre>	Property: <code>heightlet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { height: 25 } }]});});chart.appendTo('#c hart);</pre>
Width of marker	Property: <code>size.width</code> <pre>\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});});</pre>	Property: <code>widthlet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});});chart.appendTo('#c hart);</pre>
Width of marker	Property: <code>size.width</code> <pre>\$("#chart").ejChart({ series: [{ marker: { size: { width: 30 } } }]});});</pre>	Property: <code>widthlet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { width: 25 } }]});});chart.appendTo('#c hart);</pre>
DataLabelSettings of marker	Property: <code>marker.dataLabel</code> <pre>\$("#chart").ejChart({ series: [{ marker: { dataLabel: { } } }]});});</pre>	Property: <code>marker.dataLabellet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { } } }]});});chart.appendTo('#c hart);</pre>
Visibility of dataLabel	Property: <code>dataLabel.visible</code> <pre>\$("#chart").ejChart({ series: [{ marker: { dataLabel: { visible: true } } }]});});</pre>	Property: <code>dataLabel.visiblelet</code> <pre>let chart: Chart = new Chart({ series: [{ marker: { dataLabel: { visible: true } } }]});});chart.appendTo('#c hart);</pre>
Text mapping name of dataLabel	Property: <code>dataLabel.textMappingName</code> <pre>\$("#chart").ejChart({ series: [{ marker: { dataLabel: { textMappingName: '' } } }]});});</pre>	Property: <code>dataLabel.namelet</code> <pre>let 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	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

<!-- markdownlint-disable MD033 -->

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.multilevelLabelslet 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.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: [{

		maximumTextWidth: 20 }}] } }}}});chart.appendTo('#chart');
level of labels	Property:multiLevelLabels.level\$("#chart").ej Chart({ primaryXAxis: { multilevelLabels: [{ level: 2 } }}}});	Not applicable. Categories are used

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: 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

```
<!-- 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	<code>Property:axisLabelClick\$("#chart").ejChart({axisLabelClick: () { }});</code>	Not applicable
Fires before axis label render	<code>Property:axisLabelRendering\$("#chart").ejChart({axisLabelRendering: () { }});</code>	<code>Property:axisLabelRender()let chart: Chart = new Chart({ axisLabelRender: () => { }});chart.appendTo('#chart');</code>
Fires on axis label mouseMove	<code>Property:axisLabelMouseMove\$("#chart").ejChart({axisLabelMouseMove: () { }});</code>	Not applicable
Fires on axis label initialize	<code>Property:axisLabelInitialize\$("#chart").ejChart({axisLabelInitialize: () { }});</code>	Not applicable
Fires before axis range calculation	<code>Property:axesRangeCalculate\$("#chart").ejChart({axesRangeCalculate: () { }});</code>	<code>Property:axisRangeCalculated()let chart: Chart = new Chart({axisRangeCalculated: () => { }});chart.appendTo('#chart');</code>
Fires on axis title rendering	<code>Property:axisTitleRendering\$("#chart").ejChart({axisTitleRendering: () { }});</code>	Not applicable
Fires on after chart resize	<code>Property:afterResize\$("#chart").ejChart({afterResize: () { }});</code>	Not applicable
Fires on before chart resize	<code>Property:beforeResize\$("#chart").ejChart({beforeResize: () { }});</code>	<code>Property:resizedlet chart: Chart = new Chart({resized: () => { }});chart.appendTo('#chart');</code>
Fires on chart click	<code>Property:chartClick\$("#chart").ejChart({chartClick: () { }});</code>	<code>Property:chartMouseClickedlet chart: Chart = new Chart({ chartMouseClicked: () => { }});chart.appendTo('#chart');</code>
Fires on chart	<code>Property:chartMouseMove\$("#chart").ejChart({chartMouseMove: () { }});</code>	<code>Property:chartMouseMovelet 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.		<code>}}});chart.appendTo('#chart');</code>
Fires before loading the chart.	<code>Property:load\$("#chart").ejChart({ load: () { }});</code>	Property:load <code>let chart: Chart = new Chart({ load: () => { }});chart.appendTo('#chart');</code>
Fires, when multi level labels are rendering.	<code>Property:multiLevelLabelRendering\$("#chart").ejChart({ multiLevelLabelRendering: () { }});</code>	Property:axisMultiLabelRender <code>let chart: Chart = new Chart({ axisMultiLabelRender : () => { }});chart.appendTo('#chart');</code>
Fires on clicking a point in chart.	<code>Property:pointRegionClick\$("#chart").ejChart({ pointRegionClick: () { }});</code>	Property:pointClick <code>let chart: Chart = new Chart({ pointClick : () => { }});chart.appendTo('#chart');</code>
Fires when mouse is moved over a point.	<code>Property:pointRegionMouseMove\$("#chart").ejChart({ pointRegionMouseMove: () { }});</code>	Property:pointMove <code>let chart: Chart = new Chart({ pointMove : () => { }});chart.appendTo('#chart');</code>
Fires before rendering chart.	<code>Property:preRender\$("#chart").ejChart({ preRender: () { }});</code>	Not applicable
Fires when point render.	Not Applicable	Property:pointRender <code>let chart: Chart = new Chart({ pointRender : () => { }});chart.appendTo('#chart');</code>
Fires after selected the data in chart.	<code>Property:rangeSelected\$("#chart").ejChart({ rangeSelected: () { }});</code>	Not applicable
Fires after selecting a series.	<code>Property:seriesRegionClick\$("#chart").ejChart({ seriesRegionClick: () { }});</code>	Not applicable
Fires before rendering a series.	<code>Property:seriesRendering\$("#chart").ejChart({ seriesRendering: () { }});</code>	Property:seriesRender <code>let chart: Chart = new Chart({ seriesRender : () => {</code>

		<code>});chart.appendTo('#chart');</code>
Fires before rendering the marker symbols.	Property: <code>symbolRendering\$("#chart").ejChart({symbolRendering: () { }});</code>	Not applicable
Fires before rendering the trendline	Property: <code>trendlineRendering\$("#chart").ejChart({trendlineRendering: () { }});</code>	Not applicable
Fires before rendering the Chart title.	Property: <code>titleRendering\$("#chart").ejChart({titleRendering: () { }});</code>	Not applicable
Fires before rendering the Chart sub title.	Property: <code>subTitleRendering\$("#chart").ejChart({subTitleRendering: () { }});</code>	Not applicable
Fires before rendering the tooltip.	Property: <code>toolTipInitialize\$("#chart").ejChart({toolTipInitialize: () { }});</code>	Property: <code>tooltipRender</code> <code>let chart: Chart = new Chart({ tooltipRender : () => { }});chart.appendTo('#chart');</code>
Fires before rendering crosshair tooltip in axis	Property: <code>trackAxisToolTip\$("#chart").ejChart({trackAxisToolTip: () { }});</code>	Not applicable
Fires before rendering trackball tooltip.	Property: <code>trackToolTip\$("#chart").ejChart({trackToolTip: () { }});</code>	Not applicable
Event triggered when	Property: <code>scrollStart\$("#chart").ejChart({scrollStart: () { }});</code>	Property: <code>scrollStart</code> <code>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 EJ2 JavaScript 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.

```
`javascript
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.TS

```
import { Chart, LineSeries, getElement } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries);
let series1: Object[] = [];
let value: number = 10;
let i: number;
let intervalId: number;
for (i = 0; i < 50; i++) { // Data update
    if (Math.random() > .5) {
        value += Math.random() * 2.0;
    }
    series1[i] = { x: i, y: value };
}
let chart: Chart = new Chart({
    series: [
        {
            type: 'Line',
            dataSource: series1,
            xName: 'x',
            yName: 'y', animation: { enable: false }
        },
    ],
    width: '650px',
    height: '350px'
}, '#element');
let setTimeoutValue: number = 100;
intervalId = setInterval(
    (): void => {
        if (getElement('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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, DateTime, ITextRenderEventArgs, DataLabel } from
'@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, DataLabel);
let chart: Chart = new 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: (args: ITextRenderEventArgs) => {
        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 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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, DateTime, ITooltipRenderEventArgs, DataLabel,
Tooltip } from '@syncfusion/ej2-charts';
import { Internationalization } from '@syncfusion/ej2-base';
Chart.Inject(LineSeries, DateTime, DataLabel, Tooltip);
let chart: Chart = new 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:(args: ITooltipRenderEventArgs) => { // To format
the current point x value
    let intl: Internationalization = new Internationalization();
    let formattedString: string = 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>
            </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>

```

Add series in EJ2 JavaScript 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.TS

```

import { Chart, ColumnSeries, Category, Legend } from '@syncfusion/ej2-
charts';
Chart.Inject(ColumnSeries, Category, Legend);
import { Button } from '@syncfusion/ej2-buttons';
import { EmitType } from '@syncfusion/ej2-base';
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    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 EJ2 JavaScript 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="#7ddfle" stroke-
miterlimit="10" x1="0" y1="5.5" x2="10" y2="5.5"></line>
        <line fill="none" stroke="#7ddfle" 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 EJ2 JavaScript 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.TS

```

import { ChartAnnotation, Chart, StackingColumnSeries, Category, DataLabel,
Tooltip, ILoadedEventArgs, IAnnotationRenderEventArgs, Points } from
'@syncfusion/ej2-charts';
Chart.Inject(StackingColumnSeries, ChartAnnotation, DataLabel, Category,
Tooltip);
import { EmitType } from '@syncfusion/ej2-base';
let i: number = 0;
let chart: Chart = new Chart({
    //Initializing Primary X and Y Axis
    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: (args: IAnnotationRenderEventArgs) => {
        let length = chart.series.length - 1;
        let value = chart.series[length].stackedValues.endValues[i];
        i += (i == length) ? -length : 1;
        args.content.children[0].children[0].innerHTML = value;
    }
},
width: '650px',
height: '350px'
});
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="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>

```

Selected data grid in EJ2 JavaScript 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.TS

```

import { Chart, Selection } from '@syncfusion/ej2-charts';
import { Legend, Category, ScatterSeries } from '@syncfusion/ej2-charts';
Chart.Inject(Selection, Legend, Category, ScatterSeries);

```

```

import { Grid } from '@syncfusion/ej2-grids';
import { IDragCompleteEventArgs } from '@syncfusion/ej2/charts';
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    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: (args: IDragCompleteEventArgs) => {
        grid.dataSource = args.selectedDataValues[0];
        grid.refresh();
    }
});
chart.appendTo('#element');

```



```
let grid: Grid = new Grid({
  columns: [
    { field: 'x', headerText: 'x', type: 'string' },
    { field: 'y', headerText: 'y', type: 'number' }
  ],
});
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="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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, Category, IPointRenderEventArgs,
Legend, ILoadedEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category, Legend);
let shapes: string[] = [
    'Diamond', 'Circle', 'Rectangle', 'Line', 'Triangle', 'Rectangle'
];
let shapeRender: EmitType<IPointRenderEventArgs> = (args:
IPointRenderEventArgs)=> {
    args.shape = shapes[args.point.index];
};
let chart: Chart = new 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'
});
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="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 customization in EJ2 JavaScript 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.TS

```

import { Chart, StepAreaSeries, Legend, ILoadedEventArgs,
ILegendRenderEventArgs } from '@syncfusion/ej2-charts';

```

```

Chart.Inject(StepAreaSeries, Legend);
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    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: (args: ILegendRenderEventArgs)=> {
        if (args.text === 'Renewable') {
            args.shape = 'Circle';
        } else if (args.text === 'Non-Renewable') {
            args.shape = 'Triangle';
        }
    },
    width: '650px',
    height: '350px'
});
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="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>

```

Tool tip table in EJ2 JavaScript 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

<code>\$(x):</code>	<code>\$(y)</code>
---------------------	--------------------

</div>

,

Step 2:

To show that tooltip template, set the element id to the `template` property in tooltip.

INDEX.TS

```
import {
  Chart, LineSeries, Legend, Category, Tooltip
} from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category, Legend, Tooltip);
import { Browser } from '@syncfusion/ej2-base';
let chart: Chart = new 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'
});
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="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 EJ2 JavaScript 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.TS

```

import { Chart, SplineSeries, ChartAnnotation, Category, Legend } from
'@syncfusion/ej2-charts';
Chart.Inject(SplineSeries, Category, Legend, ChartAnnotation);

```

```

let chart: Chart = new 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'
});
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="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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, Category, Legend, Tooltip, ILoadedEventArgs,
StripLine, ChartTheme } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category, Legend, Tooltip, StripLine);
let chart: Chart = new 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'
        }
    ]
});
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="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 EJ2 JavaScript 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.TS

```

import { Grid, Selection, Page, ActionEventArgs } from '@syncfusion/ej2-
grids';
import { Query, DataManager } from '@syncfusion/ej2-data';
import { orderData } from '../datasource.ts';
import { Chart, LineSeries, DateTime } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime);

```

```

Grid.Inject(Selection, Page);
let chart: Chart;
let data: Object = new DataManager(orderData as JSON[]).executeLocal(new
Query().take(100));
let grid: Grid = new 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: () => {
        chart = new Chart({
            //Initializing Primary X Axis
            primaryXAxis: {
                valueType: 'DateTime',
                intervalType: 'Days'
            },
            series: [
                {
                    type: 'Line',
                    dataSource: grid.getCurrentViewRecords(),
                    xName: 'OrderDate', marker: {
                        visible: true
                    },
                    yName: 'Freight', name: 'Germany'
                }
            ]
        });
        chart.appendTo('#Chart');
    },
    actionComplete: (args: ActionEventArgs) => {
        if (args.requestType === 'paging') {
            chart.series[0].dataSource = grid.getCurrentViewRecords();
            chart.refresh();
        }
    }
});
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="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 EJ2 JavaScript 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.TS

```

import {
  AccumulationTheme, AccumulationChart, AccumulationLegend, PieSeries,
  AccumulationTooltip, IAccTooltipRenderEventArgs,
  AccumulationDataLabel
} from '@syncfusion/ej2-charts';
AccumulationChart.Inject(AccumulationLegend, PieSeries, AccumulationTooltip,
  AccumulationDataLabel);
let pie: AccumulationChart = new AccumulationChart({
  // Initialize the chart series
  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: (args: IAccTooltipRenderEventArgs) => {
    let value = args.point.y / args.series.sumOfPoints * 100;
    args.text = args.point.x + ' ' + Math.ceil(value) + ' %';
},
width: '650px',
height: '350px'
});
pie.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="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>

<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 EJ2 JavaScript 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.TS

```

import {
    Chart, LineSeries, Legend, Category, Tooltip, DataLabel
} from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, Category, Legend, Tooltip, DataLabel);
import { Browser } from '@syncfusion/ej2-base';
let datalabelData: Object[] = [
    { 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' },
  ];
let chart: Chart = new 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 EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, DateTime, ITooltipRenderEventArgs, DataLabel,
Series, Selection, Tooltip } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, DataLabel, Selection, Tooltip);
let chart: Chart = new 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: (args: ITooltipRenderEventArgs) => {
    let series: Series = <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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, Category, ChartAnnotation } from
 '@syncfusion/ej2-charts';
import { columnData } from './datasource.ts';
Chart.Inject(LineSeries, Category, ChartAnnotation);
let chart: Chart = new 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 EJ2 JavaScript 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.TS

```

import { Grid, Filter, Page, Selection, ActionEventArgs } from
'@syncfusion/ej2-grids';
import { orderData } from './datasource.ts';
import { AccumulationChart, ColumnSeries, DateTime, Category,
AccumulationDataLabel } from '@syncfusion/ej2-charts';
AccumulationChart.Inject(ColumnSeries, DateTime, Category,
AccumulationDataLabel);
Grid.Inject(Filter, Page, Selection);
let chart: AccumulationChart;
let filtertype: { [key: string]: Object }[] = [
    { id: 'Menu', type: 'Menu' },
    { id: 'CheckBox', type: 'CheckBox' },
    { id: 'Excel', type: 'Excel' }
];
let grid: Grid = new 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: () => {
            chart = new AccumulationChart({
                series: [
                    {
                        dataSource: grid.getCurrentViewRecords(),
                        type: 'Pie',
                        xName: 'CustomerName',
                        yName: 'Freight', dataLabel: { visible: true }
                    }
                ]
            });
            chart.appendTo('#Chart');
        },
        actionComplete: (args: ActionEventArgs) => {
            if (args.requestType === 'paging') {
                chart.series[0].dataSource = grid.getCurrentViewRecords();
                chart.refresh();
            }
        }
    });
    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="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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, Tooltip, getElement, Zoom,
Series, IZoomCompleteEventArgs } from '@syncfusion/ej2-charts';
import { downdata } from './datasource.ts';
Chart.Inject(LineSeries, Tooltip, Zoom);
/**
 * Sample for Downsampling
 */
let interval: number;
let chart: Chart = new 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: (args: IZoomCompleteEventArgs): void => {
            var zoomComplete:boolean = true;
            if(chart.primaryXAxis.zoomFactor<=0.4)
            {
                chart.series[0].dataSource =downdata;
            }
            else{
                var threshold = parseInt(chart.width) / 8,
                    xName = chart.series[0].xName,
                    yName = chart.series[0].yName,
                    sampledData = largestTriangleThreeBucket(downdata, threshold,
xName, yName);
                chart.series[0].dataSource = sampledData;
            }
        }
    });
chart.appendTo('#element');
function largestTriangleThreeBucket(data :any, threshold :number,
xProperty:any, yProperty:any) {
    yProperty = yProperty || 0;
    xProperty = xProperty || 1;
    var m = Math.floor,
        y = Math.abs,
        f =<number>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

```

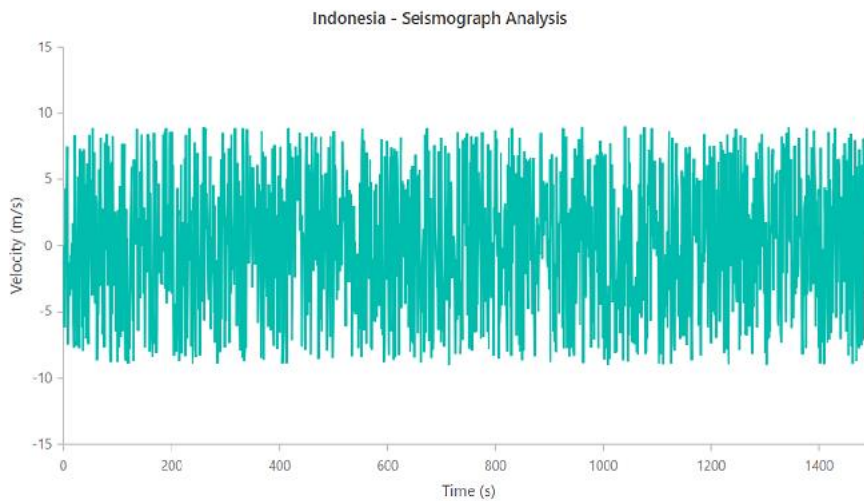
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" rel="stylesheet">

  <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>

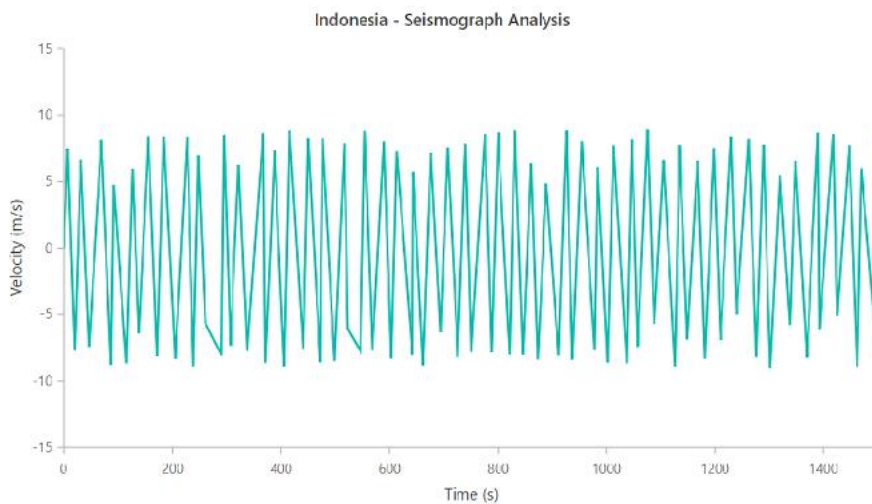
  <div id="container">
    <div id="element"></div>
    <div id="element1"></div>
  </div>
  <script>
  var ele = document.getElementById('container');
  if(ele) {
    ele.style.visibility = "visible";
  }
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```


Before applying downsampling algorithm



After applying downsampling algorithm



Clicked data in EJ2 JavaScript 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.TS

```
import {
    AccumulationTheme, AccumulationChart, AccumulationLegend, PieSeries,
    AccumulationTooltip, IAccTooltipRenderEventArgs, IPointEventArgs,
    AccumulationDataLabel
} from '@syncfusion/ej2-charts';
```

```

AccumulationChart.Inject(AccumulationLegend, PieSeries, AccumulationTooltip,
AccumulationDataLabel);
let pie: AccumulationChart = new 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: (args: IPointEventArgs) =>{
        document.getElementById("lbl1").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>

<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 EJ2 JavaScript 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.TS

```

import { Chart, LineSeries, DateTime, Legend, Tooltip, ILoadedEventArgs,
ChartTheme, Zoom, ScrollBar } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, Legend, Tooltip, Zoom, ScrollBar);
let chart: Chart = new 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: (args: ILoadedEventArgs) => {
    args.chart.zoomModule.isZoomed = 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="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 EJ2 JavaScript 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.TS

```

import { Chart, ColumnSeries, Category, Legend, DataLabel, Tooltip,
ILoadedEventArgs, IMouseEventArgs, MultiLevelLabel,
IAxisLabelRenderEventArgs, ChartAnnotation } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Category, Legend, Tooltip,
MultiLevelLabel, ChartAnnotation);
let columnData: Object[] = [

```

```

{ 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 }
];
let firstClick: Boolean = true;
let chart: Chart = new Chart({
  //Initializing Primary X and Y Axis
  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: IMouseEventArgs) => {
    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: IAxisLabelRenderEventArgs) => {
    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>

```



```

        </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 EJ2 JavaScript 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.TS

```

import { ChartTheme, Chart, LineSeries, DateTime, Legend, DataLabel,
IMouseEventArgs, Zoom } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, Legend, DataLabel, Zoom);
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'DateTime',
        labelFormat: 'y/M/d',
        edgeLabelPlacement: 'Shift',
    },
    primaryYAxis: {
        minimum: -20,
        maximum: 30,
        interval: 10,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}°C',
    },
    width: '450px',
    height: '300px',
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [
                { 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 }
    ],
    xName: 'x', width: 2,
    yName: 'y', name: 'Warmest',
    marker: {
        visible: true,
        height: 10, width: 10,
        shape: 'Pentagon',
        dataLabel: { visible: true, position: 'Top' }
    }
}, {
    type: 'Line',
    dataSource: [
        { x: new Date(2016, 3, 1), y: -5.3 },
        { x: new Date(2016, 4, 1), y: 1.0 }, { x: new Date(2016,
5, 1), y: 6.9 },
        { x: new Date(2016, 6, 1), y: 9.4 }, { x: new Date(2016,
7, 1), y: 7.6 },
        { x: new Date(2016, 8, 1), y: 2.6 }, { x: new Date(2016,
9, 1), y: -4.9 }
    ],
    xName: 'x', width: 2,
    yName: 'y', name: 'Coldest',
    marker: {
        visible: true, height: 10, width: 10, shape: 'Diamond',
        dataLabel: { visible: true, position: 'Bottom' }
    }
}
],
//Initializing Chart title
title: 'Alaska Weather Statistics - 2016',
zoomSettings:{
    enableSelectionZooming: true,
    enableMouseWheelZooming: true,
    enablePinchZooming: true
},
chartMouseMove: (args:IMouseEventArgs ) => {
    chart1.primaryXAxis.zoomFactor = chart.primaryXAxis.zoomFactor;
    chart1.primaryXAxis.zoomPosition =
chart.primaryXAxis.zoomPosition;
}
});
chart.appendTo('#element');
let chart1: Chart = new Chart({
    //Initializing Primary X and Y Axis
    primaryXAxis: {
        valueType: 'DateTime',
        labelFormat: 'MMM',
        edgeLabelPlacement: 'Shift',
    },
    primaryYAxis:
    {
        minimum: -20,
        maximum: 30,

```

```

        interval: 10,
        edgeLabelPlacement: 'Shift',
        labelFormat: '{value}°C',
    },
    width: '450px',
    height: '300px',
    //Initializing Chart Series
    series: [
        {
            type: 'Line',
            dataSource: [
                { 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 }
            ],
            xName: 'x', width: 2,
            yName: 'y', name: 'Warmest',
            marker: {
                visible: true,
                height: 10, width: 10,
                shape: 'Pentagon',
                dataLabel: { visible: true, position: 'Top' }
            }
        }, {
            type: 'Line',
            dataSource: [
                { x: new Date(2016, 3, 1), y: -5.3 },
                { x: new Date(2016, 4, 1), y: 1.0 }, { x: new Date(2016,
5, 1), y: 6.9 },
                { x: new Date(2016, 6, 1), y: 9.4 }, { x: new Date(2016,
7, 1), y: 7.6 },
                { x: new Date(2016, 8, 1), y: 2.6 }, { x: new Date(2016,
9, 1), y: -4.9 }
            ],
            xName: 'x', width: 2,
            yName: 'y', name: 'Coldest',
            marker: {
                visible: true, height: 10, width: 10, shape: 'Diamond',
                dataLabel: { visible: true, position: 'Bottom' }
            }
        }
    ],
    zoomSettings: {
        enableMouseWheelZooming: true,
        enablePinchZooming: true,
        enableSelectionZooming: true
    },
    //Initializing Chart title
    title: 'Alaska Weather Statistics - 2017',
    chartMouseMove: (args:IMouseEventArgs) => {
        chart1.primaryXAxis.zoomFactor = chart.primaryXAxis.zoomFactor;
        chart1.primaryXAxis.zoomPosition =
chart.primaryXAxis.zoomPosition;
    }
}

```

```

    }
  });
  chart1.appendTo('#element1');

```

INDEX.HTML

```

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

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
  type="text/javascript"></script>
  <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 EJ2 JavaScript 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.TS

```

import {
  Chart, ColumnSeries, IAxisLabelRenderEventArgs, DataLabel,
  ILoadedEventArgs, Tooltip, Legend, IMouseEventArgs
} from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Tooltip, Legend);
let chart: Chart = new Chart({

```

```

axes:[
  {
    rowIndex: 0,
    name: 'yAxis',
    opposedPosition: true,
    majorGridLines : {
      width : 0
    },
  },
  {
    rowIndex: 0,
    name: 'yAxis1',
    opposedPosition: true,
    majorGridLines : {
      width : 0
    },
  },
],
margin: {left: 100, right: 100},
//Initializing Chart Sample
series: [
  {
    type: 'Column',
    dataSource: [
      { x: 16, y: 2 }, { x: 17, y: 14 },
      { x: 18, y: 7 }, { x: 19, y: 7 },
      { x: 20, y: 10 }
    ],
    xName: 'x', width: 2,
    yName: 'y', name: 'England', fill: '#1e90ff',
  },
  {
    type: 'Column',
    dataSource: [
      { x: 16, y: 12 }, { x: 17, y: 10 },
      { x: 18, y: 17 }, { x: 19, y: 20 },
      { x: 20, y: 16 }
    ],
    xName: 'x', width: 2,
    yName: 'y', name: 'England', fill: 'green', yAxisName:
'yAxis1'
  },
  {
    type: 'Column',
    dataSource: [
      { x: 16, y: 7 }, { x: 17, y: 7 },
      { x: 18, y: 11 }, { x: 19, y: 8 },
      { x: 20, y: 24 }
    ],
    xName: 'x', width: 2,
    yName: 'y', name: 'West Indies', fill: '#b22222',
    yAxisName: 'yAxis',
  }
],
chartMouseClicked: (args: IMouseEventArgs) => {
  if (((args.target).indexOf('chart_legend_text') > -1) ||
((args.target).indexOf('chart_legend_shape') > -1)) ||

```

```

        (args.target).indexOf('chart_legend_shape_marker_') &&
        !(args.target).indexOf('chart_legend_element')) {
            var ids = ((args.target).indexOf('chart_legend_text') > -1) ?
            (args.target).split('chart_legend_text_')[1].split() :
            args.target.split('chart_legend_shape_marker_')[1] ||
            args.target.split('chart_legend_shape_')[1];
            var chart1 = document.getElementById("element").ej2_instances[0];
            var axesSeriesvisible1 = chart1.axes[0].series[0].visible;
            var axesSeriesvisible2 = chart1.axes[1].series[0].visible;
            if ((chart1.visibleSeries[ids].visible)) {
                if ((chart1.visibleSeries[ids].yAxisName === null)) {
                    chart1.primaryYAxis.visible = false;
                } else if ((chart1.visibleSeries[ids].yAxisName ===
            chart1.axes[0].name)) {
                    if ((!axesSeriesvisible1 && axesSeriesvisible2) ||
            (axesSeriesvisible1 && !axesSeriesvisible2) || (!axesSeriesvisible1 &&
            !axesSeriesvisible2)) {
                        chart1.axes[0].visible = false
                    } else if (((!axesSeriesvisible1 && axesSeriesvisible2) ||
            (axesSeriesvisible1 && !axesSeriesvisible2) || (axesSeriesvisible1 &&
            axesSeriesvisible1)) && !chart1.axes[1].visible) {
                        chart1.axes[1].visible = false;
                    }
                } else if ((chart1.visibleSeries[ids].yAxisName ===
            chart1.axes[1].name)) {
                    chart1.axes[1].visible = false;
                }
            } else {
                if ((chart1.visibleSeries[ids].yAxisName === null)) {
                    chart1.primaryYAxis.visible = true;
                } else if ((chart1.visibleSeries[ids].yAxisName ===
            chart1.axes[0].name)) {
                    if ((!axesSeriesvisible1 && axesSeriesvisible2) ||
            (axesSeriesvisible1 && !axesSeriesvisible2) || (!axesSeriesvisible1 &&
            !axesSeriesvisible2)) {
                        chart1.axes[0].visible = true
                    } else if (((!axesSeriesvisible1 && axesSeriesvisible2) ||
            (axesSeriesvisible1 && !axesSeriesvisible2) || (axesSeriesvisible1 &&
            axesSeriesvisible2)) && !chart1.axes[1].visible) {
                        chart1.axes[1].visible = true;
                    }
                } else if ((chart1.visibleSeries[ids].yAxisName ===
            chart1.axes[1].name)) {
                    chart1.axes[1].visible = true;
                }
            }
        }
    },
    //Initializing Chart title
    title: 'England vs West Indies',
    //Initializing User Interaction Tooltip
    tooltip: { enable: true, format: '${point.x}th Over : <b>${point.y}
Runs</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 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 EJ2 JavaScript Chart control

By using button click you can show exact time or month after scrolling. Here we have changed **zooFactor** 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.TS

```

import { Chart, LineSeries, DateTime, Legend, Tooltip, ILoadedEventArgs,
ChartTheme, Zoom, Crosshair, ScrollBar } from '@syncfusion/ej2-charts';
import { Browser } from '@syncfusion/ej2-base';
import { Internationalization } from '@syncfusion/ej2-base';
Chart.Inject(LineSeries, DateTime, Legend, Tooltip, Crosshair, Zoom,
ScrollBar);
let oneDayZoomFactor: Readonly<number> = 1 / 365;
export function numberOfDaysPassed(date: Date) {
  var current: number = +new Date(date.getTime());
  var previous: number = +new Date(date.getFullYear(), 0, 1);
  return Math.ceil((current - previous + 1) / (1000 * 60 * 60 * 24));
}
export function GetDateTimeData(start: Date, end: Date, min?: number, max?:
number, inc?: number): object[] {
  let series1: { x: Date, y: number }[] = [];
  let date: number;
  let value: number = 30;
  let option = {

```

```

    skeleton: 'full',
    type: 'dateTime'
  };
  let intl: Internationalization = new Internationalization();
  let dateParser: Function = intl.getDateParser(option);
  let dateFormatter: Function = intl.getDateFormat(option);
  for (let i: number = 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);
    }
    let point1: { x: Date, y: number } = { x: new Date(date), y:
Math.round(value) };
    if (currentMode === 'Hour') {
      new Date(start.setMinutes(start.getMinutes() + 1));
    } else {
      new Date(start.setDate(start.getDate() + 1));
    }
    series1.push(point1);
  }
  return series1;
}
export function getRandomInt(min: number, max: number): number {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
let data = GetDateTimeData(new Date('1/1/2014'), new Date('12/31/2014'));
let currentMode: string = 'Month';
let chart: Chart = new Chart({
  //Initializing Primary X Axis
  primaryXAxis: {
    title: 'Day',
    valueType: 'DateTime',
    edgeLabelPlacement: 'Shift',
    skeleton: 'short',
    skeletonType: 'Date',
    zoomFactor: oneDayZoomFactor * 31,
    zoomPosition: 0
  },
  //Initializing Primary Y Axis
  primaryYAxis:
  {
    title: 'Server Load',
    labelFormat: '{value}MB'
  },
  height: '350',
  width: '90%',
  crosshair: {
    enable: true, lineType: 'Vertical'
  },
  //Initializing Chart Series
  series: [
    {

```



```

        type: 'Line',
        dataSource: data,
        xName: 'x', width: 2, marker: {
            visible: true,
            width: 10,
            height: 10
        },
        yName: 'y', name: 'Germany',
    },
],
//Initializing Chart title
title: 'Network Load',
//Initializing User Interaction Tooltip
tooltip: {
    enable: true, shared: true,
    header: "${point.x}", format: "Server load : ${point.y}"
},
load: (args: ILoadedEventArgs) => {
    args.chart.zoomModule.isZoomed = true;
},
zoomSettings: {
    mode: 'X',
    toolbarItems: [],
    enableSelectionZooming: true,
    enableScrollbar: true
},
});
chart.appendTo('#element');
document.getElementById('lastdata').onclick = () => {
    let start: any = new Date(chart.primaryXAxis['visibleRange']['min']);
    if (currentMode === 'Month') {
        let days: number = new Date(start.getFullYear(), start.getMonth() + 1, 0).getDate();
        chart.primaryXAxis.zoomFactor = days * oneDayZoomFactor;
        chart.primaryXAxis.zoomPosition = oneDayZoomFactor *
        numberOfDaysPassed(new Date(start.getFullYear(), start.getMonth(), 0));
    } else {
        let lastHour: number = Math.floor(chart.primaryXAxis.zoomPosition /
        chart.primaryXAxis.zoomFactor);
        chart.primaryXAxis.zoomPosition = chart.primaryXAxis.zoomFactor *
        lastHour;
    }
    chart.dataBind();
};
document.getElementById('chartmode').onclick = () => {
    let modeBtn: HTMLButtonElement = document.getElementById("chartmode") as
    HTMLButtonElement;
    currentMode = modeBtn.innerHTML.indexOf('Switch To Hour') > -1 ? 'Hour' :
    'Month';
    modeBtn.innerHTML = modeBtn.innerHTML.indexOf('Switch To Hour') > -1 ?
    'Switch To Year' : 'Switch To Hour';
    document.getElementById('lastdata').innerHTML = 'Last ' + currentMode;
    if (currentMode === 'Hour') {
        let start: any = new Date(chart.primaryXAxis['visibleRange']['min']);
        let monthData: object[] = GetDateTimeData(new Date(start.getFullYear(),
        start.getMonth(), 1), new Date(start.getFullYear(), start.getMonth(), 2));
        chart.series[0].dataSource = monthData;
    }
};

```

```

    chart.primaryXAxis.zoomFactor = 60 / 1440;
    chart.primaryXAxis.zoomPosition = 0;
    chart.primaryXAxis.skeleton = 'hm';
    chart.primaryXAxis.skeletonType = 'Time';
    chart.dataBind();
  } else {
    chart.series[0].dataSource = GetDateTimeData(new Date('1/1/2014'), new
Date('12/31/2014'));
    chart.primaryXAxis.zoomFactor = oneDayZoomFactor * 31;
    chart.primaryXAxis.zoomPosition = 0;
    chart.primaryXAxis.skeleton = 'short';
    chart.primaryXAxis.skeletonType = 'Date';
    chart.dataBind();
  }
};

```

INDEX.HTML

```

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

  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.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>
    <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 EJ2 JavaScript 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.TS

```
import { Chart, ColumnSeries, DateTime, Legend, Tooltip, ILoadedEventArgs,
ChartTheme, IMouseEventArgs } from '@syncfusion/ej2-charts';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog } from '@syncfusion/ej2-popups';
Chart.Inject(ColumnSeries, DateTime, Legend, Tooltip);
let dialogObj: Dialog = new Dialog({
    header: 'Chart 2',
    target: document.getElementById('target'),
    showCloseIcon: true,
    width: '500px',
    height: '450px',
    allowDragging: true,
    visible: false,
    content: '<div id="container2"></div>'
});
dialogObj.appendTo('#defaultDialog');
let chart: Chart = new Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
        valueType: 'DateTime',
        labelFormat: 'y',
        intervalType: 'Years',
        edgeLabelPlacement: 'Shift',
    },
    //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 }
            ],
            xName: 'x', width: 2, marker: {
                visible: true,
                width: 10,
                height: 10
            },
            yName: 'y', name: 'Germany', animation: {enable: false},
        },
    ],
    //Initializing Chart title
    title: 'Inflation - Consumer Price',
});
chart.appendTo('#container');
let chart2: Chart = new Chart({
    //Initializing Primary X Axis
    primaryXAxis: {
```

```

        valueType: 'DateTime',
        labelFormat: 'Y',
        intervalType: 'Years',
        edgeLabelPlacement: 'Shift',
    },
    //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 }
            ],
            xName: 'x', width: 2, marker: {
                visible: true,
                width: 10,
                height: 10
            },
            animation: {enable: false},
            yName: 'Y', name: 'Germany', fill: 'blue'
        },
    ],
    width: '380',
    height: '300',
    //Initializing Chart title
    title: 'Inflation - Consumer Price',
});
chart2.appendTo('#container2');
document.getElementById('targetButton').onclick = (): void => {
    dialogObj.show();
};

```

INDEX.HTML

```

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

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.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 EJ2 JavaScript Chart control

By using the `chartMouseClick` 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.TS

```

import { ChartTheme, Chart, ColumnSeries, Category, Legend, DataLabel,
Tooltip, ILoadedEventArgs, IMouseEventArgs } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, DataLabel, Category, Legend, Tooltip);
import { Browser } from '@syncfusion/ej2-base';
var previousTarget = null;
let chart: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category', interval: 1,
    },
    series: [
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', name: 'Gold',
            dataSource: [{ x: 'USA', y: 46 }, { x: 'GBR', y: 27 }, { x: 'CHN', y:
26 }], animation: { enable: false },
            fill: 'red', opacity: 0.8
        },
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', name: 'Silver',
            dataSource: [{ x: 'USA', y: 37 }, { x: 'GBR', y: 23 }, { x: 'CHN', y:
18 }], animation: { enable: false },
            fill: 'green', opacity: 0.8
        },
        {
            type: 'Column', xName: 'x', width: 2, yName: 'y', name: 'Bronze',
            dataSource: [{ x: 'USA', y: 38 }, { x: 'GBR', y: 17 }, { x: 'CHN', y:
26 }], animation: { enable: false },
            fill: 'orange', opacity: 0.8
        }
    ],
    chartMouseClick: (args: IMouseEventArgs) => {
        var flag = false;
        if (((args.target).indexOf('chart_legend_text') > -1) ||
((args.target).indexOf('chart_legend_shape') > -1) ||
((args.target).indexOf('chart_legend_shape_marker_') &&
!(args.target).indexOf('chart_legend_element'))) {
            var ids = ((args.target).indexOf('chart legend text') > -1) ?

```

```

        (args.target).split('chart_legend_text_')[1] :
args.target.split('chart_legend_shape_marker_')[1] ||
args.target.split('chart_legend_shape_')[1];
    var chart1 = document.getElementById("element").ej2_instances[0];
    for (var i = 0; i < chart1.series.length; i++) {
        chart1.series[i].visible = false;
    }
    if (ids == previousTarget) {
        for (var j = 0; j < chart1.series.length; j++)
            chart1.series[j].visible = true;
        chart1.series[ids].visible = false;
        previousTarget = null;
        flag = true;
    }
    if (!flag)
        previousTarget = ids;
},
title: 'Olympic Medal Counts - RIO', 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 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>

```

Dynamic chart in EJ2 JavaScript 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.TS

```
import { Chart, LineSeries, DateTime, Legend, Tooltip, ILoadedEventArgs,
ChartTheme } from '@syncfusion/ej2-charts';
Chart.Inject(LineSeries, DateTime, Legend, Tooltip);
let count: number = 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 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 EJ2 JavaScript 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.TS

```
import { CheckBox } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Label position - Left.
let checkbox: CheckBox = new CheckBox({ label: 'Left Side Label',
labelPosition: 'Before' });
checkbox.appendTo('#checkbox1');
//Label position - Right.
checkbox = new 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.TS

```
import { CheckBox } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Small CheckBox.
let checkbox: CheckBox = new CheckBox({ label: 'Small', cssClass: 'e-small'
});
checkbox.appendTo('#checkbox1');
//Default CheckBox.
checkbox = new 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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

How To

Customized checkbox in EJ2 JavaScript 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.TS

```
import { CheckBox, Button } from '@syncfusion/ej2-buttons';
```

```
// To customize CheckBox appearance
// Refer the 'e-primary' class details in 'style.css'.
let checkbox: CheckBox = new CheckBox({ label: 'Primary', cssClass: 'e-
primary', checked: true });
checkbox.appendTo('#checkbox1');
// Refer the 'e-success' class details in 'style.css'.
checkbox = new CheckBox({ label: 'Success', cssClass: 'e-success', checked:
true });
checkbox.appendTo('#checkbox2');
// Refer the 'e-info' class details in 'style.css'.
checkbox = new CheckBox({ label: 'Info', cssClass: 'e-info', checked: true
});
checkbox.appendTo('#checkbox3');
// Refer the 'e-warning' class details in 'style.css'.
checkbox = new CheckBox({ label: 'Warning', cssClass: 'e-warning', checked:
true });
checkbox.appendTo('#checkbox4');
// Refer the 'e-danger' class details in 'style.css'.
checkbox = new CheckBox({ label: 'Danger', cssClass: 'e-danger', checked:
true });
checkbox.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.TS

```
import { CheckBox, Button } from '@syncfusion/ej2-buttons';
// To customize CheckBox frame appearance
let checkbox: CheckBox = new CheckBox({ label: 'Buy Groceries', cssClass:
'e-custom' ,checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new CheckBox({ label: 'Pay Rent', cssClass: 'e-custom' });
checkbox.appendTo('#checkbox2');
checkbox = new CheckBox({ label: 'Make Dinner', cssClass: 'e-custom' });
checkbox.appendTo('#checkbox3');
checkbox = new 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.TS

```
import { CheckBox, Button } from '@syncfusion/ej2-buttons';
// To customize CheckBox frame appearance
let checkbox: CheckBox = new CheckBox({ label: 'Buy Groceries', cssClass:
'e-checkicon', checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new CheckBox({ label: 'Pay Rent', cssClass: 'e-checkicon' });
checkbox.appendTo('#checkbox2');
checkbox = new CheckBox({ label: 'Make Dinner', cssClass: 'e-checkicon' });
checkbox.appendTo('#checkbox3');
checkbox = new 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>
<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,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXNlH+dzAAABoAAAAEJnbHlm1lv4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAYAAAAA
gbG9jYQON6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYW1l07lFxAABhAAAAIXcG9zdK9uovo
AAAhEAAAAGaABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANg+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIAJ4AAwAAAAA
AAgAAAAaOCgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAACAaaaaawAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHa/QAAwAHAAAlIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxcCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFawIBAQIDBQUHCAkJCws
MDA0ODf0mDQ4NDawLCwkJCACFBQMCA239Jg4NDQ0LCwsJCQgHBQUDAgEBAGMFBQcICQkLCwsNDQ0
OAtOODQ0NCwsLCQkIBwUFawIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEvBT8LLwG3NT8INS8FNT8NNS8JBByU/BDUvCyMPAQYtrQH5AgoEAQEBArg
hERESEyIJCsgQBiEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAWMCAGIEBAoBAQEECgCHBgUFBAMDAQEBAQQFBwkFBQUGef6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEEDBgoKDAYICAgFCAkICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBAQCBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNzM/CjUTHwcVIwcVIy8HETcXmz8KNScxBxEfDjSBHQEfDTMhMz8OES8PiZ0BLw4hA0EDBQQ
DAQIEbf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EAWQCAgIBAYYKChEQDQsJCAcEBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBwYGBgUFBAQDAgIBAQEBAgI
DBAQFBQYGBgCHBwgmAQMDAwUFBgYHBwgICQkJ/tQCiwMEbf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECAwMDBAGyAQECBAYHCAGJCgkQCaQC6/47CQkICQcIBwYGBQQEAWI
CUAGHBwCGBgYFBQQEAWMBAGIBAwMEBAUFBQcGBwCHCAImCAcHBwYGBgUFBAQDAgIBAdUJCQgICAg
GBwYFBQDAgEBAAAAAIAAAAAA6cD9AADAawAADchNSELAQcJAScBESNZA078sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAEEEEAAAAAABAAAGAAQABAAAAA
CAACACQABAAAAAADAAGAEAAABAAAAAEEAGAGAABAAAAAFAAsAIAABAAAAAAGAAGAKwABAAA

```

```

AAAAKACwAMwABAAAAAALABIAXwADAAEEECQAAAAIacQADAAEEECQABABAAcwADAAEEECQACAA4AgwA
DAAEEECQADABAakQADAAEEECQAEABAAoQADAAEEECQAFABYAsQADAAEEECQAGABAAxwADAAEEECQAKAFg
AlwADAAEEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmJ0bilpY29uVmVyc2lvbiAxLjB
idG4taWNvbkbZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZG1vd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAIAYgB0AG4ALQBpAGM
ABwBuAGIAAdABuAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMABwB
uAEYABwBuAHQAIAbNAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQB uAGMAZgB1AHM
AaQBvAG4AIAbNAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQB uAGMAZgB1AHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
GAQcBCAEJA AptZWRpYS1wbGF5C21lZG1hLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrZS0
tLTAXBGNvcHkQLWRvd25sb2FkLTAYLXdmlQAA) format('true type');
    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 EJ2 JavaScript 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.TS

```
import { CheckBox, Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Name and Value attribute in form submit.
let checkbox: CheckBox = new CheckBox({ name: 'Sport', value: 'Cricket',
label: 'Cricket', checked: true });
checkbox.appendTo('#checkbox1');
checkbox = new CheckBox({ name: 'Sport', value: 'Hockey', label: 'Hockey',
checked: true });
checkbox.appendTo('#checkbox2');
checkbox = new CheckBox({ name: 'Sport', value: 'Tennis', label: 'Tennis',
disabled: true });
checkbox.appendTo('#checkbox3');
checkbox = new CheckBox({ name: 'Sport', value: 'Basketball', label:
'Basketball' });
checkbox.appendTo('#checkbox4');
let button: Button = new 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 EJ2 JavaScript 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.TS

```

import { CheckBox } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize CheckBox component.
let checkbox: CheckBox = new 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;
}
```

Ej1 api migration in EJ2 JavaScript Check box control

This article describes the API migration process of Checkbox component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Checkbox Label | **Property:** *text*

 \$("#checkbox").ejCheckBox({
text: "Checkbox"
}); | **Property:** *label*

 var checkbox= new ej.buttons.CheckBox({
label: "Checkbox"
});
checkbox.appendTo("#checkbox"); |

| Checked state | **Property:** *checked*

 \$("#checkbox").ejCheckBox({
checked: true
}); | **Property:** *checked*

 var checkbox= new ej.buttons.CheckBox({
checked: true
});
checkbox.appendTo("#checkbox"); |

| Indeterminate state | **Property:** *enableTriState and checkState*

 \$("#checkbox").ejCheckBox({
enableTriState: true,
checkState: "indeterminate"
}); | **Property:** *indeterminate*

 var checkbox= new ej.buttons.CheckBox({
indeterminate: true
});
checkbox.appendTo("#checkbox"); |

| Adding custom class | **Property:** *cssClass*

 \$("#checkbox").ejCheckBox({
cssClass: "custom-class"
}); | **Property:** *cssClass*

 var checkbox= new ej.buttons.CheckBox({
cssClass: "custom-class"
});
checkbox.appendTo("#checkbox"); |

| Disabled state | **Property:** *enabled*

 \$("#checkbox").ejCheckBox({
enabled: false
}); | **Property:** *disabled*

 var checkbox= new ej.buttons.CheckBox({
disabled: true
});
checkbox.appendTo("#checkbox"); |

| State persistence | **Property:** *enablePersistence*

 \$("#checkbox").ejCheckBox({
enablePersistence: true
}); | **Property:** *enablePersistence*

 var checkbox= new ej.buttons.CheckBox({
enablePersistence: true
});
checkbox.appendTo("#checkbox"); |

| RTL | **Property:** *enableRTL*

 \$("#checkbox").ejCheckBox({
enableRTL: true,
 text: "Checkbox"
}); | **Property:** *enableRtl*

 var checkbox= new ej.buttons.CheckBox({
enableRtl: true,
 label: "Checkbox"
});
checkbox.appendTo("#checkbox"); |

| HTML Attributes | **Property:** *htmlAttributes*

 \$("#checkbox").ejCheckBox({
htmlAttributes : { required:"required" },
 text: "Checkbox"
}); | Not applicable |

| Id property | **Property:** *id*

 \$("#checkbox").ejCheckBox({
id: "sync"
}); | Not applicable |

| Prefix value of Id | **Property:** *idPrefix*

 \$("#checkbox").ejCheckBox({
idPrefix: "ej"
}); | Not applicable |

| Name attribute | **Property:** *name*

 \$("#checkbox").ejCheckBox({
name: "sports"
}); | **Property:** *name*

 var checkbox= new ej.buttons.CheckBox({
name: "sports"
});
checkbox.appendTo("#checkbox"); |

| Value attribute | **Property:** *value*

 \$("#checkbox").ejCheckBox({
value: "football",
name: "sports"
}); | **Property:** *value*

 var checkbox= new ej.buttons.CheckBox({
value: "football",
name: "sports"
});
checkbox.appendTo("#checkbox"); |

| Show rounded corner | **Property:** *showRoundedCorner*

 \$("#checkbox").ejCheckBox({
showRoundedCorner: true
}); | Not applicable |

| Size | **Property:** *size*

 \$("#checkbox").ejCheckBox({
size: "small"
 }); | **Property:** *cssClass*

 var checkbox= new ej.buttons.CheckBox({
cssClass: "e-small"
});
checkbox.appendTo("#checkbox"); |

| Label position | Not applicable | **Property:** *labelPosition*

 var checkbox= new ej.buttons.CheckBox({
label: "Checkbox",
labelPosition: "Before"
});
checkbox.appendTo("#checkbox"); |

| Validation rules | **Property:** *validationRules*

 \$("#checkbox").ejCheckBox({
validationRules:{ required:true }
 }); | Not applicable |

| Validation message | **Property:** *validationMessage*

 \$("#checkbox").ejCheckBox({
validationRules:{ required:true },
validationMessage: { required: "Required CheckBox value" }
 }); | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Destroy | **Method:** *destroy*

 \$("#checkbox").ejCheckBox({
text: "Checkbox"
 });
 var checkbox = \$("#checkbox").data("ejCheckBox");
checkbox.destroy(); | **Method:** *destroy*

 var checkbox= new ej.buttons.CheckBox({
label: "Checkbox"
});
checkbox.appendTo("#checkbox");
checkbox.destroy(); |

| Disable the Checkbox | **Method:** *disable*

 \$("#checkbox").ejCheckBox({
text: "Checkbox"
 });
 var checkbox = \$("#checkbox").data("ejCheckBox");
checkbox.disable(); | Not applicable |

| Enable the Checkbox | **Method:** *enable*

 \$("#checkbox").ejCheckBox({
text: "Checkbox"
 });
 var checkbox = \$("#checkbox").data("ejCheckBox");
checkbox.enable(); | Not applicable |

| Check state of the Checkbox | **Method:** *isChecked*

 \$("#checkbox").ejCheckBox({
text: "Checkbox"
 });
 var checkbox = \$("#checkbox").data("ejCheckBox");
checkbox.isChecked(); | Not applicable |

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| BeforeChange Event | **Events:** *beforeChange*

 \$("#checkbox").ejCheckBox({
beforeChange: function (args) { / code block */ }
 }); | Not applicable |

| Change Event | **Events:** *change*

 \$("#checkbox").ejCheckBox({
change: function change(args) { / code block / }
 }); | **Events:** *change*

 var checkbox= new ej.buttons.CheckBox({
change: function change(args) { / code block / }
 });
checkbox.appendTo("#checkbox"); |

| Create Event | **Events:** *create*

 \$("#checkbox").ejCheckBox({
create: function(args) { / code block / }
 }); | **Events:** *created*

 var checkbox= new ej.buttons.CheckBox({
created: function created() { / code block / }
 });
checkbox.appendTo("#checkbox"); |

| Destroy Event | **Events:** *destroy*

 \$("#checkbox").ejCheckBox({
destroy: function (args) { / code block */ }
 }); | Not applicable |

Chips

Getting started in EJ2 JavaScript Chips control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

control Initialization

The Essential JS 2 JavaScript controls can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `myapp` for Essential JS 2 JavaScript controls.

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

Syntax:

Script: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\dist\global\{PACKAGE_NAME}.min.js

Styles: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASEVERSION}\Web (Essential JS 2)\JavaScript\{PACKAGENAME}\styles\material.css

Example:

Script: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-buttons\dist\global\ej2-buttons.min.js

Styles: C:\Program Files (x86)\Syncfusion\Essential Studio\JavaScript - EJ2\16.3.0.17\Web (Essential JS 2)\JavaScript\ej2-buttons\styles\material.css

The below located script and style file contains all Syncfusion JavaScript (ES5) UI control resources in a single file.

Scripts: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\dist\ej2.min.js

Styles: **(installed location)**\Syncfusion\Essential Studio\JavaScript - EJ2\{RELEASE_VERSION}\Web (Essential JS 2)\JavaScript\ej2\material.css

Step 3: Create a folder `myapp/resources` and copy/paste the global scripts and styles from the above installed location to `myapp/resources` location.

Step 4: Create a HTML page (index.html) in `myapp` location and add the Essentials JS 2 script and style references.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Button's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-buttons.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Step 5: Now, add the `<div>` element and initiate the **Essential JS 2 Chip** control in the `index.html` by using following code

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 material theme -->
<link href="resources/base/styles/material.css" rel="stylesheet" type="text/css"/>
<link href="resources/buttons/styles/material.css" rel="stylesheet" type="text/css"/>
<!-- Essential JS 2 Button's global and dependent script -->
<script src="resources/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/ej2-buttons.min.js" type="text/javascript"></script>
</head>
<body>
<!-- Add the div element -->
<div id="chip"></div>
`
```

```
<script>
// initialize and Render chip control
new ej.buttons.ChipList({ chips: ["Janet Leverling"]}, '#chip');
</script>
</body>
</html>
`
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Chip** control.

Using CDN link for script and style reference

Step 1: Create an app folder `myapp` for the Essential JS 2 JavaScript controls.

Step 2: The Essential JS 2 control's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

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

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

Example:

Script: <https://cdn.syncfusion.com/ej2/ej2-buttons/dist/global/ej2-buttons.min.js>

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

Step 3: Create a HTML page (`index.html`) in `myapp` location and add the CDN link references. Now, add the `Chip` element and initiate the **Essential JS 2 Chip** control in the `index.html` by using following code.

INDEX.HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>

    <title>Essential JS 2</title>
    <!-- Essential JS 2 material theme -->
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet" type="text/css"/>
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet" type="text/css"/>
    <!-- Essential JS 2 Base's global script (Dependency Script) -->
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/dist/global/ej2-base.min.js" type="text/javascript"></script>
    <!-- Essential JS 2 Button's global script -->
    <script src="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/dist/global/ej2-buttons.min.js" type="text/javascript"></script>
    <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
```

```
</head>
  <body>
    <!-- Add the <div> element -->
    <div id="chip">Janet Leverling</div>
    <script>
      // initialize and Render chip control
      new ej.buttons.ChipList({ chips: ["Janet Leverling"]}, '#chip');
    </script>
  </body>
</html>
```

Step 4: Now, run the `index.html` in web browser, it will render the **Essential JS 2 Chip** control.

Types in EJ2 JavaScript 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```

import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
```

```
new ChipList({chips: ['Send a text', 'Set a remainder', 'Read my emails ',  
  'Set alarm'],  
  click: (e: ClickEventArgs)=>{  
    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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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 EJ2 JavaScript 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.TS

```

import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
```



```
new 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.TS

```
import { ChipList } from '@syncfusion/ej2-buttons';
new 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 EJ2 JavaScript 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;
}
,

```

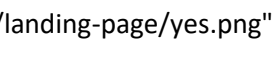
Accessibility in EJ2 JavaScript Chips component

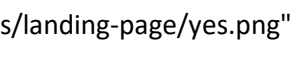
The Chips component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

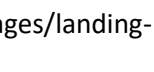
The accessibility compliance for the Chips component is outlined below.

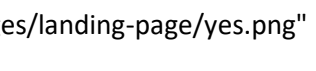
| Accessibility Criteria | Compatibility |

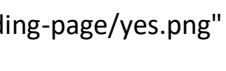
| -- | -- |

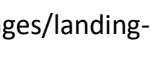
| [WCAG 2.2](#) Support |  |

| [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 |

| --- | --- |

| **role=checkbox** | Indicates the ChipList component wrapper element as **checkbox**. |

| **role=option** | Used to convey a significant and contextual message to the user(ChipList). |

| **role=button** | Used to convey a significant and contextual message to the user(Single Chip). |

| **aria-label** | Provides an accessible name for the Chip. |

| **aria-selected** | Indicates the element is selected. |

| **aria-disabled** | Indicates element is perceivable but disabled. |

| **aria-multiselectable** | 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 EJ2 JavaScript components](#)

Circular Gauge

Gauge dimensions in EJ2 JavaScript 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';  
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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 EJ2 JavaScript 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
```

```
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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>
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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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>
</body>
<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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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');

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>

```

```

<title>EJ2 Animation</title>
<meta charset="utf-8">
<meta name="viewport" content="width=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>

```

Labels

Labels of an axis can be customized by using [font](#) property in [labelStyle](#) options.

INDEX.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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>  
<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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';  
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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>
<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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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 EJ2 JavaScript 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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 take 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge, Gradient } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Gradient);
let rangeLinearGradient: Object = {
  startValue: '0%', endValue: '100%',
  colorStop: [
    { color: '#9E40DC', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '70%', opacity: 0.9 }
  ]
};
let gauge: CircularGauge = new 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/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/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/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/demos/src/circular-
gauge/images/athletics.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/demos/src/circular-
gauge/images/man1.png',
        radius: '78%', markerWidth: 28, markerHeight: 28,
        animation: { duration: 1500 }
    }, {
        type: 'Marker', value: 0.1, markerShape: 'Image',
        imageUrl: 'https://ej2.syncfusion.com/demos/src/circular-
gauge/images/man2.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.TS

```

import { CircularGauge, Gradient, Annotations } from '@syncfusion/ej2-
circulargauge';
CircularGauge.Inject(Gradient, Annotations);
let rangeRadialGradient: Object = {
    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 }
    ]
};
let gauge: CircularGauge = new 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/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/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/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/demos/src/circular-
      gauge/images/athletics.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/demos/src/circular-
      gauge/images/man1.png',
      radius: '78%', markerWidth: 28, markerHeight: 28,
      animation: { duration: 1500 }
    }, {
      type: 'Marker', value: 0.1, markerShape: 'Image',
      imageUrl: 'https://ej2.syncfusion.com/demos/src/circular-
      gauge/images/man2.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 EJ2 JavaScript Circular gauge control

Pointers are used to indicate values on the axis. Value of the pointer can be modified using the [value](#) property.

INDEX.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge, Annotations } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Annotations);
let gauge: CircularGauge = new CircularGauge({
  axes: [{
    startAngle: 270,
    endAngle: 90,
    lineStyle: { width: 3, color: '#1E7145' },
    labelStyle: {
      position: 'Outside',
      font: { size: '0px', color: '#1E7145' }
    },
    majorTicks: {
      width: 1,
      height: 0,
      interval: 100
    },
    minorTicks: {
      height: 0,
      width: 0,
    },
  },
  radius: '90%',
  minimum: 0,
  maximum: 100,
  pointers: [{
    animation: { enable: true, duration: 1000 },
    value: 70,
    radius: '80%',
    color: 'green',
    pointerWidth: 2,
    needleStartWidth: 4,
    needleEndWidth: 4,
    cap: {
      radius: 8,
      color: 'green'
    },
    needleTail: {
      length: '0%'
    }
  }
}
```

```

    }],
    annotations: [
      {
        angle: 180, zIndex: '1',
        radius: '20%',
        content: '<div style="color:#757575; font-family:Roboto; font-size:14px;padding-top: 26px">Customized Needle</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">
  <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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new CircularGauge({
  axes: [{
    pointers: [{
      value: 50,
      type: 'RangeBar',
      radius: '60%',
      color: '#007DD1',
      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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';

```

```
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new CircularGauge({
    enablePointerDrag: true,
    height: '250px',
    width: '250px',
    axes: [{
        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 Pointers

In addition to the default pointer, you can add n number of pointer to an axis by using `pointers` property.

INDEX.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge, Gradient } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Gradient);
let pointerLinearGradient: Object = {
  startValue: '0%',
  endValue: '100%',
  colorStop: [
    { color: '#FEF3F9', offset: '0%', opacity: 0.9 },
    { color: '#E63B86', offset: '70%', opacity: 0.9 }
  ]
};
let gauge: CircularGauge = new 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,
        linearGradient: pointerLinearGradient,
        cap: {
            radius: 8,
            color: 'white',
            border: {
                color: '#E63B86',
                width: 1
            }
        },
        needleTail: {
            length: '20%',
            linearGradient: pointerLinearGradient,
        }
    }, {
        radius: '60%', value: 40,
        animation: { duration: 1000 },
        pointerWidth: 10,
        linearGradient: pointerLinearGradient,
        cap: {
            radius: 8, color: 'white',
            border: { color: '#E63B86', 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>
</body>
</html>

```

```
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.TS

```
import { CircularGauge, Gradient } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Gradient);
let pointerRadialGradient: Object = {
  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 }
  ]
};
let gauge: CircularGauge = new 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: '#E63B86',
```

```

        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: '#E63B86', 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 EJ2 JavaScript 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.TS

```
import { CircularGauge, Annotations } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Annotations);
let gauge: CircularGauge = new 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>
</body>
</html>
```



```
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</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.TS

```
import { CircularGauge, Annotations } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Annotations);
let gauge: CircularGauge = new 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.TS

```

import { CircularGauge, Annotations } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Annotations);
let gauge: CircularGauge = new 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');
let gauge: CircularGauge = new 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 EJ2 JavaScript 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, pointers will be animated in the same way as [pointer animation](#).
5. If available, annotations will be animated.

The animation of the Circular Gauge is demonstrated in the following example.

INDEX.TS

```
import { CircularGauge, Annotations, Gradient } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Annotations, Gradient);
let circulargauge: CircularGauge = new 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>
</body>
</html>

```

```
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 EJ2 JavaScript 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.TS

```
import { CircularGauge, Legend } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Legend);
let gauge: CircularGauge = new 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,
    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.TS

```
import { CircularGauge, Legend } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Legend);
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge, Legend } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Legend);
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge, Legend, ILegendRenderEventArgs } from
 '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(Legend);
let gauge: CircularGauge = new 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
  },
  legendRender: (args: ILegendRenderEventArgs) => {
    args.text = "Legend Modified Text Value";
  }
}, '#element');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=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 EJ2 JavaScript 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.TS

```

import { CircularGauge, GaugeTooltip } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(GaugeTooltip);
let gauge: CircularGauge = new 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;
        ${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>

```

Template

Any HTML elements can be displayed in the tooltip by using the [template](#) property of the tooltip.

INDEX.TS

```

import { CircularGauge, GaugeTooltip } from '@syncfusion/ej2-circulargauge';

```



```
CircularGauge.Inject(GaugeTooltip);
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge, GaugeTooltip, Annotations } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(GaugeTooltip, Annotations);
let gauge: CircularGauge = new CircularGauge({
  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.TS

```

import { CircularGauge, GaugeTooltip } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(GaugeTooltip);
let gauge: CircularGauge = new 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;
        </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 print and export in EJ2 JavaScript Circular gauge control

Print

To use the print functionality, we should set the [allowPrint](#) property to **true**. The rendered circular gauge can be printed directly from the browser by calling the method [print](#).

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    allowPrint: true
}, '#element');
document.getElementById('print').onclick = () => {
    circulargauge.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-
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>
        <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

Image Export

To use the image export functionality, we should set the [allowImageExport](#) property to **true**. The rendered circular gauge can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The circular gauge can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
    allowImageExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    circulargauge.export('PNG', 'Gauge');
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=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>
        <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>

```

We can get the image file as base64 string for the JPEG and PNG formats. The circular gauge can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

INDEX.JS

```

var circulargauge = new ej.circulargauge.CircularGauge({
    allowImageExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    circulargauge.export('JPEG', 'Gauge', null, false).then((data) => {
        var base64 = data;
        document.writeln(base64);
    });
};

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=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>
        <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>
```

PDF Export

To use the PDF export functionality, we should set the [allowPdfExport](#) property to **true**. The rendered circular gauge can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and "0" indicates portrait and "1" indicates landscape.

INDEX.JS

```
var circulargauge = new ej.circulargauge.CircularGauge({
    allowPdfExport: true
}, '#element');
document.getElementById('export').onclick = () => {
    circulargauge.export("PDF", "Gauge");
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=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>
```



```
        <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>
```

Note: The exporting of the circular gauge as base64 string is not supported in the PDF export.

Gauge appearance in EJ2 JavaScript 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';  
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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%'
        }, {
            start: 70,
            end: 110,
            radius: '110%'
        }, {
            start: 110,
            end: 120,
            radius: '110%'
        }
    ]
}],
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=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.TS

```

import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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%'
    }, {
        start: 70,
        end: 110,
        radius: '110%'
    }, {
        start: 110,
        end: 120,
        radius: '110%'
    }]
    }]
}, '#element');
```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
let gauge: CircularGauge = new 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 EJ2 JavaScript 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 EJ2 JavaScript controls](#)

Internationalization in EJ2 JavaScript 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.TS

```
import { CircularGauge } from '@syncfusion/ej2-circulargauge';
import { loadCldr, L10n, setCulture, setCurrencyCode }
from '@syncfusion/ej2-base';
setCulture('de');
setCurrencyCode('EUR');
let gauge: CircularGauge = new CircularGauge({
    axes: [{
        labelStyle: {
            position: 'Inside',
            //Label format set 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.TS

```
import { CircularGauge, GaugeTooltip, Legend } from '@syncfusion/ej2-circulargauge';
CircularGauge.Inject(GaugeTooltip, Legend);
let gauge: CircularGauge = new 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 EJ2 JavaScript 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*
 let gauge: CircularGauge = new CircularGauge({ height: 400 });
 gauge.appendTo('#container');

| Width | **Property:** *width*
 let gauge: CircularGauge = new CircularGauge({ width: 100 });
 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*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ showScaleBar: true, size: 6 }] });
 gauge.appendTo('#container');

| Axisline Color | **Property:** *scales.size*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ showScaleBar: true, backgroundColor: "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*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ direction: "counterclockwise" }] });
 gauge.appendTo('#container');

| Axisline Radius | **Property:** *scales.radius*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ showScaleBar: true, radius: 150 }] });
 gauge.appendTo('#container');

| Axisline Startangle | **Property:** *scales.startAngle*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ startAngle: 80 }] });
 gauge.appendTo('#container');

| Axisline Endangle | **Property:** *scales.sweepAngle*
 let gauge: CircularGauge = new CircularGauge({ scales: [{ sweepAngle: 250 }] });
 gauge.appendTo('#container');

axes.endAngle
let gauge: CircularGauge = new CircularGauge({
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: [{
ticks: [{ 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: [{
ticks: [{ 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: [{
ticks: [{ 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: [{
ticks: [{ 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: [{
ticks: [{
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: [{
ticks: [{ 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: [{ ticks: [{ 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: [{ ticks: [{ 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: [{ ticks: [{ 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*
 scales: [{ showRanges: true , startValue: 20 }]
Property: *axes.ranges.start* let gauge: CircularGauge = new CircularGauge({
 axes: [{ start: 20 }]
 gauge.appendTo('#container');

|End Value| **Property:** *scales.ranges.endValue*
 scales: [{ showRanges: true , endValue: 30 }]
Property: *axes.ranges.end* let gauge: CircularGauge = new CircularGauge({
 axes: [{ end: 30 }]
 gauge.appendTo('#container');

|Start Width| **Property:** *scales.ranges.startWidth*
 scales: [{ showRanges: true , startWidth: 10 }]
Property: *axes.ranges.startWidth* let gauge: CircularGauge = new CircularGauge({
 axes: [{ startWidth: 10 }]
 gauge.appendTo('#container');

|End Width| **Property:** *scales.ranges.endWidth*
 scales: [{ showRanges: true , endWidth: 10 }]
Property: *axes.ranges.endWidth* let gauge: CircularGauge = new CircularGauge({
 axes: [{ endWidth: 10 }]
 gauge.appendTo('#container');

|Color| **Property:** *scales.ranges.backgroundColor*
 scales: [{ showRanges: true , backgroundColor: "red" }]
Property: *axes.ranges.color* let gauge: CircularGauge = new CircularGauge({
 axes: [{ color: "red" }]
 gauge.appendTo('#container');

|Offset| **Property:** *scales.ranges.distanceFromScale*
 scales: [{ showRanges: true , distanceFromScale: 10 }]
 Not Applicable

|Placement| **Property:** *scales.ranges.placement*
 scales: [{ showRanges: true , placement: "center" }]
 Not Applicable

|Opacity| **Property:** *scales.ranges.opacity*
 scales: [{ showRanges: true , opacity: 0.5 }]
 Not Applicable

|Radius| Not Applicable | **Property:** *axes.ranges.radius* let gauge: CircularGauge = new
 CircularGauge({
 axes: [{ radius: '80' }]
 gauge.appendTo('#container');

|Rounded Corner Radius| Not Applicable | **Property:** *axes.ranges.roundedCornerRadius* let
 gauge: CircularGauge = new CircularGauge({
 axes: [{ roundedCornerRadius: 10 }]
 gauge.appendTo('#container');

|Gradients| **Property:** *scales.ranges.gradients*
 scales: [{ 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:[{
 pointers: [{ 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*
scales: [{ type: 'marker' }]; | **Property:** *axes.pointers.type*
let gauge: CircularGauge = new CircularGauge({ axes: [{ type: 'marker', value: 20 }];
gauge.appendTo('#container');

| Marker Type | **Property:** *scales.pointers.markerType*
scales: [{ type: 'marker', markerType: 'rectangle' }]; | **Property:** *axes.pointers.markerShape*
let gauge: CircularGauge = new CircularGauge({ axes: [{ type: 'marker', markerShape: 'Diamond' }];
gauge.appendTo('#container');

| Marker Width | **Property:** *scales.pointers.width*
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*
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*
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*
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: [{
 customLabels: [{ value: 'Lineargauge' }]
 }];
Property: *axes.annotations.content*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{
 annotations: [{ content: 'Annotation' }]
 }];
 gauge.appendTo('#container');

|Angle| **Property:** *scales.customLabels.textAngle*
 \$("#container").ejCircularGauge({
 scales: [{
 customLabels: [{ textAngle: 90 }]
 }];
Property: *axes.annotations.angle*
 let gauge: CircularGauge = new CircularGauge({
 axes: [{
 annotations: [{ 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({
 gaugeMouseLeave: function(e: IMouseEventArgs): void { }
 gauge.appendTo('#container');|

| Mouseup Event | **Event:** *mouseClickUp*
 \$("#container").ejCircularGauge({
 mouseClickUp: function (args) {}
 | **Event:** *gaugeMouseUp*
 let gauge: CircularGauge = new CircularGauge({
 gaugeMouseUp: function(e: IMouseEventArgs): void { }
 gauge.appendTo('#container');|

| Pointerdrag Move Event | **Event:** *drawPointers*
 \$("#container").ejCircularGauge({
 drawPointers: function (args) {}
 | **Event:** *dragMove*
 let gauge: CircularGauge = new CircularGauge({
 dragMove: function(e: IMouseEventArgs): void { }
 gauge.appendTo('#container');|

| Draw Range Event | **Event:** *drawRange*
 \$("#container").ejCircularGauge({
 drawRange: function (args) {}
 | Not Applicable|

| Draw Ticks Event | **Event:** *drawTicks*
 \$("#container").ejCircularGauge({
 drawTicks: function (args) {}
 | Not Applicable|

| Legend Render Event | **Event:** *legendItemRender*
 \$("#container").ejCircularGauge({
 legendItemRender: function (args) {}
 | Not Applicable|

| Animation Complete Event | Not Applicable | **Event:** *animationComplete*
 let gauge: CircularGauge = new CircularGauge({
 animationComplete: function(e: IAnimationCompleteEventArgs): void { }
 gauge.appendTo('#container');|

| Right Click Event | **Event:** *rightClick*
 \$("#container").ejCircularGauge({
 rightClick: function (args) {}
 | Not Applicable|

| Double Click Event | **Event:** *doubleClick*
 \$("#container").ejCircularGauge({
 doubleClick: function (args) {}
 | Not Applicable|

How To

Gauge range in EJ2 JavaScript 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.TS

```
import { CircularGauge, Range } from '@syncfusion/ej2-circulargauge';
// initialize the circular gauge.
let circulargauge: CircularGauge = new CircularGauge({
  axes: [
    {
      minimum: 0, maximum: 120,
      // initialize the ranges.
      ranges: [
        {
          start: 0, end: 20
        }
      ]
    }
  ]
});
// render initialized circular gauge.
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>
  <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.TS

```
import { CircularGauge, Range } from '@syncfusion/ej2-circulargauge';
let circulargauge: CircularGauge = new CircularGauge({
  axes: [
    {
      minimum: 0, maximum: 120,
      ranges: [
        {
          start: 0, end: 20
        }
      ]
    }
  ]
});
circulargauge.appendTo('#element');
document.getElementById("addRange").onclick = function () {
  let start: number;
  let end: number;
  start =
+ (circulargauge.axes[0].ranges[circulargauge.axes[0].ranges.length -
1].start) + 20;
  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
Range(<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>
<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 EJ2 JavaScript 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.TS

```

import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new 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.TS

```
import { ColorPicker } from '@syncfusion/ej2-inputs';
```

```
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new 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', 'calibiri';
  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 EJ2 JavaScript 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.TS

```
import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple, L10n } from '@syncfusion/ej2-base';
enableRipple(true);
L10n.load({
  'de-DE': {
    'colorpicker': {
```

```

        "Apply": "Anwenden",
        "Cancel": "Abbrechen",
        "ModeSwitcher": "Modus wechseln"
    }
}
});
let colorPicker: ColorPicker = new 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.TS

```

import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple, L10n } from '@syncfusion/ej2-base';
enableRipple(true);
L10n.load({
  'ar-AE': {
    'colorpicker': {
      'Apply': 'تطبيق',
      'Cancel': 'إلغاء',
      'ModeSwitcher': 'مفتاح كهربائي الوضع'
    }
  }
});
let colorPicker: ColorPicker = new 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: #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;  
}
```

See Also

- [More information about localization](#)

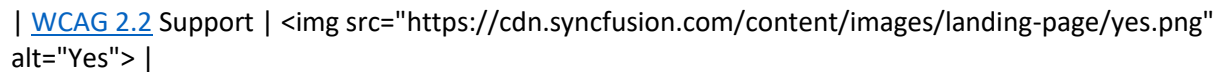
Accessibility in EJ2 JavaScript Color picker control

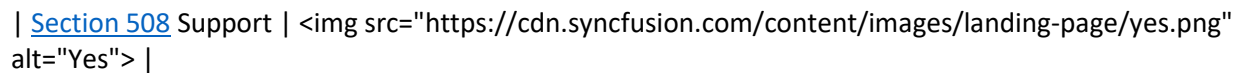
The Color picker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

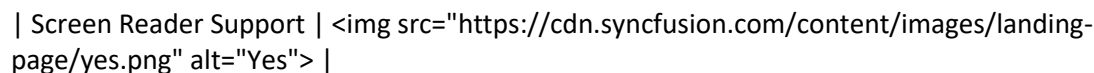
The accessibility compliance for the Color picker component is outlined below.

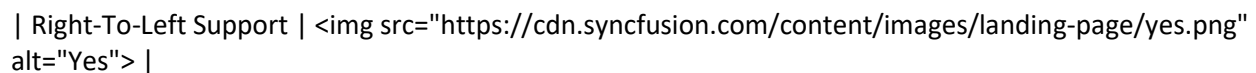
| Accessibility Criteria | Compatibility |

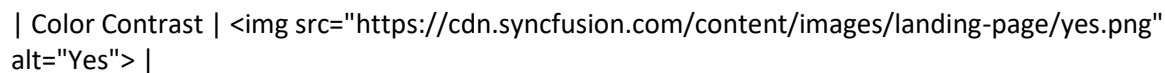
| -- | -- |

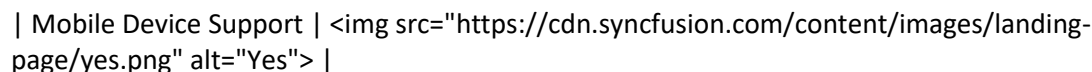
| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

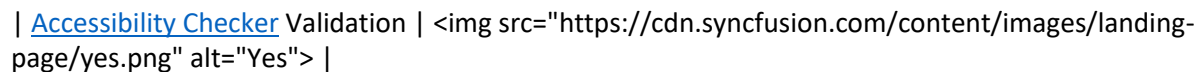
| Screen Reader Support |  |

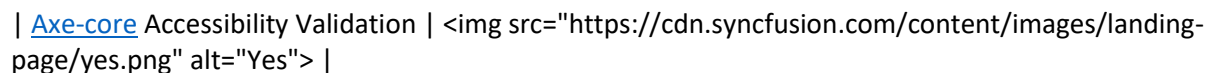
| Right-To-Left Support |  |

| 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 EJ2 JavaScript components](#)

Style and appearance in EJ2 JavaScript 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 palette 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 EJ2 JavaScript 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.TS

```
import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new ColorPicker({
    //To hide control buttons.
    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', 'calibri';
    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 EJ2 JavaScript 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.TS

```

import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);

```

```
let colorPicker: ColorPicker = new ColorPicker(
    {
        //To render Palette.
        mode: 'Palette',
        // To hide modeSwitcher.
        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: #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 render Picker alone specify the [mode](#) property as 'Picker'.

Colorpicker in dropdownbutton in EJ2 JavaScript 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.TS

```

import { ColorPicker, ColorPickerEventArgs } from '@syncfusion/ej2-inputs';
import { DropDownButton, BeforeOpenCloseMenuEventArgs,
OpenCloseMenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new ColorPicker(
    {
        inline: true,
        change: (args: ColorPickerEventArgs): void => {
            (ddb.element.children[0] as HTMLElement).style.backgroundColor =
args.currentValue.rgba;
            closePopup();
        }
    },
    '#element');
let ddb: DropDownButton = new DropDownButton(
    {
        target: ".e-colorpicker-wrapper",
        iconCss: "e-dropdownbtn-preview",
        beforeClose: (args: BeforeOpenCloseMenuEventArgs): void => {
            args.element.parentElement.querySelector('.e-
cancel').removeEventListener('click', closePopup);
        },
        open: (args: OpenCloseMenuEventArgs): void => {

```

```

        args.element.parentElement.querySelector('.e-
cancel').addEventListener('click', closePopup);
        tooltip();
    }
},
'#dropdownbtn');
function closePopup(): void {
    ddb.toggle();
}
function tooltip(): void {
    let zIndex = (document.getElementsByClassName('e-color-picker-
tooltip')[0] as HTMLElement).style.zIndex;
    let zIndexIntValue = parseInt(zIndex) + 2;
    (document.getElementsByClassName('e-color-picker-tooltip')[0] as
HTMLElement).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>
</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;
    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 EJ2 JavaScript 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.TS

```

import { ColorPicker, PaletteTileEventArgs, ColorPickerEventArgs } from
 '@syncfusion/ej2-inputs';
import { addClass, enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new ColorPicker({

```

```

mode: 'Palette',
inline: true,
modeSwitcher: false,
showButtons: false,
// 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',
    '#80deea', '#4dd0e1', '#26c6da', '#00bcd4'],
  'custom3': ['#80cbc4', '#4db6ac', '#26a69a', '#009688',
    '#a5d6a7', '#81c784',
    '#66bb6a', '#4caf50', '#c5e1a5', '#aed581', '#9ccc65',
    '#8bc34a', '#e6ee9c',
    '#dce775', '#d4e157', '#cddc39']
},
// Triggers before rendering each palette tile.
beforeTileRender: (args: PaletteTileEventArgs): void => {
  addClass([args.element], ['e-icons', 'e-custom-tile']);
},
// Triggers while selecting colors from palette.
change: (args: ColorPickerEventArgs): void => {
  document.getElementById('preview').style.backgroundColor =
args.currentValue.hex;
}
}, '#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.TS

```
import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new ColorPicker(
  {
    // To hide the input area
    cssClass: 'e-hide-value',
    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>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.TS

```
import { ColorPicker, OpenEventArgs } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new ColorPicker(
    {
        value: '#344aee',
        // To hide the input area
        cssClass: 'e-custom-picker',
        modeSwitcher: false,
        open: (args: OpenEventArgs): void => {
            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>
<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;
}
/* 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,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmL
TgiPz4KPCEtLSBHZW5lcmF0b3I6IEFkb2JlIElsbHVzdHJhdG9yIDIyLjEuMCwgU1ZHI
EV4cG9ydCBQbHVnLULuIC4gU1ZHIHFZlcnNpb246IDYuMDAgQnVpbGQgMCkgIC0tPgo8c3
ZnIHZlcnNpb249IjEuMSIgaWQ9IkkxheWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczLm9
yZy8yMDAwL3N2ZyIgeG1sbmM6eGxpbnM9Imh0dHA6Ly93d3d3LnczLm9yZy8yMDAwL3
N2ZyIjAgMCAxNiAxNiIgc3R5bGU9ImVuYVJsZS1iYWNRZ3JvdW5kOm5ldyAwIDAgaMTY
gMTY7IiB4bWw6c3BhY2U9InByZXNlcnZlIj4KPHN0eWxlIHR5cGU9InRleHQvY3NzIj4
KCS5zdDB7ZmlsbDojRkZGRkZG030KPC9zdHlsZT4KPGC+Cgk8cG9seWdvbiBjbGFzc
z0ic3QwIiBwb2ludHM9IjE2LDYgMTAsNiAxMCwwIDYsMCA2LDYgMCw2IDAsMTAgNiwx
MCA2LDE2IDEwLDE2IDEwLDEwIDE2LDEwIAkiLz4KPC9nPgo8cGF0aCBkPSJNMTAsNlY
SDZ2NkgwdjRoNnY2aDR2LTZoNlY2SDEweiBNMTUsOUg5djZINlY5SDFWN2g2VjFoMn
Y2aDZWOXoiLz4KPC9zdmc+Cg==');
    font-size: 16px;
    height: 16px;
    line-height: 16px;
    margin-left: -8px;
    margin-top: -8px;
    border: none;
    box-shadow: none;
    width: 16px;
}
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.TS

```
import { ColorPicker, ColorPickerEventArgs } from '@syncfusion/ej2-inputs';
import { addClass, enableRipple } from '@syncfusion/ej2-base';
let colorPicker: ColorPicker = new ColorPicker(
    {
        change: (args: ColorPickerEventArgs): void => {
            (colorPicker.element.nextElementSibling.querySelector('.e-selected-color') as HTMLElement).style.borderBottomColor =
            args.currentValue.rgba;
        },
        '#element');
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.TS

```
import { ColorPicker } from '@syncfusion/ej2-inputs';
import { addClass, enableRipple } from '@syncfusion/ej2-base';
let colorPicker: ColorPicker = new ColorPicker({}, '#element');
let target: Element = 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>
<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.TS

```
import { ColorPicker, ColorPickerEventArgs } from '@syncfusion/ej2-inputs';
import { Dialog } from '@syncfusion/ej2-popups';
import { addClass, enableRipple, EmitType } from '@syncfusion/ej2-base';
import { SplitButton, OpenCloseMenuEventArgs, BeforeOpenCloseMenuEventArgs }
from '@syncfusion/ej2-splitbuttons';
let colorPalette: ColorPicker = new ColorPicker(
  {
    mode: 'Palette',
    inline: true,
    showButtons: false,
    modeSwitcher: false,
    change: paletteOnChange
  }
, '#palette');
let splitBtn: SplitButton = new SplitButton(
  {
    target: '#target',
    iconCss: 'e-icons e-font-icon',
    open: (args: OpenCloseMenuEventArgs) => {
      args.element.children[1].addEventListener('click', openDialog);
    },
    beforeClose: (args: BeforeOpenCloseMenuEventArgs): void => {
      args.element.children[1].removeEventListener('click',
openDialog);
    }
  }
, '#split-btn');
let modalDialog: Dialog = new 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');
let colorPicker: ColorPicker = new ColorPicker(
  {
    inline: true,
```

```

        modeSwitcher: false,
        change: pickerOnChange
    }
    , '#picker');
function openDialog(): void {
    modalDialog.show();
}
function dialogOpen(): void {
    colorPicker.refresh();
    colorPicker.element.nextElementSibling.querySelector('.e-ctrl-btn .e-
cancel').addEventListener('click', dlgClose);
}
function dlgClose(): void {
    modalDialog.hide();
}
function paletteOnChange(args: ColorPickerEventArgs): void {
    (splitBtn.element.querySelector('.e-font-icon') as
HTMLElement).style.borderBottomColor = args.currentValue.rgba;
}
function pickerOnChange(args: ColorPickerEventArgs): void {
    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-
pops/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','calibiri';
  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 EJ2 JavaScript 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.TS

```

import { ColorPicker, ColorPickerEventArgs } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
let preview: HTMLElement = document.getElementById('preview');
let colorPicker: ColorPicker = new ColorPicker(
  {
    mode: "Palette",
    value: "#ba68c8",
    showButtons: false,
    modeSwitcher: false,
    //To enable no color support
    noColor: true,
    change: (args: ColorPickerEventArgs): void => {
      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.TS

```

import { ColorPicker, ColorPickerEventArgs, PaletteTileEventArgs } from
 '@syncfusion/ej2-inputs';
import { SplitButton } from '@syncfusion/ej2-splitbuttons';
let preview: HTMLElement = document.getElementById('preview');
let colorPicker: ColorPicker = new 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: (args: PaletteTileEventArgs): void => {
      args.element.classList.add('e-custom-tile');
    },
    change: (args: ColorPickerEventArgs): void => {
      (document.querySelector(".e-split-btn .e-picker-icon")
as HTMLElement).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');

```

```

let splitBtn: SplitButton = new SplitButton({ iconCss: "e-cp-icons e-picker-
icon", target: "#target" }, '#splitbtn')
preview.style.backgroundColor = "#f44336";
preview.textContent = "#f44336";
document.getElementById('no-color').onclick = (): void => {
    //sets color picker value property to null
    colorPicker.setProperties({ 'value': "" }, true);
    (document.querySelector('.e-split-btn .e-picker-icon') as
HTMLElement).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>

```

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%;
}
@font-face {
    font-family: 'paint';
    src:
        url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRIAAAEoAAAVMnTYXDnEodVAAABiAAAADZnbHlmIZD
+uwAAACgAAADMaGVhZBKhHhQAAADQAAAAANmhoZWEHjANrAAAArAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBMAAAAAAHAAAAABmlheHABDgBKAAABCAAAACBuYwlln6hZswAAApQAAAIINcG9zdEkLMmU
AAASkAAAAANgABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAAA19Kf2f/z//wD6gPhAAAAACAACAAAAAAAAAAAAEAAAACAD4AAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAANoCvAAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAAABAAAAAAAAABAAAAAPo//M
AAAACAAAAAwAAABQAAwABAAAAAFAAEACTIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAZgAAAAAL/8//8A+oD4QAKAD0AAAEEWBgceATc1JiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDeiRlCgulCxP+8RT+GyYDQFxoZQwTBQEDDxEBJzo
nAREOCQkPJQ4cDBcdAf6oG1a3nx8BWQ4RHKADeG1oWwHTLHVwYVmL6Kx1BHEqfwYFqWUHEX4tDAo
cEx0nJx0RHgoVUDQpDgsBFAH+px2guFUaAvkNOiCgCXnhCAWAAAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAEABQABAAEAAAAAAAAIABwAGAAEAAAAAAAAAMABQANAAEAAAAAAAAQABQASAAEAAAAAAAU
ACwAXAAEAAAAAAAYABQaiAAEAAAAAAAOALAAnAAEAAAAAAASAEgBTAAMAAQQJAAAAAgBlaAMAAQQ
JAAEACgBnAAMAAQQJAAIADgBxAAMAAQQJAAMACgB/AAMAAQQJAAQACgCJAAMAAQQJAAUAFgCTAAM
AAQQJAAAYACgCpAAMAAQQJAAoAWACzAAMAAQQJAAsAJAELIHBhaW50UmVndWxhcncBhaW50cGFpbncR
WZXJzaW9uIDEuMHBAhaW50Rm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2l2b2lBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2l2b2l5jb20AIAABwAGEAaQBuAHQAUGbLAGcAdQBsAGEAcgBwAGEAaQBuAHQ
AcABhAgKAbgB0AFYAZQByAHMAaQBvAG4AIAAxAx4AMABwAGEAaQBuAHQAARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGKAbgBnACAAUwB5AG4AYwBmAUAwAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAUAwAcwBpAG8AbgAuAGMAbwBtAAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhaW50LWJlY2tldAAAAAA=)
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,PD94bWwgdmVyc2lvdj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPHN2YyB3aWR0aD0iNnB4IiBoZWlnaHQ9IjZweCIgdmlld0JveD0iMCAwidmVyc2lvdj0iMS4xIiB4bWxuc20iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciaHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hsaW5rIj4KICAgIDwhLS0gR2VuZXJhdG9yOXBta2V0YS2ggNTAgKDU0OTgzKSAtIGh0dHA6Ly93d3cuYm9oZW1pYW5jb2RpbmcuY29tL3NrZXRxjaCatLT4KI CAgidX0aXRszT5Hcm9lcA5PC90aXRszT4KICAgIDxkZXNjPkNyZWZF0ZWQgd2l0aCBTa2V0Y2guPC9kZXNjPgogICAgPGRlZnM+PC9kZWZzPgogICAgPGcgaWQ9IlBhZ2UtMSIgc3Ryb2t1PSJub251IiBzdHJva2Utd2lkdgG9IjEiIGZpbGw9Im5vbWUiIGZpbGwtcnVsZT0iZXZlbm9kZCI+CjAgICAqICAgPGcgaWQ9Ikdyb3VwLTkiPgoqICAgICAqICAqICAqICA8cmVjdCBpZD0iUmVjdGFuZ2xlLTEuXIIiBma
```

```

WxsPSIjRTBFMEUwIiB4PSIwIiB5PSIwIiB3aWR0aD0iMyIgaGVpZ2h0PSIzIj48L3JlY3Q+CiAgI
CAgICAgICAgIDxyZWNOIGlkPSJSZWN0YW5nbGUtMTetQ29weS0yIiBmaWxsPSIjRkZGRkZGIiB4P
SIwIiB5PSIzIiB3aWR0aD0iMyIgaGVpZ2h0PSIzIj48L3JlY3Q+CiAgICAgICAgICAgIDxyZWNOI
GkPSJSZWN0YW5nbGUtMTetQ29weSIgZmlsbD0iI0ZGRkZGRiIgeD0iMyIgeT0iMCIgd2lkdGg9I
jMiIGhlaWdodD0iMyI+PC9yZWNOPgogICAgICAgICAgICA8cmVjdCBpZD0iUmVjdGFuZ2xlLTExL
UNvcHktMyIgzmlsbD0iI0UwRTBFMCigeD0iMyIgeT0iMyIgd2lkdGg9IjMiIGhlaWdodD0iMyI+P
C9yZWNOPgogICAgICAgIDwvZz4KICAgIDwvZz4KPC9zdmc+');
}
/* 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 EJ2 JavaScript 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.TS

```

import { ColorPicker } from '@syncfusion/ej2-inputs';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let colorPicker: ColorPicker = new 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>
</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;
    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;
}
```

Ej1 api migration in EJ2 JavaScript Color picker control

This article describes the API migration process of ColorPicker component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Default | **property:** `value`
`
</br>$("#colorpicker").ejColorPicker({ value: "#278787" });` | **property:** `value`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ value: '#278787' });
colorPickerObj.appendTo('#colorpicker');` |

| Inline mode color picker | **property:** `displayInline`
`
</br>$("#colorpicker").ejColorPicker({ displayInline: true });` | **property:** `inline`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ inline: true });
colorPickerObj.appendTo('#colorpicker');` |

| Adding custom class | **property:** `cssClass`
`
</br>$("#colorpicker").ejColorPicker({ cssClass: "e-custom" });` | **property:** `cssClass`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ cssClass: 'e-custom' });
colorPickerObj.appendTo('#colorpicker');` |

| Disable the ColorPicker component | **property:** `enabled`
`
</br>$("#colorpicker").ejColorPicker({ enabled: false });` | **property:** `disabled`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ disabled: true });
colorPickerObj.appendTo('#colorpicker');` |

| To display custom text in button elements | **property:** `buttonText`
`
</br>$("#colorpicker").ejColorPicker({ buttonText: { apply: "Apply", cancel: "Cancel", swatches: "Swatches" } });` | Not Applicable |

| To display customized text or content when mouse over the color picker elements | **property:** `tooltipText`
`
</br>$("#colorpicker").ejColorPicker({ tooltipText: { switcher: "Switch", currentColor: "New Color", selectedColor: "Old Color" } });` | Not Applicable |

| Disable / hide opacity | **property:** `enableOpacity`
`
</br>$("#colorpicker").ejColorPicker({ enableOpacity: false });` | **property:** `enableOpacity`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ enableOpacity: false });
colorPickerObj.appendTo('#colorpicker');` |

| ColorPicker Button mode | **property:** `buttonMode`
`
</br>$("#colorpicker").ejColorPicker({ buttonMode: "Dropdown" });` | Not Applicable |

| To show / hide the control (apply / cancel) buttons | **property:** `showApplyCancel`
`
</br>$("#colorpicker").ejColorPicker({ showApplyCancel: false });` | **property:** `showButtons`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ showButtons: false });
colorPickerObj.appendTo('#colorpicker');` |

| To show / hide the clear button | **property:** `showClearButton`
`
</br>$("#colorpicker").ejColorPicker({ showClearButton: true });` | Not Applicable |

| Show / hide the mode (picker / palette) switcher | **property:** `showSwitcher`
`
</br>$("#colorpicker").ejColorPicker({ showSwitcher: false });` | **property:** `modeSwitcher`
`
</br>var colorPickerObj = new ej.inputs.ColorPicker({ modeSwitcher: false });
colorPickerObj.appendTo('#colorpicker');` |

| To show / hide the preview area | **property:** `showPreview`
`
</br>$("#colorpicker").ejColorPicker({ showPreview: false });` | Not Applicable |

| To show / hide the recent selected color list | **property:** `showRecentColors`
`
</br>$("#colorpicker").ejColorPicker({ showRecentColors: true });` | Not Applicable |

| To show / hide the color picker slider tooltip | **property:**

showTooltip

\$("#colorpicker").ejColorPicker({
 showTooltip: false
}); | Not Applicable |

| Custom icon in dropdown control color area | **property:**

toolIcon

\$("#colorpicker").ejColorPicker({
 toolIcon: null
}); | Not Applicable |

| ColorPicker mode | **property:** *modelType*

\$("#colorpicker").ejColorPicker({

modelType: "picker"
}); | **property:** *mode*

var colorPickerObj = new ej.inputs.ColorPicker({
 mode: 'Palette'
});
colorPickerObj.appendTo('#colorpicker'); |

| Opacity value | **property:** *opacityValue*

\$("#colorpicker").ejColorPicker({
 opacityValue: 80
}); | Not Applicable |

| Number of columns in color palette | **property:**

columns

\$("#colorpicker").ejColorPicker({
 columns: 10
}); | **property:** *columns*

var colorPickerObj = new ej.inputs.ColorPicker({
 columns: 15
});
colorPickerObj.appendTo('#colorpicker'); |

| Custom colors | **property:** *palette*

\$("#colorpicker").ejColorPicker({
 palette:

ej.ColorPicker.Palette.CustomPalette
 custom: ["ffffff", "ffccff", "ff99ff", "ff66ff",
 "ff33ff", "ff00ff", "ccffff", "ccccff",
 "cc99ff", "cc66ff", "cc33ff", "cc00ff",
 "99ffff", "99ccff", "9999ff", "9966ff",
 "9933ff", "9900ff", "ffffcc", "ffcccc"
]
}); | **property:** *presetColors*

var colorPickerObj = new ej.inputs.ColorPicker({
 presetColors: {
 'custom': ["ffffff", "ffccff", "ff99ff", "ff66ff",
 "ff33ff", "ff00ff", "ccffff", "ccccff",
 "cc99ff", "cc66ff", "cc33ff", "cc00ff",
 "99ffff", "99ccff", "9999ff", "9966ff",
 "9933ff", "9900ff", "ffffcc", "ffcccc"
}
});
colorPickerObj.appendTo('#colorpicker'); |

| Rendering palette from the predefined set of palettes | **property:**

presetType

\$("#colorpicker").ejColorPicker({
 presetType: ej.ColorPicker.Presets.FlatColors
}); | Not Applicable |

| No color option in color palette | Not Applicable | **property:** *noColor*

var colorPickerObj = new ej.inputs.ColorPicker({
 noColor: true,
 modeSwitcher: false
 mode: 'Palette'
});
colorPickerObj.appendTo('#colorpicker'); |

| Localization | **property:** *locale*

\$("#colorpicker").ejColorPicker({
 locale: 'zh-

CN'
});
ej.ColorPicker.Locale["zh-CN"] = {
 buttonText: {
 apply: "应用",
 cancel: "取消",
 swatches: "色板"
 },
 tooltipText: {
 switcher: "切换器,
 addBtn: "添加颜色",
 basic: "基本"
 }
 } | **property:** *locale*

var colorPickerObj = new ej.inputs.ColorPicker({
 locale: 'ar'
});
colorPickerObj.appendTo('#colorpicker');
ej.base.L10n.load({
 'ar': {
 'colorpicker': {
 "Apply": "تطبيق",
 "Cancel": "إلغاء",
 "ModeSwitcher": "مفتاح كهربائي الوضع"
 }
 }
}); |

| Right to left | **property:** *enableRTL*

\$("#colorpicker").ejColorPicker({
 enableRTL:

true
}); | **property:** *enableRtl*

var colorPickerObj = new ej.inputs.ColorPicker({
 enableRtl: true
});
colorPickerObj.appendTo('#colorpicker'); |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Method to open color picker popup | **Method:** *show*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.show(); | **Method:** *toggle*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.toggle(); |

| Method to close color picker popup | **Method:** *hide*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.hide(); | **Method:** *toggle*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.toggle(); |

| Enable the color picker control | **Method:** *enable*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.enable(); | Not Applicable |

| Disables the color picker control | **Method:** *disable*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.disable(); | Not Applicable |

| Method returns the selected color value as hex code | **Method:** *getValue*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.getValue(); | **Method:** *getValue*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.getValue(); |

| Method returns the selected color value in RGB format | **Method:** *getColor*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.getColor(); | **Method:** *getValue*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.getValue(null, 'RGB'); |

| Method convert the color value from hexCode to RGB | **Method:** *hexCodeToRGB*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.hexCodeToRGB("#278787"); | **Method:** *getValue*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.getValue("#278787", 'RGB'); |

| Method convert the color value from RGB to Hex code | **Method:** *RGBToHEX*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.RGBToHEX({r:38,g:133,b:133}); | **Method:** *getValue*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.getValue("rgb(38,133,133)", 'Hex'); |

| Method convert the color value from RGB to HSV | **Method:** *RGBToHSV*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj = \$("#colorpicker").data("ejColorPicker");
 colorPickerObj.RGBToHSV({h:230,s:98,v:98}); | **Method:** *getValue*

var colorPickerObj = new ej.inputs.ColorPicker({});
 colorPickerObj.appendTo('#colorpicker');
 colorPickerObj.getValue("rgb(180,71.1,52.9)", 'HSV'); |

| Method convert the color value from HSV to RGB | **Method:** *HSVToRGB*

\$("#colorpicker").ejColorPicker({});
 var colorPickerObj =

```
$("#colorpicker").data("ejColorPicker"); <br/> colorPickerObj.HSVToRGB({h:230,s:98,v:98}); | Method:  
getValue<br/><br/>var colorPickerObj = new  
ej.inputs.ColorPicker({});<br/> colorPickerObj.appendTo('#colorpicker');<br/> colorPickerObj.getValue("hs  
v(180,71.1,52.9)", 'RGB');|
```

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Event triggers before opening the ColorPicker popup | Not Applicable | **Event:** *beforeOpen*

 var
colorPickerObj = new ej.inputs.ColorPicker({
beforeOpen: function(args) {

});
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers before closing the ColorPicker popup | Not Applicable | **Event:** *beforeClose*

 var
colorPickerObj = new ej.inputs.ColorPicker({
beforeClose: function(args) {

});
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers after opening the ColorPicker popup | **Event:**
open

 \$("#colorpicker").ejColorPicker({
 open: function(args) {
 }); | **Event:**
open

 var colorPickerObj = new ej.inputs.ColorPicker({
open: function(args) {

});
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers after closing the ColorPicker popup | **Event:**
close

 \$("#colorpicker").ejColorPicker({
 close: function(args) {
 }); | Not Applicable |

| Event triggers once the component rendering is completed | **Event:**
create

 \$("#colorpicker").ejColorPicker({
 create: function(args) {
 }); | **Event:**
created

 var colorPickerObj = new ej.inputs.ColorPicker({
created: function() {

});
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers once the color picker control is destroyed | **Event:**
destroy

 \$("#colorpicker").ejColorPicker({
 destroy: function(args) {
 }); | Not
Applicable |

| Event triggers before Switching between Picker / Palette mode | Not Applicable | **Event:**
beforeModeSwitch

 var colorPickerObj = new ej.inputs.ColorPicker({
beforeModeSwitch:
function(args) {
 });
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers after color value has been selected | **Event:**
select

 \$("#colorpicker").ejColorPicker({
 select: function(args) {
 }); | **Event:**
select

 var colorPickerObj = new ej.inputs.ColorPicker({
select: function(args) {

});
 colorPickerObj.appendTo('#colorpicker');|

| Event triggers after color value has been changed | **Event:**
change

 \$("#colorpicker").ejColorPicker({
 change: function(args) {
 }); | **Event:**
change

 var colorPickerObj = new ej.inputs.ColorPicker({
change: function(args) {

});
 colorPickerObj.appendTo('#colorpicker');|

ComboBox

Tags in EJ2 JavaScript 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 `optgroup` tag that is available within the element, by default. You can preselect the option by setting the `selected` attribute to an option tag.

INDEX.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// initialize ComboBox component
let comboBoxObject: ComboBox = new 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.TS

```

import { ComboBox } from '@syncfusion/ej2-dropdowns';
// initialize ComboBox component
let comboBoxObject: ComboBox = new 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 EJ2 JavaScript 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.TS

```

import { ComboBox } from '@syncfusion/ej2-dropdowns';
// defined the array of data

```

```
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf'];
// initialize ComboBox component
let comboBoxObject: ComboBox = new ComboBox({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to ComboBox input element
    placeholder: "Select a game"
});
// render initialized ComboBox
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>
```

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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { Id: 'Game1', Game: 'Badminton' },
    { Id: 'Game2', Game: 'Basketball' },
    { Id: 'Game3', Game: 'Cricket' },
    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' }
];
//initiate the ComboBox
let comboBoxObject: ComboBox = new ComboBox({
    // 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 ComboBox input
    placeholder:"Select a game"
});
//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>
```

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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let countriesData: { [key: string]: Object }[] = [
    { 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' } }
];
//initiate the ComboBox
let comboBoxObject: ComboBox = new ComboBox({
    // bind the sports Data to datasource property
    dataSource: countriesData,
    // maps the appropriate column to fields property
    fields: { text: 'Country.Name', value: 'Code.Id' },
    //set the placeholder to ComboBox input
    placeholder: "Select a country"
});
//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>
```

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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let customers: ComboBox = new ComboBox({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the placeholder to ComboBox input
    placeholder: "Select a customer"
});
//render the component
customers.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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
```

```
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //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.TS

```

import { ComboBox } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, Predicate , DataManager, ODataV4Adaptor } from
 '@syncfusion/ej2-data';
// form predicate to fetch the grouped data
let groupPredicate = new Predicate('City',
'equal','london').or('City','equal','seattle');
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(5)
  .where(groupPredicate),
  //map the appropriate columns to fields property
  fields: { text: 'FirstName', value: 'EmployeeID', 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.TS

```

import { ComboBox } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City','EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to ComboBox input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',

```

```
//set the value to header template
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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
let sportsData = ["Football", "BasketBall", "Golf", "Cricket"];
```

```
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
  //bind the data manager instance to dataSource property
  dataSource: sportsData,
  //set the placeholder to ComboBox input
  placeholder:"Select a game",
  //sort the resulted items
  sortOrder: 'Ascending',
  //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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//initiates the component
let ComboBoxObject: ComboBox = new 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        // Here, use the wrong url to display the action failure
        template
            url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
            adaptor: new ODataV4Adaptor,
            crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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>"
});
//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>
<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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
//define the data with category
let vegetableData: { [key: string]: Object }[] = [
  { Vegetable: 'Chickpea', Category: 'Beans', Id: '1' },
  { Vegetable: 'Green bean', Category: 'Beans', Id: '2' },
  { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: '3' },
  { Vegetable: 'Horse gram', Category: 'Beans', Id: '4' },
  { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: '5' },
  { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: '6' },
  { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: '7' }
];
//initiate the ComboBox
let vegetables: ComboBox = new ComboBox({
  //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 ComboBox input
  placeholder: "Select a vegetable",
  // Set the popup list height
  popupHeight: '200px'
});
//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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// initialize ComboBox component
let ComboBoxObject: ComboBox = new 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 EJ2 JavaScript 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.TS

```

import { ComboBox, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { DataManager, Query } from '@syncfusion/ej2-data';
let searchData: { [key: string]: Object; }[] = [
    { Index: "s1", Country: "California" }, { Index: "s2", Country: "Florida" },
    { Index: "s3", Country: "Alaska" }, { Index: "s4", Country: "Georgia" }
];
let filter: ComboBox = new ComboBox({
    //set the search data to dataSource property
    dataSource: searchData,
    //map the appropriate columns to fields property
    fields: { text: "Country", value: "Index" },
    //set the placeholder to ComboBox input
    placeholder: "Select a country",
    //enable to filtering in ComboBox
    allowFiltering: true,
    //sort the resulted items
    sortOrder: 'Ascending',
    //Bind the filter event

```

```

    filtering: function (e: FilteringEventArgs) {
        let query = new Query();
        //frame the query based on search string with filter type.
        query = (e.text != "") ? query.where("Country", "startswith",
e.text, true) : query;
        //pass the filter data source, filter query to updateData method.
        e.updateData(searchData, query);
    }
});
filter.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.TS

```
import { ComboBox, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: ComboBox = new ComboBox({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(6),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to ComboBox input element
    placeholder: "Select a customer",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //sort the resulted items
    sortOrder: 'Ascending',
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // 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; }
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, true) : query;
            e.updateData(searchData, query);
        }
    }
});
filter.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.TS

```
import { ComboBox, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: ComboBox = new ComboBox({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(6),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to ComboBox input element
    placeholder: "Select a customer",
    //set the height of the popup element
    popupHeight: "250px",
    //sort the resulted items
    sortOrder: 'Ascending',
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            let query: Query = new 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);
    }
}
});
filter.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.TS

```

import { ComboBox, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';

```

```
// defined the dataManager
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: ComboBox = new ComboBox({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(6),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to ComboBox input element
    placeholder: "Select a customer",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the height of the popup element
    popupHeight: "250px",
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            //enable the case sensitive filtering by passing false to 4th
            parameter.
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, false) : query;
            e.updateData(searchData, query);
        }
    }
});
filter.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.TS

```

import { ComboBox } from '@syncfusion/ej2-dropdowns';
// create local data
let data: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aerodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
// initialize ComboBox component
let comboObj: ComboBox = new 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.TS

```
import { ComboBox, VirtualScroll } from '@syncfusion/ej2-dropdowns';
ComboBox.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i: number = 1; i <= 150; i++) {
    let item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
    //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
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.TS

```
import { ComboBox, VirtualScroll } from '@syncfusion/ej2-dropdowns';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
ComboBox.Inject(VirtualScroll);
//initiates the component
let ComboBoxObject: ComboBox = new ComboBox({
    //bind the data source property
    dataSource: new DataManager({
        url: 'https://ej2services.syncfusion.com/js/development/api/orders',
        adaptor: new 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.TS

```
import { ComboBox, VirtualScroll } from '@syncfusion/ej2-dropdowns';
ComboBox.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i = 1; i <= 150; i++) {
    let item: { [key: string]: Object } = {};
    item.id = 'id' + i;
    item.text = `Item ${i}`;
    // Generate a random number between 1 and 4 to determine the group
    const randomGroup = Math.floor(Math.random() * 4) + 1;
    switch (randomGroup) {
        case 1:
            item.group = 'Group A';
            break;
        case 2:
            item.group = 'Group B';
            break;
        case 3:
            item.group = 'Group C';
            break;
        case 4:
            item.group = 'Group D';
            break;
        default:
            break;
    }
    records.push(item);
}
//initiates the component
let ComboBoxObject: ComboBox = new 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.TS

```
import { ComboBox, VirtualScroll } from '@syncfusion/ej2-dropdowns';
ComboBox.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i: number = 1; i <= 150; i++) {
    let item = {
        id: 'id' + i,
        text: "Item " + i,
    };
};
```

```

        records.push(item);
    }
    //initiates the component
    let ComboBoxObject: ComboBox = new 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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// import L10n class for load function
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
// bind remotedata to showcase actionFailureTemplate in offline.
let customerData: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ODataV4Adaptor,
  crossDomain: true
});
let ComboBoxObj: ComboBox = new ComboBox({
  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 Query().select(['ContactName', 'CustomerID']).take(0);
  // set placeholder to ComboBox input element
  placeholder: 'Sélectionnez un client'
});
ComboBoxObj.appendTo('#comboelement');
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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize ComboBox component
let comboBoxObject: ComboBox = new ComboBox({
  //set the data to dataSource property
  dataSource: sportsData,
  // set placeholder to ComboBox input element
  placeholder: "Select a game"
  floatLabelType: "auto"
});
// render initialized ComboBox
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="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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let gameList: { [key: string]: Object }[] = [
    { 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
let ComboBoxObject: ComboBox = new ComboBox({
    //set the data to dataSource property
    dataSource: gameList,
```

```
//map to colum to fields
fields: { text: 'Game', value:'Id' },
// set placeholder to ComboBox input element
placeholder: "Select a game",
// 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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize ComboBox component
let comboBoxObject: ComboBox = new ComboBox({
    //set the data to dataSource property
    dataSource: sportsData,
    // enable the autofill behavior.
    autofill: true,
    // set placeholder to ComboBox input element
    placeholder: "Select a game"
});
// render initialized ComboBox
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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';
//define the country ComboBox data
let countryData: { [key: string]: Object }[] = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
];
//define the state ComboBox data
let stateData: { [key: string]: Object }[] = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city ComboBox data
let cityData: { [key: string]: Object }[] = [
    { 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
let countryObj: ComboBox = new ComboBox({
```

```

        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 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
    let stateObj: ComboBox = new ComboBox({
        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 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
    let cityObj: ComboBox = new ComboBox({
        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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
let sortFormatData: { [key: string]: Object }[] = [
  { 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' }
  ];
  let sortFormat: ComboBox = new ComboBox({
    dataSource: sortFormatData,
    // map the icon column to iconCSS field.
    fields: { text: 'type', iconCss: 'class', value: 'id' },
    placeholder: 'Select 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 EJ2 JavaScript 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.TS

```
import { ComboBox } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
    crossDomain: true
});

// initialize ComboBox component
let comboObj: ComboBox = new ComboBox({
    // bind the DataManager instance to dataSource property
    dataSource: data,
    // bind the Query instance to query property
    query: new 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: any) {
        let operator: Query = new
Query().from('Customers').select('ContactName');
        let start: number = 7;
        let end: number = 12;
        let listElement: HTMLElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event: any) => {
                    start = end;
                    end += 5;
                    comboObj.addItem(event.result as { [key: string]: Object
}[]);
                }).catch((e: Object) => {
                });
            }
        });
    }
});
comboObj.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>
```

Ej1 api migration in EJ2 JavaScript Combo box control

This article describes the API migration process of ComboBox component from Essential JS 1 to Essential JS 2.

DataBinding

<!-- markdownlint-disable MD010 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Default	Property: <i>datasource</i> \$('#dropdown1').ejComboBox({dataSource: List});	Property: <i>DataSource</i> var comboBoxObject = new ej.dropdowns.ComboBox({dataSource: sportsData,});comboBoxObject.appendTo('#cmbelement');
Fields for mapping	Property: <i>fields</i> \$('#dropdown1').ejComboBox({fields: { text: "text", value: "empid" }});	Property: <i>fields</i> var comboBoxObject = new ej.dropdowns.ComboBox({fields: { text: "text", value: "empid" },});comboBoxObject.appendTo('#cmbelement');
Query	Property: <i>query</i> \$('#dropdown1').ejComboBox({query: ej.Query().requiresCount()})	Property: <i>query</i> var comboBoxObject = new ej.dropdowns.ComboBox({query: new Query().from('Customers').select(['ContactName', 'CustomerID']).take(6,);comboBoxObject.appendTo('#cmbelement');

| **Begin event** | **Event:***actionBegin*
\$('#dropdown1').ejComboBox({actionBegin: "begin"}); |
Event: *actionBegin*
var comboBoxObject = new ej.dropdowns.ComboBox({actionBegin:
"begin",});comboBoxObject.appendTo('#cmbelement'); |

| **Complete event** | **Event:***actionComplete*
\$('#dropdown1').ejComboBox({actionComplete:
"begin"}); | **Event:** *actionComplete*
var comboBoxObject = new
ej.dropdowns.ComboBox({actionComplete:
"begin",});comboBoxObject.appendTo('#cmbelement'); |

| **Failure event** | **Event:***actionFailure*
\$('#dropdown1').ejComboBox({actionFailure: "begin"}); |
Event: *actionFailure*
var comboBoxObject = new ej.dropdowns.ComboBox({actionFailure:
"begin",});comboBoxObject.appendTo('#cmbelement'); |

Filtering

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *allowFiltering*
\$('#dropdown1').ejComboBox({allowFiltering: true}); |
Property: *allowFiltering*
var comboBoxObject = new ej.dropdowns.ComboBox({allowFiltering:
true,});comboBoxObject.appendTo('#cmbelement'); |

| **No records template** | **Property:**
noRecordsTemplate
\$('#dropdown1').ejComboBox({allowFiltering: "
NO DATA AVAILABLE"}); | **Property:** *noRecordsTemplate*
var comboBoxObject = new
ej.dropdowns.ComboBox({allowFiltering: true,});comboBoxObject.appendTo('#cmbelement'); |

| **Ignore casing and diacritics** | **Not Applicable** | **Property:** *ignoreAccent*
var comboBoxObject =
new ej.dropdowns.ComboBox({ignoreAccent:
true,});comboBoxObject.appendTo('#cmbelement'); |

| **Custom value addition** | **Property:**
allowCustom
\$('#dropdown1').ejComboBox({allowCustom: true}); |
<https://ej2.syncfusion.com/demos/#/material/combo-box/custom-value.html> |

| **Search event** | **Event:** *filtering*
\$('#dropdown1').ejComboBox({filtering: "filtering"}); | **Event:**
filtering
var comboBoxObject = new ej.dropdowns.ComboBox({filtering:
"filtering",});comboBoxObject.appendTo('#cmbelement'); |

Template

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *itemTemplate*
\$('#dropdown1').ejComboBox({itemTemplate: "<span
class='item' >\${FirstName}<span
class='city'>\${City}"}); | **Property:** *itemTemplate*
var comboBoxObject = new
ej.dropdowns.ComboBox({itemTemplate: "<span


```

class='name'>${FirstName}</span><span
class='city'>${City}</span></span>“,,});comboBoxObject.appendTo('#cmbelement');|

| Group Template | Property: groupTemplate<br/>${('#dropdown1').ejComboBox({groupTemplate:
“<strong>${City}</strong>”}); | Property: groupTemplate<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({groupTemplate:
“<strong>${City}</strong>“,,});comboBoxObject.appendTo('#cmbelement');|

| ValueTemplate | Not Applicable | Property: valueTemplate<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({valueTemplate: “<span>${FirstName} -
${City}</span>“,,});comboBoxObject.appendTo('#cmbelement'); |

| Header Template | Property:
headerTemplate<br/>${('#dropdown1').ejComboBox({headerTemplate: “<span
class='head'><span class='name'>Name</span><span class='city'>City</span></span>”}); |
Property: headerTemplate<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({headerTemplate: “<span class='head'><span
class='name'>Name</span><span
class='city'>City</span></span>“,,});comboBoxObject.appendTo('#cmbelement'); |

| FooterTemplate | Property: footerTemplate<br/>${('#dropdown1').ejComboBox({footerTemplate:
“<span class='foot'> Total list items: “ + sportsData.length + “</span>”}); | Property:
footerTemplate<br/>var comboBoxObject = new ej.dropdowns.ComboBox({footerTemplate:
“<span class='foot'> Total list items: “ + sportsData.length +
“</span>“,,});comboBoxObject.appendTo('#cmbelement'); |

| No records Template | Property:
noRecordsTemplate<br/>${('#dropdown1').ejComboBox({noRecordsTemplate: “<span
class='norecord'> NO DATA AVAILABLE</span>”}); | Property: noRecordsTemplate<br/>var
comboBoxObject = new ej.dropdowns.ComboBox({noRecordsTemplate: “<span
class='norecord'> NO DATA AVAILABLE</span>“,,});comboBoxObject.appendTo('#cmbelement');
|

| Auto fill | Property: autoFill<br/>${('#dropdown1').ejComboBox({autoFill: true}); | Property:
autoFill<br/>var comboBoxObject = new ej.dropdowns.ComboBox({autoFill:
true,});comboBoxObject.appendTo('#cmbelement');|

| Action failure Template | Property:
actionFailureTemplate<br/>${('#dropdown1').ejComboBox({actionFailureTemplate: “<span
class='action-failure'> Data fetch get fails</span>”}); | Property: actionFailureTemplate<br/>var
comboBoxObject = new ej.dropdowns.ComboBox({actionFailureTemplate: “<span class='action-
failure'> Data fetch get fails</span>“,,});comboBoxObject.appendTo('#cmbelement');|

```

Applying CSS

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *cssClass*
\$('#dropdown1').ejComboBox({cssClass: "className"}); |
Property: *cssClass*
var comboBoxObject = new ej.dropdowns.ComboBox({cssClass: "className",});comboBoxObject.appendTo('#cmbelement'); |

| **width** | **Property:** *width*
\$('#dropdown1').ejComboBox({width: "500px"}); | **Property:** *Width*
var comboBoxObject = new ej.dropdowns.ComboBox({width: "500px",});comboBoxObject.appendTo('#cmbelement'); |

Grouping

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *fields*
\$('#dropdown1').ejComboBox({fields: { groupBy: "text", value: "empid" }}); | **Property:** *fields*
var comboBoxObject = new ej.dropdowns.ComboBox({fields: { groupBy: "text", value: "empid" },});comboBoxObject.appendTo('#cmbelement'); |

Accessibility

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Globalization** | **Property:** *locale*
\$('#dropdown1').ejComboBox({locale: true}); | **Property:** *locale*
var comboBoxObject = new ej.dropdowns.ComboBox({locale: true});comboBoxObject.appendTo('#cmbelement'); |

| **Rtl support** | **Property:** *enableRtl*
\$('#dropdown1').ejComboBox({enableRtl: true}); | **Property:** *enableRtl*
var comboBoxObject = new ej.dropdowns.ComboBox({enableRtl: true,});comboBoxObject.appendTo('#cmbelement'); |

Placeholder

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Watermark text** | **Property:** *placeholder*
\$('#dropdown1').ejComboBox({placeholder: "Select"}); |
Property: *placeholder*
var comboBoxObject = new ej.dropdowns.ComboBox({placeholder: "Select",});comboBoxObject.appendTo('#cmbelement'); |

| **Floating of watermark text** | **Not applicable** | **Property:** *floatLabelType*
var comboBoxObject = new ej.dropdowns.ComboBox({floatLabelType: "Auto",});comboBoxObject.appendTo('#cmbelement'); |

Miscellaneous

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Enable/disable | Property: `enabled`
`<#dropdown1>.ejComboBox({enabled: true});`
| Property: `enabled`
`var comboBoxObject = new ej.dropdowns.ComboBox({enabled: true,});comboBoxObject.appendTo('#cmbelement');`

| Read only | Property: `readOnly`
`<#dropdown1>.ejComboBox({readOnly: true});`
| Property: `readOnly`
`var comboBoxObject = new ej.dropdowns.ComboBox({readOnly: true,});comboBoxObject.appendTo('#cmbelement');`

| Addition of Html attributes | Property:
`htmlAttributes`
`<#dropdown1>.ejComboBox({htmlAttributes: { disabled: "disabled"}});`
| Property: `htmlAttributes`
`var comboBoxObject = new ej.dropdowns.ComboBox({htmlAttributes: { disabled: "disabled"},});comboBoxObject.appendTo('#cmbelement');`

Sorting

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Order of sorting | Property: `sortOrder`
`<#dropdown1>.ejComboBox({sortOrder: SortOrder.Ascending});`
| Property: `sortOrder`
`var comboBoxObject = new ej.dropdowns.ComboBox({sortOrder: SortOrder.Ascending,});comboBoxObject.appendTo('#cmbelement');`

Selection

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Selecting particular index | Property: `index`
`<#dropdown1>.ejComboBox({index: 1});`
| Property: `index`
`var comboBoxObject = new ej.dropdowns.ComboBox({index: 1,});comboBoxObject.appendTo('#cmbelement');`

| Selecting particular value | Property: `value`
`<#dropdown1>.ejComboBox({value: "data"});`
| Property: `value`
`var comboBoxObject = new ej.dropdowns.ComboBox({value: "data",});comboBoxObject.appendTo('#cmbelement');`

| Selecting particular text | Property: `text`
`<#dropdown1>.ejComboBox({text: "data"});`
| Property: `text`
`var comboBoxObject = new ej.dropdowns.ComboBox({text: "data",});comboBoxObject.appendTo('#cmbelement');`

| Getting data by using value | Method:
`getItemDataByValue`
`<#dropdown1>.ejComboBox({});`
`<#dropdown1>.ejComboBox('getItemDataByValue',"data")` | **| Method:** `getDataByValue`
`var comboBoxObject = new ej.dropdowns.ComboBox({enabled: true,});comboBoxObject.appendTo('#cmbelement');`
`comboBoxObject.getDataByValue("data");`

| **Select event** | **Event:** `select`
`<#dropdown1>.ejComboBox({select: "onSelect"});` | **Event:**
`select`
`var comboBoxObject = new ej.dropdowns.ComboBox({select:`
`"onSelect",});comboBoxObject.appendTo('#cmbelement');` |

Popup

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Popup height** | **Property:** `popupHeight`
`<#dropdown1>.ejComboBox({popupHeight:`
`"300px"});` | **Property:**`popupheight`
`var comboBoxObject = new`
`ej.dropdowns.ComboBox({popupHeight:`
`"300px",});comboBoxObject.appendTo('#cmbelement');` |

| **Popup width** | **Property:** `popupWidth`
`<#dropdown1>.ejComboBox({popupWidth:`
`"300px"});` | **Property:**`popupWidth`
`var comboBoxObject = new`
`ej.dropdowns.ComboBox({popupWidth:`
`"300px",});comboBoxObject.appendTo('#cmbelement');` |

| **Popup showing manually** | **Method:** `showPopup`
`<#dropdown1>.ejComboBox({})`

`<#dropdown1>.ejComboBox("showPopup");` | **Method:** `showPopup`
`var`
`comboBoxObject = new`
`ej.dropdowns.ComboBox({});comboBoxObject.appendTo('#cmbelement');`

`cmbObj.sho`
`wPopup();` |

| **Popup hiding manually** | **Method:** `hidePopup`
`<#dropdown1>.ejComboBox({})`

`<#dropdown1>.ejComboBox("hidePopup");` | **Method:** `hidePopup`
`var comboBoxObject =`
`new ej.dropdowns.ComboBox({popupHeight:`
`"300px",});comboBoxObject.appendTo('#cmbelement');`

 `comboBoxObject.hidePopup();`
|

| **Popup hide event** | **Event:** `close`
`<#dropdown1>.ejComboBox({close: "onClose"});` | **Event:**
`close`
`var comboBoxObject = new ej.dropdowns.ComboBox({close:`
`"onclose",});comboBoxObject.appendTo('#cmbelement');` |

| **Popup shown event** | **Event:** `open`
`<#dropdown1>.ejComboBox({open: "onOpen"});` | **Event:**
`open`
`var comboBoxObject = new ej.dropdowns.ComboBox({open:`
`"onOpen",});comboBoxObject.appendTo('#cmbelement');` |

Common

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Adding new item** | **Method :** `addItem`
`<#dropdown1>.ejComboBox({})`

`<#dropdown1>.ejComboBox("addItem",{ text : "India"});` | **Method:** `addItem`
`var`
`comboBoxObject = new`

```

ej.dropdowns.ComboBox({});comboBoxObject.appendTo('#cmbelement');<br/><br/>
comboBoxObject.addItem({Id: 'id', Game: 'Golf'},2);|

| Focus out event | Not applicable | Event: Blur<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({blur: "onclose",});comboBoxObject.appendTo('#cmbelement'); |

| Focus in event | Event: Focus<br/>$('#dropdown1').ejComboBox({focus:"focus"}) | Event:
focusIn<br/>var comboBoxObject = new ej.dropdowns.ComboBox({focusIn:
"focus",});comboBoxObject.appendTo('#cmbelement'); |

| Focus out | Method: focusOut<br/>$('#dropdown1').ejComboBox({}) <br/>
<br/>$('#dropdown1').ejComboBox("focusOut");| Method: focusOut<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({open:
"onOpen",});comboBoxObject.appendTo('#cmbelement');<br/><br/> comboBoxObject.focusOut();
|

| Focus in | Method: focusIn<br/>$('#dropdown1').ejComboBox({}) <br/>
<br/>$('#dropdown1').ejComboBox("focusIn"); | Method: focusIn<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({});comboBoxObject.appendTo('#cmbelement');<br/><br/>
comboBoxObject.focusIn(); |

| Getting the data | Method : getItems<br/>$('#dropdown1').ejComboBox({}) <br/>
<br/>$('#dropdown1').ejComboBox("getItems"); | Method: getItems<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({});comboBoxObject.appendTo('#cmbelement');<br/><br/>
comboBoxObject.getItems();|

| Create event | Event: create<br/>$('#dropdown1').ejComboBox({create:"onCreate"}) | Event:
created<br/>var comboBoxObject = new ej.dropdowns.ComboBox({created:
"oncreate",});comboBoxObject.appendTo('#cmbelement'); |

| Change event | Event: change<br/>$('#dropdown1').ejComboBox({focus:"focus"}) | Event:
change<br/>var comboBoxObject = new ej.dropdowns.ComboBox({change:
"onchange",});comboBoxObject.appendTo('#cmbelement'); |

| Custom value event | Event:
customValueSpecifier<br/>$('#dropdown1').ejComboBox({customValueSpecifier:"customValueSp
ecifier"}) | Event: customValueSpecifier<br/>var comboBoxObject = new
ej.dropdowns.ComboBox({customValueSpecifier:
"customValueSpecifier",});comboBoxObject.appendTo('#cmbelement'); |

```

ContextMenu

Icons and navigation in EJ2 JavaScript 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.TS

```
import { ContextMenu, MenuItemModel, ContextMenuModel } from
 '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    {
        text: 'Cut',
        iconCss: 'e-db-icons e-cut'
    },
    {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
    },
    {
        text: 'Paste',
        iconCss: 'e-db-icons e-paste',
    }
];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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.TS

```
import { ContextMenu, MenuEventArgs, MenuItemModel, ContextMenuModel } from
 '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
  {
    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'
  }
];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
  target: '#target',
  items: menuItems,
  // To open url in blank page.
  beforeItemRender: (args: MenuEventArgs) => {
    args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
  }
}
```



```

    };
    // Initialize the ContextMenu component.
    let menuObj: ContextMenu = new 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>

```

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 EJ2 JavaScript 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.TS

```
import { createElement, enableRipple } from '@syncfusion/ej2-base';
import { ContextMenu, MenuEventArgs, MenuItemModel, ContextMenuModel } from
 '@syncfusion/ej2-navigations';
enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    {
        text: 'Save as...'
    },
    {
        text: 'View page source'
    },
    {
        text: 'Inspect'
    }
];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems,
    beforeItemRender: (args: MenuEventArgs) => {
        // To render template in li.
        let shortCutSpan: HTMLElement = createElement('span');
        let text: string = args.item.text;
        let shortCutText: string = 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');
    }
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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>

```

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>
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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

Style and appearance in EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Record, data } from './datasource.ts';
import { ContextMenu, MenuItemModel, ContextMenuModel, MenuEventArgs } from
 '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let record: Record;
let menuItems: MenuItemModel[] = [];
// Iterate data from 'datasource.ts' to construct menu item model.
for (let i: number = 0; i < data.length; i++) {
    record = data[i] as Record;
    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 });
    }
}
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new ContextMenu(menuOptions, '#contextmenu');
// To insert an item after the particular item.
menuObj.insertAfter([{separator: true}], 'Sort by');
menuObj.insertAfter([{separator: true}], 'New');
```

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 EJ2 JavaScript 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.TS

```

import { ContextMenu, MenuItemModel, ContextMenuModel } from
'@syncfusion/ej2-navigations';
import { Button } from '@syncfusion/ej2-buttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    {
        text: 'Cut'
    }

```

```
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
    ]];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    items: menuItems
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new ContextMenu(menuOptions, '#contextmenu');
// Initialize Button component.
let button: Button = new Button();
// Render initialized Button.
button.appendTo('#btnElement');
// To position ContextMenu on Button click.
document.getElementById('btnElement').onclick=function() {
    menuObj.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 EJ2 JavaScript 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.TS

```

import { ContextMenu, MenuItemModel, ContextMenuModel,
BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
  {
    text: 'Cut'
  },
  {
    text: 'Copy'
  },
  {
    text: 'Paste'
  },
  {
    text: 'Add'
  },
  {
    text: 'Edit'
  },
  {
    text: 'Delete'
  }
];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
  target: '#target .e-div',
  items: menuItems,
  beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
    // To hide/show items on right click.

```

```

        if ((args.event.target as HTMLElement).id === 'right') {
            menuObj.hideItems(['Cut', 'Copy', 'Paste']);
            menuObj.showItems(['Add', 'Edit', 'Delete']);
        } else if ((args.event.target as HTMLElement).id === 'left') {
            menuObj.showItems(['Cut', 'Copy', 'Paste']);
            menuObj.hideItems(['Add', 'Edit', 'Delete']);
        }
    }
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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">
    <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>
var ele = document.getElementById('container');
if(ele) {

```

```

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

```

Template in EJ2 JavaScript 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.TS

```

import { ContextMenu } from '@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// To Initialize ContextMenu component.
let menuObj: ContextMenu = new 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.TS

```

import { Browser } from '@syncfusion/ej2-base';
import { ContextMenu, MenuEventArgs, BeforeOpenCloseMenuEventArgs } from
 '@syncfusion/ej2-navigations';
import { MenuItemModel, ContextMenuModel } from '@syncfusion/ej2-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// To create header element.
let header: HTMLElement = document.createElement('h4');
header.textContent = 'Insert Table';
// To create table with five rows and six columns.
let table: HTMLElement = document.createElement('table');
for (let i: number = 0; i < 5; i++) {
    let row: HTMLElement = document.createElement('tr');
    table.appendChild(row);
    for (let j: number = 0; j < 6; j++) {
        let col: HTMLElement = document.createElement('td');
        row.appendChild(col);
        col.setAttribute('class', 'e-border');
    }
}
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    {
        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'
            }
        ]
    }
    ]];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems,
    beforeItemRender: (args: MenuEventArgs) => {
        // To append table on `li` rendering.
        if (args.item.id === 'table') {
            args.element.classList.add('bg-transparent');
            args.element.appendChild(header);
            args.element.appendChild(table);
        }
    }
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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.TS

```

import { closest, enableRipple} from '@syncfusion/ej2-base';
import { ContextMenu, MenuEventArgs, BeforeOpenCloseMenuEventArgs } from
 '@syncfusion/ej2-navigations';
import { ContextMenuModel, MenuItemModel } from '@syncfusion/ej2-
navigations';
import { CheckBox } from '@syncfusion/ej2-buttons';

```

```

enableRipple(true);
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    { text: 'Option 1' },
    { text: 'Option 2' },
    { text: 'Option 3' }
];
let i = 1;
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems,
    beforeItemRender: (args: MenuEventArgs) => {
        // To render CheckBox component on each li.
        let checkbox: CheckBox = new CheckBox({ label: 'Option'+ i,
checked: i%2 === 0 ? true : false });
        args.element.innerHTML = '';
        checkbox.appendTo('#checkbox'+i);
        let checkboxObj = document.getElementsByClassName('e-checkbox-
wrapper');
        args.element.appendChild(checkboxObj[0]);
        i++;
    },
    beforeClose: (args: BeforeOpenCloseMenuEventArgs) => {
        if ((args.event.target as HTMLElement).closest('.e-menu-item')) {
            // To prevent ContextMenu close on item click.
            args.cancel = true;
        }
    },
    select: (args: MenuEventArgs) => {
        let selectedElem: NodeList =
args.element.parentElement.querySelectorAll('.e-selected');
        for (let i:number=0; i < selectedElem.length; i++) {
            let ele: Element = selectedElem[i] as Element;
            ele.classList.remove('e-selected');
        }
        let checkbox: HTMLElement = args.element.childNodes[0] as
HTMLElement;
        let frame: Element = checkbox.querySelector('.e-frame');
        if (checkbox && frame.classList.contains('e-check')) {
            frame.classList.remove('e-check');
        } else if (checkbox) {
            frame.classList.add('e-check');
        }
    }
};
let menuObj: ContextMenu = new 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 EJ2 JavaScript 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.TS

```

import { ContextMenu, MenuItemModel, ContextMenuModel, MenuEventArgs } from
'@syncfusion/ej2-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize menu items.
let menuItems: MenuItemModel[] = [
    {

```



```
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
    ]];
// Initialize ContextMenu options.
let menuOptions: ContextMenuModel = {
    target: '#target',
    items: menuItems,
    beforeItemRender: (args: MenuEventArgs) => {
        if (args.item.text === 'Copy') {
            // To underline a particular character.
            args.element.innerHTML = '<u>C</u>opy';
        }
    }
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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>
</script>
<script>
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 EJ2 JavaScript 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.TS

```

import { ContextMenu, MenuEventArgs, MenuItemModel } from '@syncfusion/ej2-
navigations';
import { ListView, ListViewModel } from '@syncfusion/ej2-lists';
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// To initialize Dialog component.
let dialog: Dialog = new 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',
    visible: false
});
// Render initialized dialog.
dialog.appendTo('#dialog');
// Initialize menu items.
let menuItems: MenuItemModel[] = [
    {
        text: 'Back'
    },
    {
        text: 'Forward'
    },
    {

```

```

        text: 'Reload'
    },
    {
        separator: true
    },
    {
        text: 'Save As...'
    },
    {
        text: 'Print'
    },
    {
        text: 'Cast'
    }
    ]];
// Initialize ContextMenu options.
let menuOptions: ContextMenu = {
    target: '#target',
    items: menuItems,
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Save As...') {
            dialog.show();
        }
    }
};
// Initialize ContextMenu component.
let menuObj: ContextMenu = new 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 EJ2 JavaScript 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 EJ2 JavaScript 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>
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 EJ2 JavaScript 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>
<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 EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  cellAspectRatio: 100 / 50,
  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>

```

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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  showGridLines: true,
  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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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>
```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Panels

Position sizing of panels in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({
    cellSpacing: [10, 10],
    columns: 6,
    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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Setting header of panels in EJ2 JavaScript 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 summarizes 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { Chart, ColumnSeries, Category, LineSeries, AccumulationChart,
AccumulationTooltip, PieSeries } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
AccumulationChart.Inject(AccumulationTooltip, PieSeries );
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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: any[] = [
    { 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: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }],
    height: "162px"
}, '#column');
let lineData: any[] = [
    { 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: Chart = new Chart({
    series: [{
        dataSource: lineData,
        xName: 'x', yName: 'y',
        //Series type as line
        type: 'Line'
    }],
    height: "162px"
}, '#line');
let accChart: AccumulationChart = new 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: AccumulationChart = new 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 the 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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Add remove panels in EJ2 JavaScript 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Button } from '@syncfusion/ej2-buttons';
import { NumericTextBox } from '@syncfusion/ej2-inputs';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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');
let count: number = 7;
let data: string[] = ["Panel0", "Panel1", "Panel2", "Panel3", "Panel4",
"Panel5", "Panel6"];
let sizeX: NumericTextBox = new NumericTextBox({
  placeholder: 'Ex: 1',
  floatLabelType: 'Never',
  value: 1,
  min: 1,
  max: 5
});
sizeX.appendTo('#sizeX');
let idValue: DropDownList = new DropDownList({
  dataSource: data
});
idValue.appendTo("#value");
let sizeY: NumericTextBox = new NumericTextBox({
  //set the data to dataSource property
  placeholder: 'Ex: 1',
  floatLabelType: 'Never',
  value: 1,
  min: 1,
  max: 5
});
sizeY.appendTo('#sizeY');
let row: NumericTextBox = new NumericTextBox({
  //set the data to dataSource property
  placeholder: 'Ex: 1',
  floatLabelType: 'Never',
  value: 0,
  min: 0,
  max: 5
});
row.appendTo('#row');
let column: NumericTextBox = new 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 = () => {
  let panel: any = {
    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;
    (<string[]>idValue.dataSource).push(panel.id);
    idValue.refresh();
};
document.getElementById('remove').onclick = () => {
    dashboard.removePanel(idValue.value.toString());

    (<string[]>idValue.dataSource).splice((<string[]>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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Interaction With Panels

Dragging moving of panels in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  //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>' } ]
});
// render initialized dashboardlayout
dashboard.appendTo('#dashboard_layout');
//Dashboard Layout's dragstart event function
function onDragStart(args: any) {
```



```

    console.log("Drag start");
  }
  //Dashboard Layout's drag event function
  function onDrag(args: any) {
    console.log("Dragging");
  }
  //Dashboard Layout's dragstop event function
  function onDragStop(args: any) {
    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>

```

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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { Chart, ColumnSeries, Category, LineSeries, AccumulationChart,
AccumulationTooltip, PieSeries } from '@syncfusion/ej2-charts';
Chart.Inject(ColumnSeries, Category, LineSeries);
AccumulationChart.Inject(AccumulationTooltip, PieSeries);
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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');
let chartData: any[] = [
    { 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: Chart = new Chart({
    primaryXAxis: {
        valueType: 'Category'
    },
    series: [{
        dataSource: chartData,
        xName: 'month',
        yName: 'sales',
        type: 'Column'
    }],
    height: "162px"
}, '#column');
let lineData: any[] = [
    { 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: Chart = new Chart({
    series: [{
        dataSource: lineData,
        xName: 'x', yName: 'y',

```

```

        //Series type as line
        type: 'Line'
    }],
    height: "162px"
}, '#line');
let accChart: AccumulationChart = new AccumulationChart({
    series: [
        {
            dataSource: [{ x: 'TypeScript', y: 13, text: 'TS 13%' }, { x:
'React', y: 12.5, text: 'React 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: AccumulationChart = new 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 the 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Moving panels in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({
  cellSpacing: [10, 10],
  columns: 5,
  //Dashboard Layout's created event
  created: onCreated,
  //Dashboard Layout's change event
  change: onChange,
  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: any) {
  // movePanel("id", row, col)
  this.movePanel("layout_0", 1, 0);
}
//Dashboard Layout's change event function
function onChange(args: any) {
```

```
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>
```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Resizing of panels in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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: any) {
    console.log("Resize start");
}
//Dashboard Layout's resize event function
function onResize(args: any) {
    console.log("Resizing");
}
//Dashboard Layout's resizestop event function
function onResizeStop(args: any) {
    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 programatically

The Dashboard Layout panels can also be resized programatically 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 programatically in the Dashboard Layout's [created](#) event.

INDEX.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new DashboardLayout({

```

```

    cellSpacing: [10, 10],
    columns: 5,
    //Dashboard Layout's created event
    created: onCreate,
    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: any) {
        //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>
```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Floating of panels in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { Button } from '@syncfusion/ej2-buttons';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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');
let 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>';
let toggleBtn: Button = new Button({
  cssClass: "e-flat e-primary e-outline",
  content: "Enable Floating",
  isToggle: true
});
toggleBtn.appendTo("#toggle");
document.getElementById('toggle').onclick = () => {
  let 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>
```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Responsive adaptive in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Save restore in EJ2 JavaScript 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.TS

```

import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { Button } from '@syncfusion/ej2-buttons';
// initialize dashboardlayout component
let dashboard: DashboardLayout = new 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');
let restoreModel: any;
let saveBtn: Button = new Button({
  cssClass: "e-primary",
  content: "Save",
});
saveBtn.appendTo("#save");
let restoreBtn: Button = new Button({
  cssClass: "e-flat e-outline",
  content: "Restore",
});
restoreBtn.appendTo("#restore");
document.getElementById('save').onclick = () => {
  restorePanelModel();
};
document.getElementById('restore').onclick = () => {
  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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Style in EJ2 JavaScript 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;
}
```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

How To

Resize the panel dynamically in EJ2 JavaScript 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.TS

```
import { DashboardLayout } from '@syncfusion/ej2-layouts';
import { Button } from '@syncfusion/ej2-buttons';
let dashboardObject: DashboardLayout = new 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>'
    }
    ]
});
dashboardObject.appendTo('#defaultLayout');
let btnInstance: Button = new Button({cssClass: "e-outline e-success"});
btnInstance.appendTo('#editbtn');
btnInstance.element.onclick = () => {
    let panelContent: HTMLElement =
document.getElementById("panelContent");
    let panelHeight: number = panelContent.offsetHeight;
    let panelWidth: number = panelContent.offsetWidth;
    let diff: number = 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>

```

You can refer to our [JavaScript Dashboard Layout](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Dashboard Layout example](#) to know how to present and manipulate data.

Accessibility in EJ2 JavaScript 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 EJ2 JavaScript components](#)

DataManager

Getting started in EJ2 JavaScript Data control

This section explains you the steps required to create a simple Essential JS 2 Data Manager and demonstrate the basic usage of the Data Manager control in a JavaScript application.

Dependencies

Below is the list of minimum dependencies required to use the DataManager.

```
`javascript
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-base
|-- es6-promise (Required when window.Promise is not available)
`
```

@syncfusion/ej2-data requires the presence of a Promise feature in global environment. In the browser, window.Promise must be available.

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 files.

Step 3: Create `myapp/index.js` and `myapp/index.html` files for initializing Essential JS 2 data manager control.

Adding Syncfusion resources

The Essential JS 2 Data Manager 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 script from the [Essential Studio JavaScript \(Essential JS 2\)](#) build installed location.

After installing the Essential JS 2 product build, you can copy the data manager and its dependencies scripts file into the `resources/scripts` folder.

Refer the below code to find location data manager's script file.

Syntax:

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

Example:

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

After copying the files, then you can refer the pager's scripts into the `index.html` file.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 Data Manager's dependent script -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<!-- Essential JS 2 Data Manager's global script -->
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
`
```

Using CDN link for script

Using CDN link, you can directly refer the data manager and its dependencies script into the `index.html`.

Refer the data manager'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-data/dist/global/ej2-data.min.js>

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- Essential JS 2 Data Manager's dependent script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<!-- Essential JS 2 Pager's global script -->
<script src="http://cdn.syncfusion.com/ej2/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
</head>
<body>
```

```
</body>
```

```
</html>
```

```
,
```

Connection to a data source

The DataManager can act as gateway for both local and remote data source which will uses the query to interact with the data source.

Binding to JSON data

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.

Create **my-app/es5-datasource.js** file to bind JSON data.

Add the CSS below to the **myapp/index.html** file to style the table.

```
`html
```

```
<style>
```

```
.e-table {
```

```
border: solid 1px #e0e0e0;
```

```
border-collapse: collapse;
```

```
font-family: Roboto;
```

```
}
```

```
.e-table td,
```

```
.e-table th {
```

```
border-style: solid;
```

```
border-width: 1px 0 0;
```

```
border-color: #e0e0e0;
```

```
display: table-cell;
```

```
font-size: 14px;
```

```
line-height: 20px;
```

```
overflow: hidden;
```

```
padding: 8px 21px;
```

```
vertical-align: middle;
```

```
white-space: nowrap;
```

```
width: auto;
```

```
}
```

```
</style>
```

```
,
```

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>
  <style>
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
```



```

        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
<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>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new
Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France',
        Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new
Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
'Luisenstr. 48',
        ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany',
        Freight: 11.61, Verified: !1
    },
    {

```

```

OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new
Date(8367642e5),
    ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
    ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 65.83, Verified: !0
},
{
    OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new
Date(8367642e5),
    ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
rue du Commerce',
    ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France',
Freight: 41.34, Verified: !0
},
{
    OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new
Date(8368506e5),
    ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
'Boulevard Tirou, 255',
    ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium',
Freight: 51.3, Verified: !0
},
{
    OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new
Date(836937e6),
    ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
    ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 58.17, Verified: !0
},
{
    OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new
Date(8370234e5),
    ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
'Hauptstr. 31',
    ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry:
'Switzerland', Freight: 22.98, Verified: !1
},
{
    OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new
Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
'Starenweg 5',
    ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry:
'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new
Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry:
'Brazil', Freight: 13.97, Verified: !1
},
{

```

```

    OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new
    Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
    ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry:
    'Venezuela', Freight: 81.91, Verified: !0
  },
  {
    OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new
    Date(8375418e5),
    ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
    6',
    ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria',
    Freight: 140.51, Verified: !0
  },
  {
    OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new
    Date(8376282e5),
    ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
    ShipAddress: 'Sierras de Granada 9993',
    ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico',
    Freight: 3.25, Verified: !1
  },
  {
    OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new
    Date(8377146e5),
    ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
    'Mehrheimerstr. 369',
    ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany',
    Freight: 55.09, Verified: !0
  },
  {
    OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new
    Date(8377146e5),
    ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
    'Rua da Panificadora, 12',
    ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry:
    'Brazil', Freight: 3.05, Verified: !1
  },
  {
    OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new
    Date(8379738e5),
    ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
    ShipAddress: '2817 Milton Dr.',
    ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA',
    Freight: 48.29, Verified: !0
  }
  ]];

```

Binding to OData

DataManager can be bound to remote data source by assigning service end point URL to the **url** property. Now all **DataManager** operations will address the provided service end point.

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()).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>
  <style>
    .e-table {
      border: solid 1px #e0e0e0;
      border-collapse: collapse;
      font-family: Roboto;
    }
    .e-table td, .e-table th {
      border-style: solid;
      border-width: 1px 0 0;
      border-color: #e0e0e0;
      display: table-cell;
```

```

        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
<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

The data filtering is a trivial operation which will let us to get reduced view of data based on filter criteria. The filter expression can be built easily using `where` method of `Query` class.

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().where('EmployeeID', 'equal', 3));
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>
<style>
    .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
<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>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new
Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France',
        Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new
Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
'Luisenstr. 48',
        ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany',
        Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new
Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new
Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
rue du Commerce',
        ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France',
        Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new
Date(8368506e5),

```

```

        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
        'Boulevard Tirou, 255',
        ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium',
        Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new
        Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
        'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new
        Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
    {
        OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new
        Date(8371098e5),
        ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
        'Starenweg 5',
        ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry:
        'Switzerland', Freight: 148.33, Verified: !0
    },
    {
        OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new
        Date(837369e6),
        ShipName: 'Wellington Importadora', ShipCity: 'Resende',
        ShipAddress: 'Rua do Mercado, 12',
        ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry:
        'Brazil', Freight: 13.97, Verified: !1
    },
    {
        OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new
        Date(8374554e5),
        ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
        ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
        ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry:
        'Venezuela', Freight: 81.91, Verified: !0
    },
    {
        OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new
        Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
        6',
        ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria',
        Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new
        Date(8376282e5),

```



```

        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
        ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico',
        Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new
        Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
        'Mehrheimerstr. 369',
        ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany',
        Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new
        Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua da Panificadora, 12',
        ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry:
        'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new
        Date(8379738e5),
        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
        ShipAddress: '2817 Milton Dr.',
        ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA',
        Freight: 48.29, Verified: !0
    }
    ]];

```

Sort

The data can be ordered either in ascending or descending using `sortBy` method of `Query` class.

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().sortBy('CustomerID').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>
<style>
    .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
<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>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new
Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
        ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France',
        Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new
Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
'Luisenstr. 48',
        ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany',
        Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new
Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new
Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
rue du Commerce',
        ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France',
        Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new
Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
'Boulevard Tirou, 255',
        ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium',
        Freight: 51.3, Verified: !0
    },

```

```

{
    OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new
Date(836937e6),
    ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
    ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 58.17, Verified: !0
},
{
    OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new
Date(8370234e5),
    ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
'Hauptstr. 31',
    ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry:
'Switzerland', Freight: 22.98, Verified: !1
},
{
    OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new
Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
'Starenweg 5',
    ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry:
'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new
Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry:
'Brazil', Freight: 13.97, Verified: !1
},
{
    OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new
Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry:
'Venezuela', Freight: 81.91, Verified: !0
},
{
    OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new
Date(8375418e5),
    ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
6',
    ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria',
Freight: 140.51, Verified: !0
},
{
    OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new
Date(8376282e5),
    ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
ShipAddress: 'Sierras de Granada 9993',
    ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico',
Freight: 3.25, Verified: !1
},
{

```

```

    OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new
    Date(8377146e5),
    ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
    'Mehrheimerstr. 369',
    ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany',
    Freight: 55.09, Verified: !0
  },
  {
    OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new
    Date(8377146e5),
    ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
    'Rua da Panificadora, 12',
    ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry:
    'Brazil', Freight: 3.05, Verified: !1
  },
  {
    OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new
    Date(8379738e5),
    ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
    ShipAddress: '2817 Milton Dr.',
    ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA',
    Freight: 48.29, Verified: !0
  }
];

```

Page

The `page` method of the Query class is used to get range of data based on the page number and the total page size.

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().page(1, 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>
<style>
    .e-table {
        border: solid 1px #e0e0e0;
        border-collapse: collapse;
        font-family: Roboto;
    }
    .e-table td, .e-table th {
        border-style: solid;
        border-width: 1px 0 0;
        border-color: #e0e0e0;
        display: table-cell;
        font-size: 14px;
        line-height: 20px;
        overflow: hidden;
        padding: 8px 21px;
        vertical-align: middle;
        white-space: nowrap;
        width: auto;
    }
</style>
<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>

```

ES5-DATASOURCE.JS

```

var data = [
    {
        OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new
Date(8364186e5),
        ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
ShipAddress: '59 rue de l Abbaye',
        ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France',
Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new
Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
'Luisenstr. 48',
        ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany',
Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new
Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new
Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
rue du Commerce',
        ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France',
Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new
Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
'Boulevard Tirou, 255',
        ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium',
Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new
Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',

```

```

        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
        'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new
        Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
        'Hauptstr. 31',
        ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry:
        'Switzerland', Freight: 22.98, Verified: !1
    },
    {
        OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new
        Date(8371098e5),
        ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
        'Starenweg 5',
        ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry:
        'Switzerland', Freight: 148.33, Verified: !0
    },
    {
        OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new
        Date(837369e6),
        ShipName: 'Wellington Importadora', ShipCity: 'Resende',
        ShipAddress: 'Rua do Mercado, 12',
        ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry:
        'Brazil', Freight: 13.97, Verified: !1
    },
    {
        OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new
        Date(8374554e5),
        ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
        ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
        ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry:
        'Venezuela', Freight: 81.91, Verified: !0
    },
    {
        OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new
        Date(8375418e5),
        ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
        6',
        ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria',
        Freight: 140.51, Verified: !0
    },
    {
        OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new
        Date(8376282e5),
        ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
        ShipAddress: 'Sierras de Granada 9993',
        ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico',
        Freight: 3.25, Verified: !1
    },
    {
        OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new
        Date(8377146e5),
        ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
        'Mehrheimerstr. 369',

```



```

        ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany',
        Freight: 55.09, Verified: !0
    },
    {
        OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new
        Date(8377146e5),
        ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
        'Rua da Panificadora, 12',
        ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry:
        'Brazil', Freight: 3.05, Verified: !1
    },
    {
        OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new
        Date(8379738e5),
        ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
        ShipAddress: '2817 Milton Dr.',
        ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA',
        Freight: 48.29, Verified: !0
    }
    ]];

```

Component binding

DataManager component can be used with Syncfusion components which supports data binding.

In the following samples, the grid component is bound. To render the grid with the necessary configurations, please refer to the [Grid Getting Started](#) documentation.

Local data binding

A DataSource can be created in-line with other Syncfusion component configuration settings.

INDEX.JS

```

(document.getElementById('datatable')).style.display = 'none';
var grid = new ej.grids.Grid({
    dataSource: new ej.data.DataManager(data),
    columns: [
        { field: 'OrderID', headerText: 'Order ID', textAlign:
        'Right', width: 90, type: 'number' },
        { field: 'CustomerID', width: 120, headerText: 'Customer
        ID', type: 'string' },
        { field: 'Freight', headerText: 'Freight', textAlign:
        'Right', width: 90, format: 'C' },
        { field: 'OrderDate', headerText: 'Order Date', width: 120,
        format: 'yMd' },
    ],
    height: 315,
    allowPaging: false
});
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>
        <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>

```

ES5-DATASOURCE.JS

```

var data = [
    {

```

```

    OrderID: 10248, CustomerID: 'VINET', EmployeeID: 5, OrderDate: new
Date(8364186e5),
    ShipName: 'Vins et alcools Chevalier', ShipCity: 'Reims',
ShipAddress: '59 rue de l Abbaye',
    ShipRegion: 'CJ', ShipPostalCode: '51100', ShipCountry: 'France',
Freight: 32.38, Verified: !0
    },
    {
        OrderID: 10249, CustomerID: 'TOMSP', EmployeeID: 6, OrderDate: new
Date(836505e6),
        ShipName: 'Toms Spezialitäten', ShipCity: 'Münster', ShipAddress:
'Luisenstr. 48',
        ShipRegion: 'CJ', ShipPostalCode: '44087', ShipCountry: 'Germany',
Freight: 11.61, Verified: !1
    },
    {
        OrderID: 10250, CustomerID: 'HANAR', EmployeeID: 4, OrderDate: new
Date(8367642e5),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 65.83, Verified: !0
    },
    {
        OrderID: 10251, CustomerID: 'VICTE', EmployeeID: 3, OrderDate: new
Date(8367642e5),
        ShipName: 'Victuailles en stock', ShipCity: 'Lyon', ShipAddress: '2,
rue du Commerce',
        ShipRegion: 'CJ', ShipPostalCode: '69004', ShipCountry: 'France',
Freight: 41.34, Verified: !0
    },
    {
        OrderID: 10252, CustomerID: 'SUPRD', EmployeeID: 4, OrderDate: new
Date(8368506e5),
        ShipName: 'Suprêmes délices', ShipCity: 'Charleroi', ShipAddress:
'Boulevard Tirou, 255',
        ShipRegion: 'CJ', ShipPostalCode: 'B-6000', ShipCountry: 'Belgium',
Freight: 51.3, Verified: !0
    },
    {
        OrderID: 10253, CustomerID: 'HANAR', EmployeeID: 3, OrderDate: new
Date(836937e6),
        ShipName: 'Hanari Carnes', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua do Paço, 67',
        ShipRegion: 'RJ', ShipPostalCode: '05454-876', ShipCountry:
'Brazil', Freight: 58.17, Verified: !0
    },
    {
        OrderID: 10254, CustomerID: 'CHOPS', EmployeeID: 5, OrderDate: new
Date(8370234e5),
        ShipName: 'Chop-suey Chinese', ShipCity: 'Bern', ShipAddress:
'Hauptstr. 31',
        ShipRegion: 'CJ', ShipPostalCode: '3012', ShipCountry:
'Switzerland', Freight: 22.98, Verified: !1
    },
    {

```

```

    OrderID: 10255, CustomerID: 'RICSU', EmployeeID: 9, OrderDate: new
Date(8371098e5),
    ShipName: 'Richter Supermarkt', ShipCity: 'Genève', ShipAddress:
'Starenweg 5',
    ShipRegion: 'CJ', ShipPostalCode: '1204', ShipCountry:
'Switzerland', Freight: 148.33, Verified: !0
},
{
    OrderID: 10256, CustomerID: 'WELLI', EmployeeID: 3, OrderDate: new
Date(837369e6),
    ShipName: 'Wellington Importadora', ShipCity: 'Resende',
ShipAddress: 'Rua do Mercado, 12',
    ShipRegion: 'SP', ShipPostalCode: '08737-363', ShipCountry:
'Brazil', Freight: 13.97, Verified: !1
},
{
    OrderID: 10257, CustomerID: 'HILAA', EmployeeID: 4, OrderDate: new
Date(8374554e5),
    ShipName: 'HILARION-Abastos', ShipCity: 'San Cristóbal',
ShipAddress: 'Carrera 22 con Ave. Carlos Soublette #8-35',
    ShipRegion: 'Táchira', ShipPostalCode: '5022', ShipCountry:
'Venezuela', Freight: 81.91, Verified: !0
},
{
    OrderID: 10258, CustomerID: 'ERNSH', EmployeeID: 1, OrderDate: new
Date(8375418e5),
    ShipName: 'Ernst Handel', ShipCity: 'Graz', ShipAddress: 'Kirchgasse
6',
    ShipRegion: 'CJ', ShipPostalCode: '8010', ShipCountry: 'Austria',
Freight: 140.51, Verified: !0
},
{
    OrderID: 10259, CustomerID: 'CENTC', EmployeeID: 4, OrderDate: new
Date(8376282e5),
    ShipName: 'Centro comercial Moctezuma', ShipCity: 'México D.F.',
ShipAddress: 'Sierras de Granada 9993',
    ShipRegion: 'CJ', ShipPostalCode: '05022', ShipCountry: 'Mexico',
Freight: 3.25, Verified: !1
},
{
    OrderID: 10260, CustomerID: 'OTTIK', EmployeeID: 4, OrderDate: new
Date(8377146e5),
    ShipName: 'Ottilies Käseladen', ShipCity: 'Köln', ShipAddress:
'Mehrheimerstr. 369',
    ShipRegion: 'CJ', ShipPostalCode: '50739', ShipCountry: 'Germany',
Freight: 55.09, Verified: !0
},
{
    OrderID: 10261, CustomerID: 'QUEDE', EmployeeID: 4, OrderDate: new
Date(8377146e5),
    ShipName: 'Que Delícia', ShipCity: 'Rio de Janeiro', ShipAddress:
'Rua da Panificadora, 12',
    ShipRegion: 'RJ', ShipPostalCode: '02389-673', ShipCountry:
'Brazil', Freight: 3.05, Verified: !1
},
{

```

```

    OrderID: 10262, CustomerID: 'RATTC', EmployeeID: 8, OrderDate: new
    Date(8379738e5),
    ShipName: 'Rattlesnake Canyon Grocery', ShipCity: 'Albuquerque',
    ShipAddress: '2817 Milton Dr.',
    ShipRegion: 'NM', ShipPostalCode: '87110', ShipCountry: 'USA',
    Freight: 48.29, Verified: !0
  }];

```

Remote data binding

To bind remote data to Syncfusion component, you can assign a service data as an instance of **DataManager** to the **dataSource** property.

INDEX.JS

```

const SERVICE_URI =
'https://services.syncfusion.com/js/production/api/orders';
var grid = new ej.grids.Grid({
  dataSource: new ej.data.DataManager({ url: SERVICE_URI }),
  columns: [
    { field: 'OrderID', headerText: 'Order ID', textAlign:
    'Right', width: 90, type: 'number' },
    { field: 'CustomerID', width: 120, headerText: 'Customer
    ID', type: 'string' },
    { field: 'EmployeeID', headerText: 'Employee ID', textAlign:
    'Right', width: 90 }
  ],
  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>

```

Data binding in EJ2 JavaScript 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.TS

```

import { DataManager, Query } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let result: Object[] = new DataManager(data).executeLocal(new
Query().take(8));
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
result.forEach((data: Object) => {
    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.TS

```
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.syncfusion.com/js/production/api/orders';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI }).executeQuery(new
Query().take(8)).then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
        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 EJ2 JavaScript 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.TS

```
import { DataManager, Query, JsonAdaptor } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let result: Object[] = new DataManager({ json: data, adaptor: new
JsonAdaptor })
                                .executeLocal(new Query().take(8));
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
result.forEach((data: Object) => {
    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.TS

```
import { DataManager, Query, ReturnOption, ODataAdaptor } from
 '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
 '<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
 'https://services.syncfusion.com/js/production/api/Orders';
let table: HTMLElement =
 (<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataAdaptor
 }).executeQuery(new Query().take(8)).then((e: ReturnOption) => {
    (<Object[]>e.result.items).forEach((data: Object) => {
        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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor
}).executeQuery(new Query().take(8)).then((e: ReturnOption) => {
(<Object[]>e.result).forEach((data: Object) => {

```

```

        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
<code>RequiresCounts</code>	If it is <code>true</code> then the total count of records will be included in response.
<code>Skip</code>	Holds the number of records to skip.
<code>Take</code>	Holds the number of records to take.
<code>Sorted</code>	Contains details about current sorted column and its direction.
<code>Where</code>	Contains details about current filter column name and its constraints.
<code>Group</code>	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.TS

```
import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
class SerialNoAdaptor extends ODataV4Adaptor {

    public processResponse(): Object {
        let i: number = 0;
        //calling base class processResponse function
        let original: Object[] = super.processResponse.apply(this,
arguments);
        //Adding serial number
        original.forEach((item: Object) => item['SNO'] = ++i);
        return original;
    }
}
let template: string =
'<tr><td>${SNO}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
table.innerHTML = '<tr><th>SNO</th><th>Customer ID</th><th>Employee
ID</th></tr>';
new DataManager({ url: SERVICE_URI, adaptor: new SerialNoAdaptor
}).executeQuery(new Query().take(8)).then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
        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 EJ2 JavaScript 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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);

```

```

const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().from('Orders').take(8)).then((e: ReturnOption)
=> {
    (<Object[]>e.result).forEach((data: Object) => {
        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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().select(['OrderID', 'CustomerID',
'EmployeeID'])).take(8)
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${Employee.FirstName}</td>
</tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';

```

```

let table: HTMLElement =
  (<HTMLElement>document.getElementById('datatable'));
table.innerHTML = '<tr><th>OrderID</th><th>CustomerID</th><th>Employee
Name</th></tr>';
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
  .executeQuery(new Query().expand('Employee').select(['OrderID',
'CustomerID', 'Employee.FirstName']).take(8))
  .then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
      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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().sortBy('CustomerID', 'descending').take(8))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';

```

```

let template: string =
  '<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
  'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
  (<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
  .executeQuery(new Query().where('EmployeeID', 'equal', 3).take(8))
  .then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
      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.TS

```

import { DataManager, Query, ODataV4Adaptor, Predicate, ReturnOption } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
//Building complex filter criteria using `Predicate`
let predicate: Predicate = new Predicate('EmployeeID', 'equal', 3);
predicate = predicate.or('EmployeeID', 'equal', 2);
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().where(predicate).take(8))

```

```
.then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
        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>
            </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.TS

```

import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager(data)
    .executeQuery(new Query().search('VI', ['CustomerID']))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string = '<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}';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));

```

```

new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().group('CustomerID').take(8))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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>
            </tbody>
        </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>

```

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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
 '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
 '<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
 'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
 (<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().page(2, 8))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let footerFn: Function = compile('<tr><td></td><td></td><td>Minimum:
${min}</td></tr>');
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().take(5).requiresCount().aggregate('min',
'EmployeeID'))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>'
let group: string = '<tr><td colspan=3>' +
'<table id="datatable" class="e-
table"><tr><th>ID</th><th>Price</th><th>Quantity</th></tr>' +
'${for(detail of
Order_Details)}<tr><td>${detail.ProductID}</td><td>${detail.UnitPrice}</td><
td>${detail.Quantity}</td></tr>${/for}' +
'<table></td></tr>';
let compiledFunction: Function = compile(template);
let groupFn = compile(group);
const SERVICE_URI =
'https://services.odata.org/V4/Northwind/Northwind.svc/';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
    .executeQuery(new Query().from('Orders').take(3).hierarchy(
        new Query()
            .foreignKey("OrderID")

```

```

        .from("Order_Details")
        .sortBy("Quantity"),
        function () {
            // Selective loading of child elements
            return [10248, 10249, 10250]
        }
    ))
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            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 EJ2 JavaScript 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.TS

```

import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let table: HTMLTableElement =
(<HTMLTableElement>document.getElementById('datatable'));
let dm: DataManager = new DataManager(data.slice(0, 4));
dm.executeQuery(new Query())
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
        });
    });
interface IOrder {
    OrderID: string;
    CustomerID: string;
    EmployeeID: string;
}
let button: HTMLInputElement =
<HTMLInputElement>document.getElementById('manipulate');
let orderid: HTMLInputElement =
<HTMLInputElement>document.getElementById('OrderID');

```

```
let cusid: HTMLInputElement =
<HTMLInputElement>document.getElementById('CustomerID');
let empid: HTMLInputElement =
<HTMLInputElement>document.getElementById('EmployeeID');
button.onclick = () => {
    let data: IOrder = {
        OrderID: orderid.value,
        CustomerID: cusid.value,
        EmployeeID: empid.value
    };
    if (!data.OrderID) { return; }
    dm.insert(data);
    dm.executeQuery(new Query())
        .then((e: ReturnOption) => {
            table.tBodies[0].innerHTML = '';
            (<Object[]>e.result).forEach((data: Object) => {
                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.TS

```

import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';

```

```

let template: string =
  '<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let table: HTMLTableElement =
  (<HTMLTableElement>document.getElementById('datatable'));
let dm: DataManager = new DataManager(data.slice(0, 4));
dm.executeQuery(new Query())
  .then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
    });
  });
let button: HTMLInputElement =
  <HTMLInputElement>document.getElementById('manipulate');
button.value = 'Update';
let orderid: HTMLInputElement =
  <HTMLInputElement>document.getElementById('OrderID');
let cusid: HTMLInputElement =
  <HTMLInputElement>document.getElementById('CustomerID');
let empid: HTMLInputElement =
  <HTMLInputElement>document.getElementById('EmployeeID');
interface IOrder {
  OrderID: number;
  CustomerID: string;
  EmployeeID: number;
}
button.onclick = () => {
  let data: IOrder = {
    OrderID: +orderid.value,
    CustomerID: cusid.value,
    EmployeeID: +empid.value
  };
  if (!data.OrderID) { return; }
  dm.update('OrderID', data);
  dm.executeQuery(new Query())
    .then((e: ReturnOption) => {
      table.tBodies[0].innerHTML = '';
      (<Object[]>e.result).forEach((data: Object) => {

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.TS

```
import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let table: HTMLTableElement =
(<HTMLTableElement>document.getElementById('datatable'));
let dm: DataManager = new DataManager(data.slice(0, 4));
dm.executeQuery(new Query())
  .then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {

table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
    });
  });
let button: HTMLInputElement =
<HTMLInputElement>document.getElementById('manipulate');
button.value = 'Remove';
let orderid: HTMLInputElement =
<HTMLInputElement>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 Query())
    .then((e: ReturnOption) => {
      table.tBodies[0].innerHTML = '';
      (<Object[]>e.result).forEach((data: Object) => {

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.TS

```

import { DataManager, Query, ReturnOption } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
import { data } from './datasource.ts';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
let table: HTMLTableElement =
(<HTMLTableElement>document.getElementById('datatable'));
let dm: DataManager = new DataManager({ json: (<Object[]>data).slice(0, 4)
});
dm.executeQuery(new Query()
    .then((e: ReturnOption) => {
        (<Object[]>e.result).forEach((data: Object) => {
            table.tBodies[0].appendChild(compiledFunction(data)[0].firstChild);
        });
    });
let changes: { changedRecords: Object[], addedRecords: Object[],
deletedRecords: Object[] } = {
    changedRecords: [], addedRecords: [], deletedRecords: []
};
let orderid: HTMLInputElement =
<HTMLInputElement>document.getElementById('OrderID');
let cusid: HTMLInputElement =
<HTMLInputElement>document.getElementById('CustomerID');
let empid: HTMLInputElement =
<HTMLInputElement>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 Query())
        .then((e: ReturnOption) => {
            table.tBodies[0].innerHTML = '';
            (<Object[]>e.result).forEach((data: Object) => {
                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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { DataManager, ReturnOption, ODataV4Adaptor } from '@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';
let template: string =
'<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/?$top=7';
let table: HTMLElement =
(<HTMLElement>document.getElementById('datatable'));
let dm: DataManager = new DataManager({ url: SERVICE_URI, adaptor: new
ODataV4Adaptor, offline: true });
dm.ready.then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
        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.TS

```

import { DataManager, Query, ReturnOption, ODataV4Adaptor } from
'@syncfusion/ej2-data';
import { compile } from '@syncfusion/ej2-base';

```

```

let template: string =
  '<tr><td>${OrderID}</td><td>${CustomerID}</td><td>${EmployeeID}</td></tr>';
let compiledFunction: Function = compile(template);
const SERVICE_URI: string =
  'https://services.odata.org/V4/Northwind/Northwind.svc/Orders/';
let table: HTMLElement =
  (<HTMLElement>document.getElementById('datatable'));
new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
  .executeQuery(new Query().addParams('$top', '7'))
  .then((e: ReturnOption) => {
    (<Object[]>e.result).forEach((data: Object) => {
      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 EJ2 JavaScript 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.TS

```

import { DatePicker } from '@syncfusion/ej2-calendars';
let today: Date = new Date();

```

```
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
// creates a datepicker with min max property
let datePickerObject: DatePicker = new DatePicker({
    //sets the min.
    min: new Date(currentYear, currentMonth, 7),
    //sets the max.
    max: new Date(currentYear, currentMonth, 27),
    //sets the value.
    value: new Date(new Date().setDate(14))
});
datePickerObject.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 EJ2 JavaScript Datpicker 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';  
// creates a DatePicker component with custom format.  
let datePickerObject: DatePicker = new DatePicker({  
    value: new Date(),  
    // sets the format.  
    format: 'yyyy-MM-dd'  
});  
datePickerObject.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 EJ2 JavaScript 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.TS

```
import { DatePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DatePicker.Inject(MaskedDateTime);
let datepickerObject: DatePicker = new 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>
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.TS

```
import { DatePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DatePicker.Inject(MaskedDateTime);
let datePickerObject: DatePicker = new DatePicker({
    enableMask: true
});
datePickerObject.appendTo('#mask');
let datePickerFormat: DatePicker = new 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.TS

```
import { DatePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DatePicker.Inject(MaskedDateTime);
let datePickerObject: DatePicker = new DatePicker({
    enableMask: true
});
datePickerObject.appendTo('#mask');
let datePickerPlaceholder: DatePicker = new 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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
  'de': {
    'datepicker': {
      placeholder: 'Wählen Sie ein Datum aus',
      today: 'heute'
    }
  }
});
// creates the simple DatePicker component
```

```
let datePickerObject: DatePicker = new DatePicker({
    // sets the locale property.
    locale: 'de',
    value: new Date()
});
datePickerObject.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">
  <!--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.TS

```

import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DatePicker } from '@syncfusion/ej2-calendars';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';

```



```
import * as numbers from './numbers.json';
import * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
  'he': {
    'datepicker': {
      placeholder: 'הזן תאריך',
      today: 'היום'
    }
  }
});
// creates the simple DatePicker component
let datePickerObject: DatePicker = new DatePicker({
  //sets the locale
  locale: 'he',
  //sets the enableRtl
  enableRtl: true,
  //sets the value
  value: new Date()
});
datePickerObject.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">
  <!--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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
// creates a datepicker with strictMode property
let datepickerObject: DatePicker = new DatePicker({
    // sets the strictMode
    strictMode: true,
    // sets the format
    format: 'dd/MM/yyyy',
    // sets the value
    value: new Date('5/28/2017'),
    //sets the min
    min: new Date('5/5/2017'),
    //sets the max
    max: new Date('5/25/2017')
});
datepickerObject.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.TS

```

import { DatePicker } from '@syncfusion/ej2-calendars';
// creates a datepicker with strictMode property
let datePickerObject: DatePicker = new DatePicker({
    // sets the value
    value: new Date('5/28/2017'),
    //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',
    // sets the placeholder
    placeholder: 'Enter date'
});
datePickerObject.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 EJ2 JavaScript Datpicker 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.TS

```
import { DatePicker, RenderDayCellEventArgs } from '@syncfusion/ej2-calendars';
// creates datepicker with placeholder.
let datepickerObject: DatePicker = new DatePicker({
    // Bind the renderDayCell event to customize the each day cell.
    renderDayCell: onRenderCell,
    // sets the placeholder
    placeholder: 'Enter date',
    cssClass: 'e-custom-style'
});
datepickerObject.appendTo('#element');
function onRenderCell(args: RenderDayCellEventArgs): void {
    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".
    }
}
```

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.TS

```

import { DatePicker } from '@syncfusion/ej2-calendars';
let month: number = new Date().getMonth();
let fullYear: number = new Date().getFullYear();
// creates a datepicker with min max property
let datePickerObject: DatePicker = new DatePicker({
    // 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)
    floatLabelType: 'Auto'
    placeholder:"Select Date"
});
datePickerObject.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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';  
//creates a datepicker with year view.  
let datePickerObj: DatePicker = new DatePicker({  
    // sets the placeholder  
    placeholder: 'Enter date',  
    //sets the start  
    start: 'Decade'  
});  
datePickerObj.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>
<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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
//creates a datepicker with decade view and navigate up to year view.
let datePickerObj: DatePicker = new DatePicker({
    //sets the start
    start: 'Decade',
    //sets the depth
    depth: 'Year',
    // sets the placeholder
    placeholder: 'Enter date',
});
datePickerObj.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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
// creates a DatePicker component
let datePickerObject: DatePicker = new DatePicker({
    // sets the placeholder
    placeholder: 'Enter date'
});
datePickerObject.appendTo('#element');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the component.
        datePickerObject.element.focus();
    }
};
```

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 EJ2 JavaScript 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.

```
`typescript
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 EJ2 JavaScript Datepicker control

To disable the DatePicker, use its [enable](#) property.

The following example demonstrates the DatePicker in a disabled state.

INDEX.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';  
// creates datepicker with enabled property  
let datepickerObject: DatePicker = new DatePicker({  
    // sets the enabled  
    enabled: false  
});  
datepickerObject.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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';  
// creates datepicker with placeholder.  
let datePickerObject: DatePicker = new DatePicker({  
    // sets the palceholder property.  
    placeholder: 'Enter date'  
});  
datePickerObject.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 EJ2 JavaScript DatePicker control

The following example demonstrates how to set `readonly` in DatePicker component. You can achieve this by using `readonly` property.

INDEX.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
// creates datepicker with readonly.
let datePickerObject: DatePicker = new DatePicker({
    // sets the readonly.
    readonly: true,
    // sets the value.
    value: new Date(),
    // sets the palceholder property.
    placeholder: 'Enter date'
});
datePickerObject.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 EJ2 JavaScript 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.TS

```
import { DatePicker, PreventableEventArgs } from '@syncfusion/ej2-  
calendars';  
// creates datepicker with readonly.  
let datePickerObject: DatePicker = new DatePicker({  
  close: function (args: PreventableEventArgs) {  
    // prevent the popup close  
    args.preventDefault();  
  },  
  // sets the palceholder property.  
  placeholder: 'Enter date'  
});  
datePickerObject.appendTo('#element');  
// open the datepicker popup  
datePickerObject.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 EJ2 JavaScript 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.TS

```
import { DatePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// creates datepicker component
let datePickerObject: DatePicker = new DatePicker({
    dayHeaderFormat: "Short"
});
datePickerObject.appendTo('#element');
let formatLabel: DropDownList = new DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: (args: any) => {
        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 EJ2 JavaScript 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.TS

```

import { DatePicker } from '@syncfusion/ej2-calendars';
// creates datepicker component
let datepickerObject: DatePicker = new DatePicker({
    focus: function () {
        // To open the popup upon input click
        datepickerObject.show();
    },

```

```
// sets the palceholder property.
placeholder: 'Enter date'
});
datepickerObject.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 EJ2 JavaScript Datpicker 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.TS

```

import { DatePicker } from '@syncfusion/ej2-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
// creates datepicker with readonly.
let datePickerObject: DatePicker = new DatePicker({
    // sets the palceholder property.
    placeholder: 'Enter date',
    // sets the value
    value: new Date()
});
datePickerObject.appendTo('#element');
let options: FormValidatorModel = {
    rules: {
        'datevalue': { required: true }
    },
    customPlacement: function (inputElement, errorElement) {
        //to place the error message in custom position.
        (<HTMLElement>inputElement).parentElement.parentElement.appendChild(errorElement);
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);

```

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>

```

Ej1 api migration in EJ2 JavaScript Datepicker control

This article describes the API migration process of DatePicker component from Essential JS 1 to Essential JS 2.

Date Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	property: valuejavascript\$("#datepicker").ejDatePicker({ value: new Date("5/5/2014") });	property: valuejavascriptvar datepicker = new ej.calendars.DatePicker({ value: new Date("05/11/2017")});datepicker.appendTo('#datepicker');

Date Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display the date format	property: dateFormatjavascript\$("#datepicker").ejDatePicker({ dateFormat: "dd/MM/yyyy"});	property: formatjavascriptvar datepicker = new ej.calendars.DatePicker({ value: new Date("05/11/2017"); format: 'yyyy-MM-dd'});datepicker.appendTo('#element');
Day header format	property: dayHeaderFormatjavascript\$("#datepicker").ejDatePicker({ dayHeaderFormat: ej.DatePicker.Header.Short});	Not Applicable

Calendar Views

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Start	property: startLeveljavascript\$("#datepicker").ejDatePicker({ startLevel: ej.DatePicker.Level.Year});	property: startjavascriptvar datepicker = new ej.calendars.DatePicker({ start:'Decade'});datepicker.appendTo('#element');
Depth	property: depthLeveljavascript\$("#datepicker").ejDatePicker({depthLevel: ej.DatePicker.Level.Year});	property: depthjavascriptvar datepicker = new ej.calendars.DatePicker({ depth:'Year'});datepicker.appendTo('#element');

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum date	property: minDatejavascript\$("#datepicker").ejDatePicker({ minDate: new Date("5/1/2013")});	property: minjavascriptvar datepicker = new ej.calendars.DatePicker({ min: new Date("5/1/2013")});datepicker.appendTo('#element');
Maximum date	property: maxDatejavascript\$("#datepicker").ejDatePicker({ maxDate : new Date("5/30/2015")});	property: maxjavascriptvar datepicker = new ej.calendars.DatePicker({ max : new Date("5/30/2015")});datepicker.appendTo('#element');

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Block-out dates	property: blackoutDatesjavascript\$("#datepicker").ejDatePicker({ blackoutDates: [new Date(2016, 4, 10), new Date(2016, 4, 15), new Date(2016, 4, 20), new Date(2016, 4, 22), new Date(2016, 5, 12), new Date(2016, 5, 24)]});	Can be achieved byjavascriptvar datepicker = new ej.calendars.DatePicker({ renderDayCell: disableDate});datepicker.appendTo('#element');function disableDate(args) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; } }

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	property: cssClassjavascript\$("#datepicker").ejDatePicker({ cssClass: "gradient-lime"});	property: cssClassjavascriptvar datepicker = new ej.calendars.DatePicker({ cssClass: 'e-custom-style'});datepicker.appendTo('#element');
Event callback for each cell creation	Not Applicable	Event: renderDayCelljavascriptvar datepicker = new ej.calendars.DatePicker({ renderDayCell: onRenderCell});datepicker.appendTo('#element');function onRenderCell(args) {/code block/}
Show/Hide the today button	property: showFooterjavascript\$("#datepicker").ejDatePicker({ showFooter: false});	property: showTodayButtonjavascriptvar datepicker = new ej.calendars.DatePicker({ showTodayButton: false});datepicker.appendTo('#element');
Show?Hide the other month dates	property: showOtherMonthsjavascript\$("#datepicker").ejDatePicker({ showOtherMonths: false});	javascriptvar datepicker = new ej.calendars.DatePicker();datepicker.appendTo('#element');.e-datepicker .e-calendar .e-content tr.e-month-hide,.e-datepicker .e-calendar .e-content td.e-other-month > .e-day { visibility: none;}.e-datepicker .e-calendar .e-content td.e-month-hide,.e-datepicker .e-calendar .e-content td.e-other-month { pointer-events: none; touch-action: none;}
Show/Hide the disabled range	property: showDisabledRangejavascript\$("#datepicker").ejDatePicker({ blackoutDates: [new Date(2016, 4, 10), new Date(2016, 4, 15), new Date(2016, 4, 20), new Date(2016, 4, 22), new Date(2016, 5, 12), new Date(2016, 5, 24)] showDisabledRange: false});	Not Applicable
Show/Hide the popup button	property: showPopupButtonjavascript\$("#datepicker").ejDatePicker({ showPopupButton: true});	Can be achieved byjavascriptvar datepicker = new ej.calendars.DatePicker({ focus: onFocus});datepicker.appendTo('#element');function onFocus(args) {

	<code>r").ejDatePicker({ showPopupButton: false});</code>	<code>datepicker.show();</code> ; <code>.e-control-wrapper .e-input-group-icon.e-date-icon { display: none;}</code>
Enable/Disable rounded corner	property: <code>showRoundedCorner</code> <code>javascript\$("#datepicker").ejDatePicker({ showRoundedCorner: true});</code>	Can be achieved by <code>javascript</code> <code>var datepicker = new ej.calendars.DatePicker({ cssClass: 'e-custom-style' });</code> <code>datepicker.appendTo('#element');</code> ; <code>.e-control-wrapper.e-custom-style.e-date-wrapper.e-input-group { border-radius: 4px;}</code>
Skip a month	property: <code>stepMonths</code> <code>javascript\$("#datepicker").ejDatePicker({ stepMonths: 2});</code>	Can be achieved by <code>javascript</code> <code>var datepicker = new ej.calendars.DatePicker({ value: new Date("09/04/2018"), open: onOpen});</code> <code>datepicker.appendTo('#element');</code> <code>function onOpen(args) {</code> <code> datepicker.navigateTo('Year', new Date("03/18/2018"));</code> <code>}</code>
Show/Hide the tooltip	property: <code>showTooltip</code> <code>javascript\$("#datepicker").ejDatePicker({ showTooltip: false});</code>	Not Applicable
Today button text	property: <code>buttonText</code> <code>javascript\$("#datepicker").ejDatePicker({ buttonText : "Now"});</code>	Can be achieved by <code>javascript</code> <code>L10n.load({ 'en': { 'datepicker': { today: 'Now' } } });</code> <code>var datepicker = new ej.calendars.DatePicker({ locale: 'en' });</code> <code>datepicker.appendTo('#element');</code>
Display inline	property: <code>displayInline</code> <code>javascript\$("#datepicker").ejDatePicker({ displayInline: true});</code>	Not Applicable
Enable/Disable the animation	property: <code>enableAnimation</code> <code>javascript\$("#datepicker").ejDatePicker({ enableAnimation : false});</code>	Not Applicable
Highlight dates	property: <code>highlightSection</code> <code>javascript\$("#datepicker").ejDatePicker({ highlightSection: "week"});</code>	Can be achieved by <code>javascript</code> <code>var datepicker = new ej.calendars.DatePicker({ renderDayCell: highlightDate});</code> <code>datepicker.appendTo('#element');</code> <code>function highlightDate(args) {</code> <code> if (args.date.getDate() === 10) {</code> <code> args.element.classList.add('e-</code>

		<code>highlightweekend'); }}.e-highlightweekend { background-color: #cfe9f3;}</code>
Highlight weekend	property: <code>highlightWeekendjavascript\$("#datepicker").ejDatePicker({ highlightWeekend : true});</code>	Can be achieved by <code>javascriptvar datepicker = new ej.calendars.DatePicker({ renderDayCell: highlightDate});datepicker.appendTo('#element');function highlightDate(args) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.element.classList.add('e-highlightweekend'); }}.e-highlightweekend { background-color: #cfe9f3;}</code>
Tooltip format	property: <code>tooltipFormatjavascript\$("#datepicker").ejDatePicker({ tooltipFormat : "dd/MM/yyyy"});</code>	Not Applicable
Special Dates	property: <code>specialDatesjavascriptspecialDays = [{ date: new Date(2018, 09, 08), tooltip: "In Australia", iconClass: "flags sprite-Australia" }, { date: new Date(2018, 09, 21), tooltip: "In France", iconClass: "flags sprite-France" }]\$("#datepicker").ejDatePicker({ specialDates: specialDays});</code>	Can be achieved by <code>javascriptvar datepicker = new ej.calendars.DatePicker({ renderDayCell: customDates, value: new Date('5/5/2017')});datepicker.appendTo('#element');function customDates(args) { if (args.date.getDate() === 10) { var span = document.createElement('span'); span.setAttribute('class', 'e-icons highlight'); args.element.firstElementChild.setAttribute('title', 'Birthday !'); args.element.setAttribute('title', 'Birthday !'); args.element.setAttribute('data-val', 'Birthday !'); args.element.appendChild(span); }}</code>
FocusIn event	Event: <code>focusInjavascript\$("#datepicker").ejDatePicker({ focusIn: function (args) { /code block/}});</code>	Event: focus <code>javascriptvar datepicker = new ej.calendars.DatePicker({ focus: onFocus});datepicker.appendTo('#element');function onFocus(args){ /code block/}</code>
FocusOut event	Event: <code>focusOutjavascript\$("#datepicker").ejDatePicker({ focusOut: function (args) { /code block/}});</code>	Event: blur <code>javascriptvar datepicker = new ej.calendars.DatePicker({ blur: onBlur});datepicker.appendTo('#element');function onBlur(args){ /code block/}</code>

FocusIn method	Not Applicable	Method: focusIn() javascriptvar datepicker = new ej.calendars.DatePicker({ placeholder: 'Enter date'});datepicker.appendTo('#element');datepicker.focusIn();
FocusOut method	Not Applicable	Method: focusOut() javascriptvar datepicker = new ej.calendars.DatePicker({ placeholder: 'Enter date'});datepicker.appendTo('#element');datepicker.focusOut();
Prevent popup close	Not Applicable	Event: close javascriptvar datepicker = new ej.calendars.DatePicker({ close: function (args) { args.preventDefault(); }});datepicker.appendTo('#element');datepicker.show();
Prevent popup open	Not Applicable	Event: open javascriptvar datepicker = new ej.calendars.DatePicker({ open: function (args) { args.preventDefault(); }});datepicker.appendTo('#element');

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the RTL	property: enableRTL javascript\$("#datepicker").ejDatePicker({ enableRTL : true});	property: enableRtl javascriptvar datepicker = new ej.calendars.DatePicker({ enableRtl : true});datepicker.appendTo('#element');

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable Persistence	property: enablePersistence javascript\$("#datepicker").ejDatePicker({enablePersistence : true});	property: enablePersistence javascriptvar datepicker = new ej.calendars.DatePicker({ enablePersistence :

		<code>true});datepicker.appendTo('#element');</code>
--	--	--

Validation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Validation rules	property: <code>validationRulesjavascript\$("#datepicker").ejDatePicker({ validationRules:{ required:true }});</code>	Can be achieved by <code>javascriptvar datepicker = new ej.calendars.DatePicker({ placeholder: 'Enter date'});datepicker.appendTo('#element');var options = { rules: { 'datevalue': { required: true } }}var formObject: FormValidator = new FormValidator('#form-element', options);</code>
Validation message	property: <code>validationMessagejavascript\$("#datepicker").ejDatePicker({ validationRules:{ required:true }, validationMessage:{ required: "Required Date value"}});</code>	Can be achieved by <code>javascriptvar datepicker = new ej.calendars.DatePicker({ placeholder: 'Enter date'});var options = { rules: { 'datevalue': { required: [true, 'Date value required'] } }, customPlacement: function (inputElement, errorElement) { inputElement.parentElement.parentElement.appendChild(errorElement); }}; var formObject = new ej.inputs.FormValidator('#form-element', options); datepicker.appendTo('#element');</code>

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	property: <code>widthjavascript\$("#datepicker").ejDatePicker({ width: 200});</code>	property: width <code>javascriptvar datepicker = new ej.calendars.DatePicker({ width: '200px'});datepicker.appendTo('#element');</code>
Readonly	property: <code>readOnlyjavascript\$("#datepicker").ejDatePicker({ readOnly : true});</code>	property: readonly <code>javascriptvar datepicker = new ej.calendars.DatePicker({ readonly:true, value:new Date()});datepicker.appendTo('#element');</code>
Show/Hide the Clear Button	Not Applicable	property: showClearButton <code>javascriptvar datepicker = new ej.calendars.DatePicker({ showClearButton: true});datepicker.appendTo('#element');</code>

Height	property: heightjavascript\$("#datepicker").ejDatePicker({ height: 35});	Can be achieved byjavascriptvar datepicker = new ej.calendars.DatePicker({cssClass: 'e-custom-style'});datepicker.appendTo('#element');e-control-wrapper.e-custom-style.e-date-wrapper.e-input-group{ height: 35px;}
Html Attributes	property: htmlAttributesjavascript\$("#datepicker").ejDatePicker({ htmlAttributes : {required:"required"}});	Not Applicable
Enable/Disable the Week Number	property: weekNumberjavascript\$("#datepicker").ejDatePicker({ weekNumber : true});	property: weekNumberjavascriptvar datepicker = new ej.calendars.DatePicker({ weekNumber:true});datepicker.appendTo('#element');
Watermark text	property: watermarkTextjavascript\$("#datepicker").ejDatePicker({ watermarkText: "Enter date"});	property: placeholderjavascriptvar datepicker = new ej.calendars.DatePicker({ placeholder: 'Enter date'});datepicker.appendTo('#element');
Disable/Enable	property: enabledjavascript\$("#datepicker").ejDatePicker({ enabled: false});	property: enabledjavascriptvar datepicker = new ej.calendars.DatePicker({enabled:false});datepicker.appendTo('#element');
Disable the DatePicker	Method: disable()javascript\$("#datepicker").ejDatePicker();var dateObj = \$("#datepicker").data("ejDatePicker");dateObj.disable();	Not Applicable
Enable the DatePicker	Method: enable()javascript\$("#datepicker").ejDatePicker();var dateObj = \$("#datepicker").data("ejDatePicker");dateObj.enable();	Not Applicable
Enable/Disable the textbox editing	property: allowEditjavascript\$("#datepicker").ejDatePicker({ allowEdit : true});	property: allowEditjavascriptvar datepicker = new ej.calendars.DatePicker({ allowEdit : true});datepicker.appendTo('#element');
zIndex	Not Applicable	property: zIndexjavascriptvar datepicker = new ej.calendars.DatePicker({ zIndex: 100});datepicker.appendTo('#element');
Specify the	Not Applicable	property: floatLabelTypejavascriptvar datepicker = new ej.calendars.DatePicker({ placeholder:

placeholder text behavior		'Enter date', floatLabelType: 'Auto'});datepicker.appendTo('#element');
---------------------------	--	---

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	property: localejavascript\$("#datepicker").ejDatePicker({locale: "en-US"});	property: localejavascriptvar datepicker = new ej.calendars.DatePicker({ locale: 'en'});datepicker.appendTo('#element');
First day of week	property: startDayjavascript\$("#datepicker").ejDatePicker({startDay: 2});	property: firstDayOfWeekjavascriptvar datepicker = new ej.calendars.DatePicker({ firstDayOfWeek: 2});datepicker.appendTo('#element');

Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	property: enableStrictModejavascript\$("#datepicker").ejDatePicker({ enableStrictMode : true});	property: strictModejavascriptvar datepicker = new ej.calendars.DatePicker({ strictMode: true, value: new Date('5/28/2017'), min: new Date('5/5/2017'), max: new Date('5/25/2017')});datepicker.appendTo('#element');

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Close	Event: closejavascript\$("#datepicker").ejDatePicker({ close: function (args) { /code block/}});	Event: closejavascriptvar datepicker = new ej.calendars.DatePicker({ close: function (args) { /code block/}});datepicker.appendTo('#element');
Hide	Method: hide()javascript\$("#datepicker").ejDatePicker();var dateObj = \$("#datepicker").data("ejDatePicker");dateObj.hide();	Method: hide()javascriptvar datepicker = new ej.calendars.DatePicker({ value: new Date("05/11/2017")});datepicker.appendTo('#element');datepicker.hide();
Open	Event: openjavascript\$("#datepicker").ejDatePicker({open: function (args) { /code block/}});	Event: openjavascriptvar datepicker = new ej.calendars.DatePicker({ open: function (args) { /code block/}});datepicker.appendTo('#element');
Show	Method: show()javascript\$("#datepicker").ejDatePicker();var dateObj = \$("#datepicker").data("ejDatePicker");dateObj.show();	Method: show()javascriptvar datepicker = new ej.calendars.DatePicker({ value: new Date("05/11/2017")});datepicker.appendTo('#element');datepicker.show();

View Navigation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Navigate to specific month	Not Applicable	Method: navigateTo()javascriptvar datepicker = new ej.calendars.DatePicker({ value: new Date("09/04/2018"), open:onOpen});datepicker.appendTo('#element');function onOpen(args) { datepicker.navigateTo('Year', new Date("03/18/2018"));
Navigation callback	Event: navigatejavascript\$("#datepicker").ejDatePicker({ navigate: function (args) { /code block/}});	Event: navigatedjavascriptvar datepicker = new ej.calendars.DatePicker({navigated: onNavigate});datepicker.appendTo('#element');function onNavigate(args) { /code block/}
Enable/Disable the drill down	property: allowDrillDownjavascript\$("#datepicker").ejDatePicker({ allowDrillDown : true});	Not Applicable

DateRangePicker

Range restriction in EJ2 JavaScript 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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
let today: Date = new Date();
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
// creates a daterangepicker with min max property
let daterangeObject: DateRangePicker = new DateRangePicker({
    //sets the min.
    min: new Date(currentYear, currentMonth, 15),
    //sets the max.
    max: new Date(currentYear, currentMonth+1, 15),
    placeholder:"Select a Range"
});
daterangeObject.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.TS

```

import { DateRangePicker } from '@syncfusion/ej2-calendars';
let today: Date = new Date();
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
// creates a daterangepicker with min max property
let daterangeObject: DateRangePicker = new DateRangePicker({
    //sets the minimum number of days
    minDays: 4,
    //sets the maximum number of days
    maxDays: 7,
    placeholder:"Select a Range"
});

```

```
daterangeObject.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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates a daterangepicker in strict mode
let daterangeObject: DateRangePicker = new DateRangePicker({
  //enabling strict mode
```

```

        strictMode: true, placeholder:"Select a Range"
    });
    daterangeObject.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 EJ2 JavaScript 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.

`

```
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-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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
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',
        customRange: 'benutzerdefinierten Bereich'
    }
});
// creates the simple DateRangePicker component with de culture
let daterangeObject: DateRangePicker = new DateRangePicker({
    // sets the locale property.
    locale: 'de',
    value: new Date(),
});
daterangeObject.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">
    </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.TS

```

import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateRangePicker } from '@syncfusion/ej2-calendars';
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as hetimeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, hetimeZoneNames);
L10n.load({
  'he': {
    'daterangepicker': {
      placeholder: 'בחר טווח',
      startLabel: 'תווית התחלה',
      endLabel: 'סוף',
      applyText: 'להחיל טקסט',
      cancelText: 'בטל טקסט',
      selectedDays: 'ימים נבחרים',
      days: 'ימים',
      customRange: 'טווח מותאם אישית'
    }
  }
});
// creates the simple DateRangePicker component
let daterangeobject: DateRangePicker = new DateRangePicker({
  //sets the locale
  locale: 'he',
  //sets the enableRtl
  enableRtl: true,
});
daterangeobject.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">
    </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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates a daterangepicker with format property
let daterangeObject: DateRangePicker = new DateRangePicker({
    format: "yyyy-MM-dd",
    separator: "to",
    placeholder: "Select Range"
});
daterangeObject.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 EJ2 JavaScript 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.TS

```

import { DateRangePicker, RenderDayCellEventArgs } from '@syncfusion/ej2-
calendars';
// creates daterangepicker with placeholder.
let daterangeObject: DateRangePicker = new DateRangePicker({
    // Bind the renderDayCell event to customize the each day cell.
    renderDayCell: onRenderCell,
    // sets the placeholder

```

```

        placeholder: 'Select a range'
    });
    daterangeObject.appendTo('#element');
    function onRenderCell(args: RenderDayCellEventArgs): void {
        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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
let today: Date = new Date();
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
// creates a daterangepicker with predefined ranges (label, start date, end date)
let daterangeObject: DateRangePicker = new 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() },
    ]
});
daterangeObject.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.TS

```

import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates daterangepicker with readonly.
let daterangeObject: DateRangePicker = new DateRangePicker({
  // sets the firstDayOfWeek to Monday.
  firstDayOfWeek:1, placeholder:"Select Range"
});
daterangeObject.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 EJ2 JavaScript 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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates a DateRangePicker component
let daterangeObject: DateRangePicker = new DateRangePicker({
    // sets the placeholder
    placeholder: 'Select a range'
});
daterangeObject.appendTo('#element');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the component.
        daterangeObject.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 EJ2 JavaScript 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

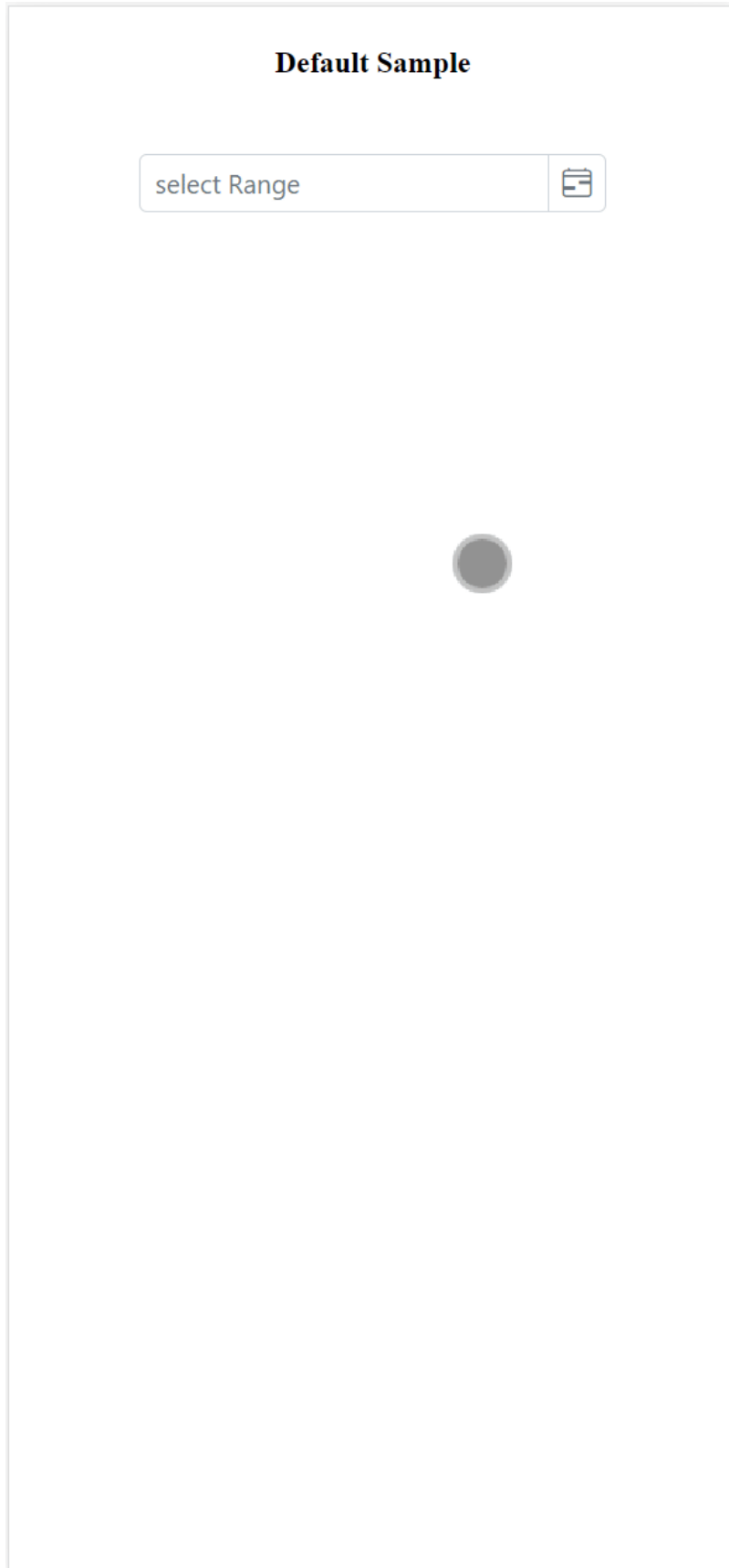
/ To specify background and border /

```
.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.

```
`typescript  
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');
```



![DateRangePickerPresetsFullScreen](../images/DateRangePickerPresetsFullScreen.gif)

How To

Disable the `daterangepicker` component in EJ2 JavaScript `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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';  
// creates daterangepicker with enabled property  
let daterangeObject: DateRangePicker = new DateRangePicker({  
    // sets the enabled as false  
    enabled: false, placeholder: "Select Range"  
});  
daterangeObject.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 EJ2 JavaScript 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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
// creates DateRangePicker with placeholder.
let daterangeObject: DateRangePicker = new DateRangePicker({
    // sets the placeholder property.
    placeholder: 'Select a range'
});
daterangeObject.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 EJ2 JavaScript 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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';  
let daterangeObject: DateRangePicker = new DateRangePicker({
```

```
// set the css class.
cssClass:"customCSS", placeholder:"Select Range"
});
daterangeObject.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 EJ2 JavaScript 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.TS

```
import { DateRangePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// creates daterangepicker component
let daterangepickerObject: DateRangePicker = new DateRangePicker({
    dayHeaderFormat: "Short",
    cssClass: "format-wide"
});
daterangepickerObject.appendTo('#element');
let formatLabel: DropDownList = new DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: (args: any) => {
        daterangepickerObject.dayHeaderFormat = args.value;
    }
});
formatLabel.appendTo('#select');
```

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-
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>
</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 EJ2 JavaScript Daterangepicker control

This article describes the API migration process of DateRangePicker component from Essential JS 1 to Essential JS 2.

Date Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	property: value valuejavascript\$("#daterangepicker").ejDateRangePicker({value: "5/5/2014 - 6/6/2018"});	property: valuejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ value : [new Date("5/5/2019"), new Date("6/6/2019")]});daterangepicker.appendTo('#element');

Date Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Display the date format	<pre> property: dateFormatjavascript\$("#daterangepicker").ejDateRangePicker({dateFormat: "dd/MM/yyyy"}); </pre>	<pre> property: formatjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ format: 'dd\'\'\'MMM\'\'\'\'yy hh:mm a', startDate: new Date("11/9/2017"), endDate: new Date("11/21/2017"))};daterangepicker. appendTo('#element'); </pre>
-------------------------	--	---

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum date	<pre> property: minDatejavascript\$("#daterangepicker").ejDateRangePicker({minDate: new Date("1/1/2017")}); </pre>	<pre> property: minjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({min: new Date("1/1/2017")});daterangepicker.appendTo('# element'); </pre>
Maximum date	<pre> property: maxDatejavascript\$("#daterangepicker").ejDateRangePicker({maxDate: new Date("1/1/2017")}); </pre>	<pre> property: maxjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({max: new Date("1/1/2017")});daterangepicker.appendTo('# element'); </pre>
Start date	<pre> property: startDatejavascript\$("#daterangepicker").ejDateRangePicker({startDate : new Date("5/30/2015")}); </pre>	<pre> property: startDatejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({startDate: new Date("11/9/2017")});daterangepicker.appendTo('# element'); </pre>
End date	<pre> property: endDatejavascript\$("#daterangepicker").ejDateRangePicker({endDate: new Date("5/1/2013")}); </pre>	<pre> property: endDatejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({endDate: new Date("11/21/2017")});daterangepicker.appendTo('#element'); </pre>
Presets ranges	<pre> property: rangesjavascript\$("#daterangepicker").ejDateRangePicker({ ranges: [{ label: "Today", range: [new Date(), new Date()] }, { label: "Last 1 Week", range: [new Date(new Date().setDate(new Date().getDate() - 7)), new Date()] }, { label: "Last 1 Month", range: [new Date(new Date().setMonth(new Date().getMonth() - 1)), new Date()] }, { </pre>	<pre> property: presetsjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ 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() - </pre>

	label: "Last 2 Month", range: [new Date(new Date().setMonth(new Date().getMonth() - 2)), new Date()],],});	1, 0, 1), end: new Date(),]);daterangepicker.appendTo('#element');
Add ranges	Method: addRanges()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.addRanges("new Range", [new Date("11/12/2019"),new Date("11/12/2021")]);	Not Applicable
Clear ranges	Method: clearRanges()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.clearRanges();	Not Applicable
Get selected range	Method: getSelectedRange()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.getSelectedRange();	Method: getSelectedRange()javascriptvar daterangepicker = new ej.calendars.DateRangePicker({ startDate: new Date("11/9/2017"), endDate: new Date("11/21/2017")});daterangepicker.appendTo('#element');daterangepicker.getSelectedRange();
Set date range	Method: setRange()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.setRange([new Date("11/12/2011"), new Date("11/12/2019")]);	Not Applicable

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disabled dates	Not Applicable	Can be achieved by:javascriptvar daterangepicker = new ej.calendars.DateRangePicker({renderDayCell: onRenderCell});daterangepicker.appendTo('#element');function

		onRenderCell(args){ if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; }}
--	--	--

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	property: cssClassjavascript\$("#daterangepicker").ejDateRangePicker({cssClass: "gradient-lime"});	property: cssClassjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({cssClass: "customCSS"});daterangepicker.appendTo('#element');
Enable/Disable DateRangePicker with TimePicker	property: enableTimePickerjavascript\$("#daterangepicker").ejDateRangePicker({enableTimePicker: true});	Not Applicable
Time format	property: timeFormatjavascript\$("#daterangepicker").ejDateRangePicker({timeFormat: "HH:mm"});	Not Applicable
Minimum days span	Not Applicable	property: minDaysjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({minDays: 5});daterangepicker.appendTo('#element');
Maximum days span	Not Applicable	property: maxDaysjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({maxDays: 10});daterangepicker.appendTo('#element');
Button text	property: buttonTextjavascript\$("#daterangepicker").ejDateRangePicker({ buttonText : {reset: "Reset", cancel: "Cancel", apply: "Apply"}});	Can be achieved by:javascriptL10n.load({'en': {'daterangepicker': { applyText: 'Apply' } }});var daterangepicker = new ej.calendars.DateRangePicker({ locale: 'en'});daterangepicker.appendTo('#element');

Show/hide the popup button	property: showPopupButton javascript\$("#daterangepicker").ejDateRangePicker({showPopupButton: false});	Can be achieved by:javascriptvar daterangepicker = new ej.calendars.DateRangePicker({focus: onFocus});daterangepicker.appendTo('#element');function onFocus(args) { daterangepicker.show();}e-control-wrapper .e-input-group-icon.e-range-icon {display: none;}
Enable/Disable the rounded corner	property: showRoundedCorner javascript\$("#daterangepicker").ejDateRangePicker({showRoundedCorner: true});	Can be achieved by:javascriptvar daterangepicker = new ej.calendars.DateRangePicker({cssClass: 'e-custom-style'});daterangepicker.appendTo('#element');e-control-wrapper.e-custom-style.e-date-range-wrapper.e-input-group {border-radius: 4px;}
FocusIn event	Not Applicable	Event: focusjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ focus: onFocus});daterangepicker.appendTo('#element');function onFocus(args) {/ code block /}
FocusOut event	Not Applicable	Event: blurjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({blur: onBlur});daterangepicker.appendTo('#element');function onBlur(args) {/ code block /}
FocusIn method	Not Applicable	Method: focusIn()javascriptvar daterangepicker = new ej.calendars.DateRangePicker({ value: new Date()});daterangepicker.appendTo('#element');daterangepicker.focusIn();
FocusOut method	Not Applicable	Method: focusOut()javascriptvar daterangepicker = new ej.calendars.DateRangePicker({value: new Date()});daterangepicker.appendTo('#element');daterangepicker.focusOut();

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Enable/Disable the RTL	Not Applicable	property: enableRtljavascriptvar daterangepicker = new ej.calendars.DateRangePicker({enableRtl: true});daterangepicker.appendTo('#element');
------------------------	----------------	--

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the Persistence	property: enablePersistencejavascript\$("#daterangepicker").ejDateRangePicker({enablePersistence: true});	property: enablePersistencejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({enablePersistence: true, startDate: new Date("11/9/2017"), endDate: new Date("11/21/2017")});daterangepicker.appendTo('#element');

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	property: widthjavascript\$("#daterangepicker").ejDateRangePicker({width: 200});	property: widthjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({width: 200});daterangepicker.appendTo('#element');
Readonly	Not Applicable	property: readonlyjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({readonly: true});daterangepicker.appendTo('#element');
Show/Hide the clear button	Not Applicable	property: showClearButtonjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({showClearButton: false});daterangepicker.appendTo('#element');

Height	property: heightjavascript\$("#daterangepicker").ejDateRangePicker({height: 35});	Can be achieved by:javascriptvar daterangepicker = new ej.calendars.DateRangePicker({cssClass:"e-custom-style"});daterangepicker.appendTo('#element');.e-control-wrapper.e-custom-style.e-date-range-wrapper.e-input-group {height: 35px;}
Enable/Disable the week number	Not Applicable	property: weekNumberjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({weekNumber: true});daterangepicker.appendTo('#element');
Watermark text	property: watermarkTextjavascript\$("#daterangepicker").ejDateRangePicker({watermarkText: "Enter dateRange"});	property: placeholderjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({placeholder: 'Enter dateRange'});daterangepicker.appendTo('#element');
Enable/Disable	property: enabledjavascript\$("#daterangepicker").ejDateRangePicker({enabled: false});	property: enabledjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({enabled: false});daterangepicker.appendTo('#element');
Disable the DateRangePicker	Method: disable()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.disable();	Not Applicable
Enable the DateRangePicker	Method: enable()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.enable();	Not Applicable
Enable/Disable the textbox editing	property: allowEditjavascript\$("#daterangepicker").ejDateRangePicker({allowEdit: false});	property: allowEditjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({allowEdit: false});daterangepicker.appendTo('#element');

Specify the placeholder text behavior	Not Applicable	property: floatLabelTypejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ placeholder: 'Enter date', floatLabelType: 'Auto'});daterangepicker.appendTo('#element');
zIndex	Not Applicable	property: zIndexjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({zIndex: 100});daterangepicker.appendTo('#element');
Specify the date range separator	property: separatorjavascript\$("#daterangepicker"). ejDateRangePicker({separator : "\$"});	property: separatorjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({separator : "\$"});daterangepicker.appendTo('#element');

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	property: localejavascript\$("#daterangepicker").ejDateRangePicker({locale: "en-US"});	property: localejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({locale: 'en'});daterangepicker.appendTo('#element');
First day of week	Not Applicable	property: firstDayOfWeekjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({firstDayOfWeek: 2});daterangepicker.appendTo('#element');

Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Strict mode	Not Applicable	property: strictModejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({strictMode: true});daterangepicker.appendTo('#element');
-------------	----------------	--

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	Event: closejavascript\$("#daterangepicker").ejDateRangePicker({ close: function (args) {/ code block /}});	Event: closejavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ close: function (args) {/ code block /}});daterangepicker.appendTo('#element');
Hide	Method: popupHide()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.popupHide();	Method: hide()javascriptvar daterangepicker = new ej.calendars.DateRangePicker({value: new Date()});daterangepicker.appendTo('#element');daterangepicker.hide();
Open	Event: openjavascript\$("#daterangepicker").ejDateRangePicker({open: function (args) {/ code block /}});	Event: openjavascriptvar daterangepicker = new ej.calendars.DateRangePicker({ open: function (args) {/ code block /}});daterangepicker.appendTo('#element');
Show	Method: popupShow()javascript\$("#daterangepicker").ejDateRangePicker();var dateObj = \$("#daterangepicker").data("ejDateRangePicker");dateObj.popupShow();	Method: show()javascriptvar daterangepicker = new ej.calendars.DateRangePicker({value: new Date()});daterangepicker.appendTo('#element');daterangepicker.show();

DateTimePicker

Globalization in EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
import { enableRipple } from '@syncfusion/ej2-base';
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
L10n.load({
  'de': {
    'datetimepicker': {
      placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
      today: 'heute'
    }
  }
});
```

```

    }
  }
});
// creates the simple DateTimePicker component with de culture
let datetimeObject: DateTimePicker = new DateTimePicker({
  // sets the locale property.
  locale: 'de',
  value: new Date("12/11/2017 1:00 AM"),
});
datetimeObject.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 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
//Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
//load the CLDR data files.
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import * as detimeZoneNames from './timeZoneNames.json';
import { enableRipple } from '@syncfusion/ej2-base';
loadCldr(numberingSystems, gregorian, numbers, detimeZoneNames);
L10n.load({
    'ar': {
        'datetimepicker': {
            placeholder: 'حدد التاريخ والوقت',
            today: 'اليوم'
        }
    }
});
// creates the simple DateTimePicker component
let datetimeobject: DateTimePicker = new DateTimePicker({
    //sets the locale
    locale: 'ar',
    //sets the enableRtl
    enableRtl: true,
});
datetimeobject.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="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.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 EJ2 JavaScript 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.TS

```

import { DateTimePicker } from '@syncfusion/ej2-calendars';
// creates a datetimepicker with strictMode property
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    // sets the strictMode
    strictMode: true,
    // sets the value
    value: new Date('5/28/2019 2:00 AM'),
    //sets the min
    min: new Date('5/5/2019 2:00 AM'),
    //sets the max
    max: new Date('5/25/2019 2:00 AM')
});
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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
// creates a datetimepicker with strictMode property
let datetimepickerObject: DateTimePicker = new 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 EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
let today: Date = new Date();
let currentYear: number = today.getFullYear();
let currentMonth: number = today.getMonth();
let currentDay: number = today.getDate();
let currentHour: number = today.getHours();
let currentMinute: number = today.getMinutes();
let currentSecond: number = today.getSeconds();
// creates a datetimepicker with min max property
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    //sets the min.
    min: new Date(currentYear, currentMonth, 7, 0, 0, 0),
    //sets the max.
    max: new Date(currentYear, currentMonth,
27, currentHour, currentMinute, currentSecond),
    //sets the value.
    value: new Date(new Date().setDate(14))
});
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 format in EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';  
// creates a DatePicker component with custom format.  
let datetimepickerObject: DateTimePicker = new DateTimePicker({  
    value: new Date(),  
    // sets the format.  
    format: 'yyyy-MM-dd hh:mm'  
});  
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>
```

[Date time masking in EJ2 JavaScript 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.TS

```
import { DateTimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DateTimePicker.Inject(MaskedDateTime);
let datetimepickerObject: DateTimePicker = new 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.TS

```
import { DateTimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DateTimePicker.Inject(MaskedDateTime);
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    enableMask: true
});
datetimepickerObject.appendTo('#mask');
let datetimepickerFormat: DateTimePicker = new 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.TS

```
import { DateTimePicker, MaskedDateTime } from '@syncfusion/ej2-calendars';
DateTimePicker.Inject(MaskedDateTime);
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    enableMask: true
});
datetimepickerObject.appendTo('#mask');
let datetimeplaceholder: DateTimePicker = new 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 EJ2 JavaScript 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.TS

```
import { DateTimePicker, RenderDayCellEventArgs, ItemEventArgs } from
 '@syncfusion/ej2-calendars';
// creates datetimepicker with placeholder.
let datetimeObject: DateTimePicker = new DateTimePicker({
    // Bind the renderDayCell event to customize the each day cell.
    renderDayCell: onRenderCell,
    // sets the placeholder
    placeholder: 'Select a date and time'
});
datetimeObject.appendTo('#element');
function onRenderCell(args: RenderDayCellEventArgs): void {
    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 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.TS

```

import { DateTimePicker } from '@syncfusion/ej2-calendars';
// creates a simple datetimepicker component
let datetimepicker: DateTimePicker = new DateTimePicker({
    //sets the place holder
    placeholder:"Select DateTime"
    floatLabelType: 'Auto'
});
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 EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
// creates a DateTimePicker component
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    // sets the placeholder
    placeholder: 'Select a date and time'
});
datetimepickerObject.appendTo('#element');
document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the component.
        datetimepickerObject.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 EJ2 JavaScript 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

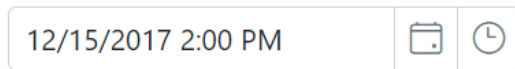
Please check the below section, to customize the style and appearance of the Calendar component in the DateTimePicker

[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.

```
`typescript  
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

How To

Disable the datetimepicker component in EJ2 JavaScript Datetimepicker control

To disable the DateTimePicker, use its [enable](#) property to `false`.

INDEX.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';  
// creates datetimepicker with enabled property  
let datetimeObject: DateTimePicker = new DateTimePicker({  
    // sets the enabled as false  
    enabled: false, placeholder: "Select a date and time"  
});  
datetimeObject.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 EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';  
// creates DateTimePicker with placeholder.  
let datetimeObject: DateTimePicker = new DateTimePicker({  
    // sets the placeholder property.  
    placeholder: 'Select a date and time'  
});  
datetimeObject.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 EJ2 JavaScript 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.TS

```
import { DateTimePicker } from '@syncfusion/ej2-calendars';
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// creates datetimepicker component
let datetimepickerObject: DateTimePicker = new DateTimePicker({
    dayHeaderFormat: "Short"
});
datetimepickerObject.appendTo('#element');
let formatLabel: DropDownList = new DropDownList({
    // set the height of the popup element
    popupHeight: '200px',
    // bind the change event
    change: (args: any) => {
        datetimepickerObject.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>
</body>
<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 EJ2 JavaScript Datetimepicker control

This article describes the API migration process of DateTimePicker component from Essential JS 1 to Essential JS 2.

DateTime Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	property: valuejavascript\$("#datetimepicker").ejDateTimePicker({value: new Date("5/5/2014 12:00 PM")});	property: valuejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("05/11/2017 12:00 PM")});datetimepicker.appendTo("#element");

DateTime Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display the datetime format	property: <code>dateTimeFormatjavascript\$("#datetimepicker").ejDateTimePicker({ value: new Date("05/11/2017 12:00 PM"), dateTimeFormat: "dd/MM/yyyy hh:mm tt"});</code>	property: <code>formatjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("05/11/2017 12:00 PM"), format: 'dd/MM/yyyy hh:mm a'});datetimepicker.appendTo('#element');</code>
Day header format	property: <code>dayHeaderFormatjavascript\$("#datetimepicker").ejDateTimePicker({ dayHeaderFormat: "short"});</code>	Not Applicable

Calendar Views

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Start view	property: <code>startLeveljavascript\$("#datetimepicker").ejDateTimePicker({ startLevel: ej.DateTimePicker.Level.Year});</code>	property: <code>startjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ start:'Decade'});datetimepicker.appendTo('#element');</code>
Depth view	property: <code>depthLeveljavascript\$("#datetimepicker").ejDateTimePicker({ depthLevel: ej.DateTimePicker.Level.Year});</code>	property: <code>depthjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ depth:'Year'});datetimepicker.appendTo('#element');</code>

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum datetime	property: <code>minDateTimejavascript\$("#datetimepicker").ejDateTimePicker({ minDateTime: new Date("5/1/2013 10:00 AM")});</code>	property: <code>minjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ min: new Date("5/1/2013 10:00</code>

		AM"))});datetimepicker.appendTo('#element');
Maximum datetime	property: maxDateTimejavascript\$("#datetimepicker").ejDateTimePicker({ maxDateTime : new Date("5/30/2015 10:00 PM"))});	property: maxjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ max : new Date("5/30/2015 10:00 PM"))});datetimepicker.appendTo('#element');

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disabled dates	property: disabledDateRangesjavascript\$("#datetimepicker").ejDateTimePicker({ disabledDateRanges: [{ startDateTime: new Date("11/30/2018 11:59 PM"), endDateTime:new Date("12/1/2018 11:59 PM")}]});	Can be achieved by:javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ renderDayCell: disableDate});datetimepicker.appendTo('#element');function disableDate(args) { if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; }}

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	property: cssClassjavascript\$("#datetimepicker").ejDateTimePicker({ cssClass: "gradient-lime"});	property: cssClassjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ cssClass: 'e-custom- style'});datetimepicker.appendTo('#element');
Show/Hide the button	Can be achieved by:javascript\$("#datetimepicker").ejDateTimePicker({ cssClass: "e-custom-class").e-datetime-popup.e-popup.e-custom-class.e-button-container { display: none !important;}	property: showTodayButtonjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ showTodayButton: false});datetimepicker.appendTo('#element');

Show/Hide the other month dates	property: showOtherMonths: false});	Can be achieved by: <pre>javascriptvar datetimepicker = new ej.calendars.DateTimePicker();datetimepic ker.appendTo('#element');.e- DateTimePicker .e-calendar .e-content tr.e-month-hide,.e-DateTimePicker .e- calendar .e-content td.e-other-month > .e- day { visibility: none;}.e-DateTimePicker .e- calendar .e-content td.e-month-hide,.e- DateTimePicker .e-calendar .e-content td.e-other-month { pointer-events: none; touch-action: none;}}</pre>
Show/Hide the popup button	property: showPopupButton: false});	Can be achieved by: <pre>javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ focus: onFocus});datetimepicker.appendTo('#ele ment');function onFocus(args) { datetimepicker.show();.e-control-wrapper .e-input-group-icon.e-date-icon { display: none; }</pre>
Enable/Disable the rounded corner	property: showRoundedCorner: true});	Can be achieved by: <pre>javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ cssClass: 'e- custom- style'});datetimepicker.appendTo('#eleme nt');.e-control-wrapper.e-custom-style.e- date-wrapper.e-input-group { border- radius: 4px;}</pre>
Skip a month	property: stepMonths: 2});	Can be achieved by: <pre>javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("09/04/2018"), open: onOpen});datetimepicker.appendTo('#element');function onOpen(args) { datetimepicker.navigateTo('Year', new Date("03/18/2018"));</pre>
Show/Hide the tooltip	property: showTooltip: false});	Not Applicable

Interval	property: intervaljavascript\$("#datetimepicker").ejDateTimePicker({ interval: 60});	property: stepjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ step: 60});datetimepicker.appendTo('#element');
Button text	property: buttonTextjavascript\$("#datetimepicker").ejDateTimePicker({ buttonText : { today: "Today", timeNow: "Time Now", done: "Done", timeTitle: "Time" }});	Can be achieved by:javascriptL10n.load({'en': { 'DateTimePicker': {today:'Now' } }});var datetimepicker = new ej.calendars.DateTimePicker({ locale: 'en'});datetimepicker.appendTo('#element');
Enable/Disable the animation	property: enableAnimationjavascript\$("#datetimepicker").ejDateTimePicker({ enableAnimation : false});	Not Applicable
FocusIn method	Not Applicable	Method: focusIn()javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter date'});datetimepicker.appendTo('#element');datetimepicker.focusIn();
FocusOut method	Not Applicable	Method: focusOut()javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter date'});datetimepicker.appendTo('#element');datetimepicker.focusOut();
Prevent popup close	Not Applicable	Event: closejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ close: function (args) { args.preventDefault(); }});datetimepicker.appendTo('#element');datetimepicker.show();
Prevent popup open	Not Applicable	Event: openjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ open: function (args) { args.preventDefault(); }});datetimepicker.appendTo('#element');datetimepicker.show();

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable RTL	<pre>property: enableRTLjavascript\$("#datetimepicker").ejDateTimePicker({ enableRTL : true});</pre>	<pre>property: enableRtljavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ enableRtl : true});datetimepicker.appendTo('#element');</pre>

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable Persistence	<pre>property: enablePersistencejavascript\$("#datetimepicker").ejDateTimePicker({ enablePersistence : true});</pre>	<pre>property: enablePersistencejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ enablePersistence : true});datetimepicker.appendTo('#element');</pre>

Validation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Validation rules	<pre>property: validationRulesjavascript\$("#datetimepicker").ejDateTimePicker({ validationRules:{ required : true }});</pre>	<pre>Can be achieved by:javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter date'});datetimepicker.appendTo('#element'); var options = { rules: { 'datetime': { required: true }} }var formObject: FormValidator = new FormValidator('#form-element', options);</pre>
Validation message	<pre>property: validationMessagejavascript\$("#datetimepicker").ejDateTimePicker({ validationRules:{required:true},validation Message:{ required: "Required Date value" }});</pre>	<pre>Can be achieved by:javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter date'});datetimepicker.appendTo('#element'); var options = { rules: { 'datetime': { required:</pre>

		<pre>[true, 'DateTime value required'] } }, customPlacement: function (inputElement, errorElement) { inputElement.parentElement.parentElement. appendChild(errorElement); }); var formObject = new ej.inputs.FormValidator('#form-element', options);</pre>
--	--	--

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	<pre>property: widthjavascript\$("#datetimepicker").ejD ateTimePicker({width: 200});</pre>	<pre>property: widthjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ width: '200px'});datetimepicker.appendTo('#elemen t');</pre>
Readonly	<pre>property: readOnlyjavascript\$("#datetimepicker"). ejDateTimePicker({ readOnly : true});</pre>	<pre>property: readOnlyjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ readOnly:true, value:new Date() });datetimepicker.appendTo('#element');</pre>
Show/Hide the clear button	Not Applicable	<pre>property: showClearButtonjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ showClearButton: true});datetimepicker.appendTo('#element');</pre>
Height	<pre>property: heightjavascript\$("#datetimepicker").ejD ateTimePicker({ height: 35});</pre>	<pre>Can be achieved by:javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ cssClass: 'e- custom- style'});datetimepicker.appendTo('#element') ;e-control-wrapper.e-custom-style.e-date- wrapper.e-input-group{ height: 35px;}</pre>
Html Attributes	<pre>property: htmlAttributesjavascript\$("#datetimepick er").ejDateTimePicker({ htmlAttributes : {required:"required"}});</pre>	Not Applicable
Enable/Disable	<pre>property: weekNumberjavascript\$("#datetimepicke</pre>	<pre>property: weekNumberjavascriptvar datetimepicker = new</pre>

the Week Number	<code>r").ejDateTimePicker({ weekNumber : true});</code>	<code>ej.calendars.DateTimePicker({ weekNumber:true});datetimepicker.appendTo('#element');</code>
Watermark text	property: <code>watermarkTextjavascript\$("#datetimepicker").ejDateTimePicker({ watermarkText: "Enter dateTime"});</code>	property: placeholderjavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter dateTime'});datetimepicker.appendTo('#element');</code>
Disable/Enable	property: <code>enabledjavascript\$("#datetimepicker").ejDateTimePicker({ enabled: false});</code>	property: enabledjavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ enabled:false});datetimepicker.appendTo('#element');</code>
Enable/Disable the textbox editing	property: <code>allowEditjavascript\$("#datetimepicker").ejDateTimePicker({ allowEdit : true});</code>	property: allowEditjavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ allowEdit : true});datetimepicker.appendTo('#element');</code>
zIndex	Can be achieved by: <code>javascript\$("#datetimepicker").ejDateTimePicker({ cssClass: "e-custom-class"}).e-datetime-popup.e-popup.e-custom-class { z-index: 100 !important;}</code>	property: zIndexjavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ zIndex: 100});datetimepicker.appendTo('#element');</code>
Specify the placeholder text behavior	Not Applicable	property: floatLabelTypejavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ placeholder: 'Enter date', floatLabelType: 'Auto'});datetimepicker.appendTo('#element');</code>
Event callback for each cell creation	Not Applicable	Event: renderDayCelljavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ renderDayCell: onRenderCell});datetimepicker.appendTo('#element');</code> <code>function onRenderCell(args){/code block/}</code>
FocusIn event	Event: <code>focusInjavascript\$("#datetimepicker").ejDateTimePicker({focusIn: function (args) {/code block/}});</code>	Event: focusjavascriptvar <code>datetimepicker = new ej.calendars.DateTimePicker({ focus: onFocus});datetimepicker.appendTo('#element');</code> <code>function onFocus(args){/code block/}</code>

FocusOut event	Event: focusOutjavascript\$("#datetimepicker").ejDateTimePicker({ focusOut: function (args) {/code block/}});	Event: blurjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ blur: onBlur});datetimepicker.appendTo('#element');function onBlur(args){/code block/}
Change event	Event: changejavascript\$("#datetimepicker").ejDateTimePicker({change: function (args) {/code block/}});	Event: changejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({change: onChange});datetimepicker.appendTo('#element');onChange(args){/code block/}
Created event	Event: createjavascript\$("#datetimepicker").ejDateTimePicker({create: function (args) {/code block/}});	Event: createdjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ created: onCreate});datetimepicker.appendTo('#element');function onCreate(args) {/code block/}
Destroy event	Event: destroyjavascript\$("#datetimepicker").ejDateTimePicker({ destroy: function (args) {/code block/}});	Event: destroyedjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ destroyed: onDestroy});datetimepicker.appendTo('#element');function onDestroy(args){/code block/}

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	property: localejavascript\$("#datetimepicker").ejDateTimePicker({ locale: "en-US"});	property: localejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ locale: 'en'});datetimepicker.appendTo('#element');
Specify the first day of week	property: startDayjavascript\$("#datetimepicker").ejDateTimePicker({ startDay: 2});	property: firstDayOfWeekjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ firstDayOfWeek: 2});datetimepicker.appendTo('#element');

Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	<pre>property: enableStrictModejavascript\$("#datetimepicker").ejDateTimePicker({ enableStrictMode : true});</pre>	<pre>property: strictModejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ strictMode: true, value: new Date('5/28/2017'), min: new Date('5/5/2017'), max: new Date('5/25/2017')});datetimepicker.append To('#element');</pre>

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	<pre>Event: closejavascript\$("#datetimepicker").ej DateTimePicker({close: function (args) {/code block/}});</pre>	<pre>Event: closejavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ close: function (args) {/code block/}});datetimepicker.appendTo('#element');</pre>
Hide	<pre>Method: hide()javascript\$("#datetimepicker").ej DateTimePicker();var dateObj = \$("#datetimepicker").data("ejDateTim ePicker");dateObj.hide();</pre>	<pre>Method: hide()javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("05/11/2017")});datetimepicker.appendTo('# element');datetimepicker.hide();</pre>
Open	<pre>Event: openjavascript\$("#datetimepicker").ej DateTimePicker({open: function (args) {/code block/}});</pre>	<pre>Event: openjavascriptvar datetimepicker = new ej.calendars.DateTimePicker({ open: function (args) {/code block/}});datetimepicker.appendTo('#element');</pre>
Show	<pre>Method: show()javascript\$("#datetimepicker"). ejDateTimePicker();var dateObj = \$("#datetimepicker").data("ejDateTim ePicker");dateObj.show();</pre>	<pre>Method: show()javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("05/11/2017")});datetimepicker.appendTo('# element');datetimepicker.show();</pre>

View Navigation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Navigate to specific month	Not Applicable	Method: navigateTo() javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ value: new Date("09/04/2018"), open: onOpen}); datetimepicker.appendTo ('#element'); function onOpen(args) { datetimepicker.navigateTo('Year', new Date("03/18/2018")); }
Navigation callback	Not Applicable	Event: navigated javascriptvar datetimepicker = new ej.calendars.DateTimePicker({ navigated: onNavigate}); datetimepicker.appendTo('# element'); function onNavigate(args) { }
Enable/Disable the drill down	property: timeDrillDown javascript\$(" #datetimepicker ").ejDateTimePicker({ timeDrillDown: { enabled: true, showMeridian: true, autoClose: true, interval: 10 }});	Not Applicable

Diagram

Getting started in EJ2 JavaScript 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.

```
`javascript
|-- @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](https://github.com/syncfusion/ej2-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.

```
`javascript
```

```
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.TS

```
import { Diagram } from '@syncfusion/ej2-diagrams';
let diagram: Diagram = new 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.

```
`javascript
```

```
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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  // Unique name for the node
  name: "Start",
  // Position of the node
  offsetX: 300,
  offsetY: 50,
  // Size of the node
  width: 140,
  height: 50,
  // Text(label) added to the node
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  // Shape for the node
  shape: { type: 'Flow', shape: 'Terminator' }
}]
```



```
    ];  
    // initialize Diagram component  
    let diagram: Diagram = new Diagram({  
        width: '100%', height: '600px',  
        // Add node  
        nodes: nodes  
    });  
    // 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>
```

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.TS

```
import { Diagram, NodeModel, ConnectorModel } from '@syncfusion/ej2-
diagrams';
let nodes: NodeModel[] = [
  {
    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;' }]
  }
];
let connectors: ConnectorModel = {
  // Unique name for the connector
  id: "connector1",
  // Source and Target node's name to which connector needs to be
  // connected.
  sourceID: "Start",
  targetID: "Init",
  type: 'Orthogonal'
};
let diagram: Diagram = new Diagram({
  width: '100%', height: '600px', nodes: nodes, connectors: [connectors]
});
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>

```

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.TS

```

import {
    Diagram, NodeModel, ConnectorModel
} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [
    { 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' } },
];
let connector: ConnectorModel[] = [
    { 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" }
      ]
    }
  ]];
let diagram: Diagram = new Diagram({
  width: '100%', height: '600px', nodes: nodes, connectors: connector,
  getNodeDefaults: (node: NodeModel) => {
    node.height = 50;
    node.width = 140;
    node.offsetX = 300;
    return node;
  },
  getConnectorDefaults: (obj: ConnectorModel): ConnectorModel => {
    obj.type = "Orthogonal";
    obj.targetDecorator = { shape: 'Arrow', width: 10, height: 10 };
    return obj;
  }
});
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>

```

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 an 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.TS

```
import {
    Diagram, ConnectorModel, Node, DataBinding, HierarchicalTree, TreeInfo,
    SnapConstraints,
} from '@syncfusion/ej2-diagrams';
Diagram.Inject(DataBinding, HierarchicalTree);
import { DataManager, Query } from '@syncfusion/ej2-data';
export interface EmployeeInfo {
    Name: string;
    Role: string;
}
//To represent the roles
let codes: object[] = {
    Director: "rgb(0, 139,139)",
    Manager: "rgb(30, 30,113)",
    Lead: "rgb(0, 100,0)"
}
// Bind custom data with node
function getNodeTemplate(node) {
```

```

node.annotations[0].content = (node.data as EmployeeInfo).Name;
node.style.fill = codes[(node.data as EmployeeInfo).Role];
}
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));
let diagram: Diagram = new Diagram({
width: '100%', height: '600px',
snapSettings: { constraints: SnapConstraints.None },
//Use automatic layout to arrange elements on the page
layout: {
type: 'OrganizationalChart',
margin: {left: 10, top: 10},
horizontalSpacing: 50,
verticalSpacing: 50,
orientation: 'TopToBottom',
getLayoutInfo: (node: Node, tree: TreeInfo) => {
if (!tree.hasSubTree) {
tree.orientation = 'Vertical';
tree.type = 'Alternate';
}
}
},
dataSourceSettings: {
id: 'Name', parentId: 'ReportingPerson', dataManager: items
},
getNodeDefaults: (obj: Node, diagram: 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: (connector: ConnectorModel, diagram: Diagram) => {
connector.targetDecorator.shape = 'Arrow';
connector.type = 'Orthogonal';
return connector;
},
setNodeTemplate: (node: NodeModel) => {
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">

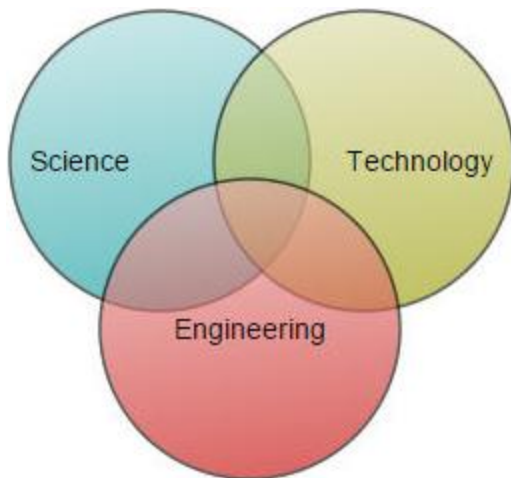
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>

  <div id="container">
    <div id="element"></div>
  </div>
<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 refer to our [JavaScript Diagram](#) feature tour page for its groundbreaking feature representations. You can also explore our [JavaScript Diagram example](#) that shows how to render the Diagram in JavaScript.

Nodes in EJ2 JavaScript Diagram control

Nodes are graphical objects used to visually represent the geometrical information, process flow, internal business procedure, entity, or any other kind of data.



<!-- markdownlint-disable MD033 -->

Create node

A node can be created and added to the diagram, either programmatically or interactively. Nodes are stacked on the diagram area from bottom to top in the order they are added.

Add node through nodes collection

To create a node, define the [node](#) object and add that to nodes collection of the diagram model. The following code example illustrates how to add a node to the diagram.

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' },  
  // Text(label) added to the node  
};  
// 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>
```

Note: Node id should not begin with numbers(should begin with a letter).Node Id should be unique for all the shapes and connectors.

Add/Remove node at runtime

- Nodes can be added at runtime by using public method, add and can be removed at runtime by using public method, remove. On adding node at runtime, the nodes collection is changed and the [collectionChange](#) event will trigger.
- The node's ID property is used to define the name of the node and its further used to find the node at runtime and do any customization.

The following code illustrates how to add a node.

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' },  
    // 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>

  <div id="container">
    <div id="element"></div>
  </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 node from palette

Nodes can be predefined and added to the palette, and can be dropped into the diagram when needed. For more information about adding nodes from symbol palette, refer to [Symbol Palette](#).

- Once you drag a node/connector from the palette to the diagram, the following events can be used to do your customization.

- When a symbol is dragged into diagram from symbol palette, the [dragEnter](#) event gets triggered.
- When a symbol is dragged over diagram, the [dragOver](#) event gets triggered.
- When a symbol is dragged and dropped from symbol palette to diagram area, the [drop](#) event gets triggered.
- When a symbol is dragged outside of the diagram, the [dragLeave](#) event gets triggered.

Create node through data source

Nodes can be generated automatically with the information provided through data source. The default properties for these nodes are fetched from default settings. For more information about data source, refer to Data Binding.

Draw nodes

Nodes can be interactively drawn by clicking and dragging the diagram surface by using [NodeDrawingTool](#). For more information about drawing nodes, refer to Draw Nodes.

Position

- Position of a node is controlled by using its [offsetX](#) and [offsetY](#) properties. By default, these offset properties represent the distance between the origin of the diagram's page and node's center point.
- You may expect this offset values to represent the distance between page origin and node's top-left corner instead of center. The Pivot property helps to solve this problem. Default value of node's [pivot](#) point is (0.5, 0.5), that means center of the node.
- The size of the node can be controlled by using its [width](#) and [height](#) properties.
- Rotation of a node is controlled by using its [rotateAngle](#) property.

The following table illustrates how pivot relates offset values with node boundaries.

Pivot	Offset
-----	-----
(0.5,0.5)	offsetX and offsetY values are considered as the node's center point.
(0,0)	offsetX and offsetY values are considered as the top-left corner of the node.
(1,1)	offsetX and offsetY values are considered as the bottom-right corner of the node.

The following code illustrates how to change the [pivot](#) value.

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' },
  pivot: {x: 0, y: 0}
};
// 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>
```

Flip

The diagram Provides support to flip the node. [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip

[Horizontal](#) is used to change the element in horizontal direction.

- VerticalFlip

[Vertical](#) is used to change the element in vertical direction

- Both

[Both](#) which involves both vertical and horizontal changes of the element.

The following code illustrates how to provide the mirror image of the original element.

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' },
  pivot: {x: 0, y: 0}
};
// 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>
```

Appearance

- The appearance of a node can be customized by changing its [fill](#) color, [borderColor](#), [borderWidth](#), [strokeDashArray](#), [opacity](#), and [shadow](#).
- The [visible](#) property of the node enables or disables the visibility of the node.

The following code illustrates how to customize the appearance of the shape.

INDEX.JS

```
var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: { fill: '#6BA5D7', strokeDashArray: '5,5', borderWidth: 2,
borderColor: 'red',
    // Text(label) added to the node
    };
// 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>
```

Note: The flip is also applicable for group and BPMN shapes.

Gradient

The [gradient](#) property of the node allows you to define and apply the gradient effect to that node.

The gradient stop property defines the color and a position, where the previous color transition ends and a new color transition starts.

The gradient stop's opacity property defines the transparency level of the region.

There are two types of gradients as follows:

- Linear gradient
- Radial gradient

Linear gradient

- [LinearGradient](#) defines a smooth transition between a set of colors (so-called stops) on a line.
- A linear gradient's x1, y1, x2, y2 properties are used to define the position (relative to the node) of the rectangular region that needs to be painted.

INDEX.JS

```
var linearGradient;
```



```

linearGradient = {
    //Start point of linear gradient
    x1: 0, y1: 0,
    //End point of linear gradient
    x2: 50, y2: 50,
    //Sets an array of stop objects
    stops: [{ color: 'white', offset: 0 },
    { color: '#6BA5D7', offset: 100 }],
    type: 'Linear'
};
var radialGradient;
radialGradient = {
    //Center point of outer circle
    cx: 50, cy: 50,
    //Center point of inner circle
    fx: 25, fy: 25,
    //Radius of a radial gradient
    r: 50,
    //Sets an array of stop objects
    stops: [{ color: 'white', offset: 0 },
    { color: '#6BA5D7', offset: 100 }],
    type: 'Radial'
};
var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {gradient: linearGradient, strokeColor: 'white' }
    // Text(label) added to the node
};
var node2 = {
    // Position of the node
    offsetX: 400,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {gradient: radialGradient, strokeColor: 'white' }
};
// 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>
```

Radial gradient

- [RadialGradient](#) defines a smooth transition between stops on a circle.
- A radial gradient's cx, cy, fx, fy properties are used to define the position (relative to the node) of the outermost or the innermost circle of the radial gradient.

INDEX.JS

```
var linearGradient;
linearGradient = {
    //Start point of linear gradient
    x1: 0, y1: 0,
    //End point of linear gradient
    x2: 50, y2: 50,
    //Sets an array of stop objects
    stops: [{ color: 'white', offset: 0 },
    { color: '#6BA5D7', offset: 100 }],
    type: 'Linear'
};
var radialGradient;
radialGradient = {
    //Center point of outer circle
```

```

        cx: 50, cy: 50,
        //Center point of inner circle
        fx: 25, fy: 25,
        //Radius of a radial gradient
        r: 50,
        //Sets an array of stop objects
        stops: [{ color: 'white', offset: 0 },
        { color: '#6BA5D7', offset: 100 }],
        type: 'Radial'
    };
var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {gradient: linearGradient, strokeColor: 'white' }
    // Text(label) added to the node
};
var node2 = {
    // Position of the node
    offsetX: 400,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {gradient: radialGradient, strokeColor: 'white' }
};
// 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>

```

Shadow

Diagram provides support to add [shadow](#) effect to a node that is disabled, by default. It can be enabled with the constraints property of the node. The following code illustrates how to drop shadow.

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' },
    constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Shadow,
    // Text(label) added to the node
};
// 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 shadow

The angle, distance, and opacity of the shadow can be customized with the shadow property of the node. The following code

example illustrates how to customize shadow.

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' },
    constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Shadow,
    shadow: { angle: 50, opacity: 0.8, distance: 9 }
};
// 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>

```

Icon

Diagram provides support to describe the state of the node. i.e., the node is expanded or collapsed state.

Note: Icon can be created only when the node has outEdges.

- To explore the properties of expand and collapse icon, refer to [expandIcon](#) and [collapseIcon](#).
- The expandIcon's and collapseIcon's shape properties allow to define the shape of the icon.

The following code example illustrates how to create an icon of various shapes.

INDEX.JS

```

var nodes = [
  {
    id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 50,
    annotations: [{
      content: 'Node1'
    }],
    style: { fill: '#6BA5D7', strokeColor: 'white' },
    expandIcon: {shape: 'ArrowDown', width: 10, height: 10},
    collapseIcon: {shape: 'ArrowUp', width: 10, height: 10}
  },
  {
    id: 'Init', width: 140, height: 50, offsetX: 300, offsetY: 140,
    style: { fill: '#6BA5D7', strokeColor: 'white' },
    annotations: [{
      content: 'Node2'
    }],
  }
];
var connectors = [{
  // Unique name for the connector
  id: "connector1",
  // Source and Target node's name to which connector needs to be
  // connected.
  sourceID: "Start",
  targetID: "Init",
  type: 'Orthogonal'
}];
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: nodes, connectors: connectors
}, '#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 expand icon

- Set the `borderColor`, `borderWidth`, and background color for an `expandIcon` using `borderColor`, `borderWidth`, and `fill` properties.
- Set a size for an `expandIcon` by using `width` and `height` properties.
- The `expand icon` can be aligned relative to the node boundaries. It has `margin`, `offset`, `horizontalAlignment`, and `verticalAlignment` settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.
- The [iconColor](#) property can be used to set the `strokeColor` of the Icon.

Customizing collapse icon

- Set the [borderColor](#), [borderWidth](#), background color for an `collapseIcon` using `borderColor`, `borderWidth`, and [fill](#) properties.
- Set a size for `collapseIcon` by using [width](#) and [height](#) properties.
- Like `expand icon`, `collapse icon` also can be aligned relative to the node boundaries. It has `margin`, `offset`, `horizontalAlignment`, and `verticalAlignment` settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.
- The [iconColor](#) property can be used to set the `strokeColor` of the Icon.

Interaction

Diagram provides support to drag, resize, or rotate the node interactively. For more information about editing a node at runtime, refer to [Edit Nodes](#).

Constraints

The `constraints` property of the node allows you to enable/disable certain features. For more information about node constraints, refer to [Node Constraints](#).

Custom properties

The [addInfo](#) property of the node allows to maintain additional information to the node.

Stack order

The nodes z-order property specifies the stack order of the node. A node with greater stack order is always in front of a node with a lower stack order.

Data flow

Node has the InEdges and OutEdges read-only property. In this property, you can find what are all the connectors that are connected to the node, and then you can find these connectors by using the [getObject](#) method in the diagram.

```
`javascript
var node = {
id: 'node1',
// Position of the node
offsetX: 450,
offsetY: 100,
// Size of the node
width: 80,
height: 50,
style: { fill: '#6BA5D7', strokeColor: 'white' },
};
var node2 = {
id: 'node2',
// Position of the node
offsetX: 350,
offsetY: 200,
// Size of the node
width: 80,
height: 50,
style: { fill: '#6BA5D7', strokeColor: 'white' },
};
var node3 = {
id: 'node3',
// Position of the node
offsetX: 450,
offsetY: 200,
// Size of the node
width: 80,
```

```
height: 50,
style: { fill: '#6BA5D7', strokeColor: 'white' },
};
var node4 = {
id: 'node4',
// Position of the node
offsetX: 550,
offsetY: 200,
// Size of the node
width: 80,
height: 50,
style: { fill: '#6BA5D7', strokeColor: 'white' },
};
var connector = {
id: 'connector1', sourceID: 'node1', targetID: 'node2', type: 'Orthogonal'
};
var connector2 = {
id: 'connector2', sourceID: 'node1', targetID: 'node3', type: 'Orthogonal'
};
var connector3 = {
id: 'connector3', sourceID: 'node1', targetID: 'node4', type: 'Orthogonal'
};
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 600, nodes: [node, node2, node3, node4],
connectors: [connector, connector2, connector3]
}, '#element');
diagram.getObject('connector1');
```

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 edit the node visual interface](#)

- [How to create diagram nodes using drawing tools](#)

Shapes in EJ2 JavaScript 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.TS

```
import { Diagram, NodeModel } 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,
  //Sets type of the shape as text
  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
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  // Add node
  nodes: [node]
});
// render initialized diagram
```

```

diagram.appendTo('#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>

```

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.TS

```
import { Diagram, NodeModel } 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,
    // sets the type of the shape as image
    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
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>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

  <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>

```

Base64 Encoded Image Into The Image Node:

The following code illustrates how add Base64 image into image node.

INDEX.TS

```

import { Diagram, NodeModel } 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,
  // sets the type of the shape as image
  shape: {
    type: 'Image',
    source:
      'data:image/gif;base64,R0lGODlhPQBEAPeoAJosM//AwO/AwHVYZ/z595kzAP/s7P+goOXMv
      8+fhw/v739/f+8PD98fH/8mJl+fn/9ZWb8/PzWlWv///6wWGbImAPgTEMImIN9gUFCEm/gDALULD
      N8PAD6atYdCTX9gUNKlj8wZAKUsAOzZz+UMAOsJAP/Z2ccMDA8PD/95eX5NWvsJCOVNQPtfX/8zM
      8+QePL138MGBr8JCP+zs9myn/8GBqwpAP/GxgwJCPny78lZYLgJAJ8vAP9fX/+MjMUCAN8zM/9wc
      M8ZGcATEL+QePdZwf/29uc/P9cmJu9MTDImIN+/r7+/vz8/P8VNQGNugV8AAF9fX8swMNgtAF1DO
      ICAGPNSUnNWSMQ5MBAQEJE3QPIGAM9AQmGcG9vb6MhJsEdGM8vLx8fH98AANIWAMuQeL8fABkTE
      PPQ00M5OSYdGf15jo+Pj/+pqcsTE78wMFNGQLYmID4dGPvd3UBAQJmTkP+8vH9QUK+vr8ZWSHpc
      JMmILdwcLOGCHRQUHxwcK9PT9DQ00/v70w5MLypoG8wKOuwsP/g4P/Q0IcwKEswKM18aJ9fX2xjd
      OtGRs/Pz+Dg4GImIP8gIH0sKEAwKKmTiKZ8aB/f39Wsl+Lft8dgUE9PT5x5aHBwcP+AgP+WltdgY
      MyZfyYwz78AAAAAAD///8AAP9mZv///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      CH5BAEAAKqALAAAAA9AEQAAj/AFEJHEiwoMGDCBMqXmiwocAbBww4nEhxoYkUpzJGrMixogkfG
      UNq1NixJEIDB0SqHGmyJSojM1bKZOmyop0gM3Oe2liTISKMOoPy7GnwY9CjIYcSRYm0aVKSLE6n

```

```

fq05QycVLPUhDrxBlCtYJUqNAq2bNWEbj6ZXruyxZyDRtqwnXvkhACDV+euTeJm1Ki7A73qNWTFi
F+/gA95Gly2CJLDhwEHMOUAAuOpLYDEgBxZ4GRTlC1fDnpkM+fOqD6DDj1aZpITp0dtGCDhr+fVu
Cu3zlg49ijaokTZTo27uG7Gjn2P+hI8+PDPERoUB318bWbfAJ5sUNFcUGRTYUqV/3ogfXp1rWlMc
6awJjiAAAd2fm4ogXjz56aypOoIde4OE5u/F9x199dlXnnGiHZWEYbGpsAEA3QXYnHwEFliKAgsWG
J8LPeiUXGwedCAKABACCN+EAlpYIIYaFlcDhytd51sGAJbo3onOpajiih1092KHGaUXGwWjUBChj
SPiWJuOO/LYIm4v1tXfE6J4gCSJEZ7YgRYUNrkji9P55sF/ogxw5ZkSqIDaZBV6aSGYq/lGZplnd
kckZ98xoICbTcIJGQAZcNmduC210hs35nCyJ58fgmIKX5RQGOZowxaZwYA+JaoKQswGijBV4C6
SiTUmpphMspJx9unX4KaimjDv9aaXOEbteBqmuuxgEHoLX6Kqx+yXqqBANsgCtit4FWQAEkrNbpq
7HSOmtwag5w57GrmlJBASEU18ADjUYb3ADTinIttsgSBloJFFa63bduimuqKB1keqwUhoCSK374w
bujvOSu4QG6UvxBRydcPksav++Ca6G8A6Pr1x2kVMYHwsVxUALDq/krrrhPSOzXG11UTIoffqGR7
Goi2MAxbv602kEG56I7CSlRsEFKfVYovDJoIRTg7sugNRDGqCJzJgcKE0ywc0ELm6KBCCJo8DIPF
eCWNGcyqNFE06ToAfV0HBRgxsvLThHn1oddQMrXj5DyAQgjEHSaJMWZwS3HPxT/QMbabi/iBCliM
LEJXK2EEkomBAUCxRi42VDAxyTYDVogV+wSCHqmKxEKCDAYFDFj4OmwbY7bDGDhtrntQYOigeC
hUmc1K3QTnAUfEgGFgAwT88hKA6aCRIXhxnQ1yg3BCayK44EWdkUQcBBYEQChFXfCB776aQsG0BI
lQgQgE8q026X1h8cEUep8ngRBnOy74E9QgRgEAC8SvOfQkh7FDBDmS43PmGoIiKUUEGkMEC/PJHg
xw0xH74yx/3XnaYRJgMB8obxQW6kL9QYEJ0FIFgByFIL7/IQAlvQwEpnAC7DtLNJCKUoO/w45c44
GwCXiAFB/OXAATQryUxdN4LfFiwgjCNYg+kYMIEFkCKDs6PKAIJouyGWMS1FSKJOMRB/BoIxYJIU
XFUXnWoIkEKPAGCBZSQHQ1A2EWDfDEUVLyAdj5AchSIQW6gu10bE/JG2VnCZGfo4R4d0sdQoBAHh
PjhIB94v/wRoRKQWGRHgrhGSQJxCS+0pCZbEhAAOw== '
    },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: {
        fill: 'none'
    }
};
// 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>
  <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.TS

```

import {
  Diagram,
  NodeModel,
  NodeConstraints
} 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,
  //sets the type of the shape as Image

```



```

        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
    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">

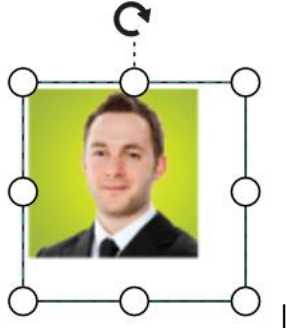
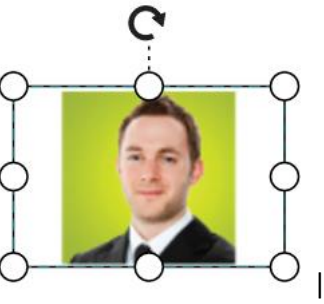
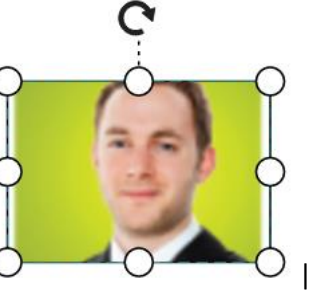
<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
    <script id="nodetemplate" type="text/x-template">
        <input type="button" id="button" value="{id}">
    </script>

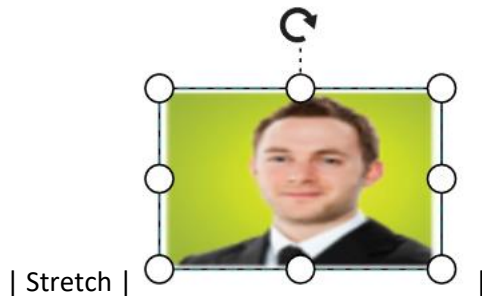
    <div id="container">
        <div id="element"></div>

```

```
</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
None	
Meet	
Slice	



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.TS

```
import { Diagram, NodeModel, NodeConstraints } 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'
  },
  //sets the type of the shape as HTML
  shape: {
    type: 'HTML',
    content: '<div
style="background:#6BA5D7;height:100%;width:100%;"><button type="button"
style="width:100px"> Button</button></div>'
  }
  as NodeModel
};
// 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>
  <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.TS

```

import { Diagram, NodeModel, NodeConstraints } from '@syncfusion/ej2-
diagrams';
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  //Id of the node
  id: "Node",

```

```

    //Position of the node
    offsetX: 100,
    offsetY: 100,
    //Size of the node
    width: 100,
    height: 100,
    //sets the type of the shape as HTML
    shape: {
        type: 'HTML'
    }
}
];
//initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: nodes,
    nodeTemplate: "#nodetemplate"
});
//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>
    <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.TS

```

import { Diagram, NodeModel, NodeConstraints } 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,
  //sets the type of the shape as Native
  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 C62.726,435.13,62.354,435.072,61.979,435.057"/><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-
3.335,8.671-0.054,12.01218.533,8.533      c3.341,3.281,8.701,3.256,12.012-
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-
3.311,3.335-8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
      <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-
3.335-12.012-0.0541-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/>      <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/>      <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/>      <path
d="M164.636,454.7711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
      <path d="M232.903,429.1711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
      <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-
12.012-0.0541-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/>      <path
d="M70.77,463.3041-8.533,8.533c-2.188,2.149-3.055,5.307-
```

```
2.27,8.271s3.1,5.28,6.065,6.065      c2.965,0.785,6.122-0.082,8.271-
2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/>      <path
d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
      <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.51-8.533,8.533      c-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-
0.073,8.262-2.25118.533-8.533      c1.601-1.6,2.5-3.77,2.5-
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-3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
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-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
      <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.01218.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>      <path
d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.05418.533,8.533c-
2.44,2.441-3.169,6.11-
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-8.671-3.335-12.012-0.05418.
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>'
    },
};
// 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>
  <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.

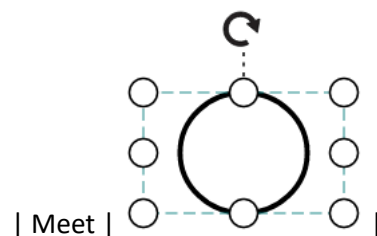
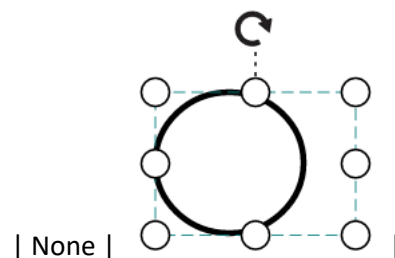
The scale property of the node allows to stretch the svg content as you desired (either to maintain proportion or to stretch). By default, the `scale` property of node is set as **meet**.

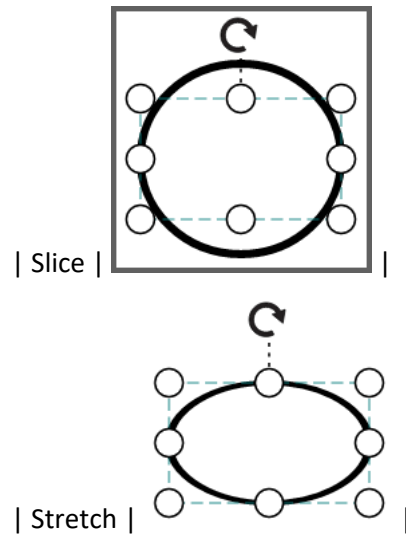
The following code illustrates how to scale or stretch the content of the node.

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.TS

```
import { Diagram, NodeModel, NodeConstraints} 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,
  //sets the type of the shape as Basic
  shape: {
    type: 'Basic',
    shape: 'Rectangle',
    cornerRadius: 10
  },
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
};
// 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>
    <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.TS

```
import { Diagram, NodeModel, NodeConstraints } 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,
  //sets the type of the shape as Path
  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
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>
  <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.TS

```

import { Diagram, NodeModel, NodeConstraints } 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,
  //sets the type of the shape as Flow

```

```
    shape: {
      type: 'Flow',
      shape: 'Document'
    },
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
  },
};
// 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>
  <script id="nodetemplate" type="text/x-template">
    <input type="button" id="button" value="{id}">
  </script>

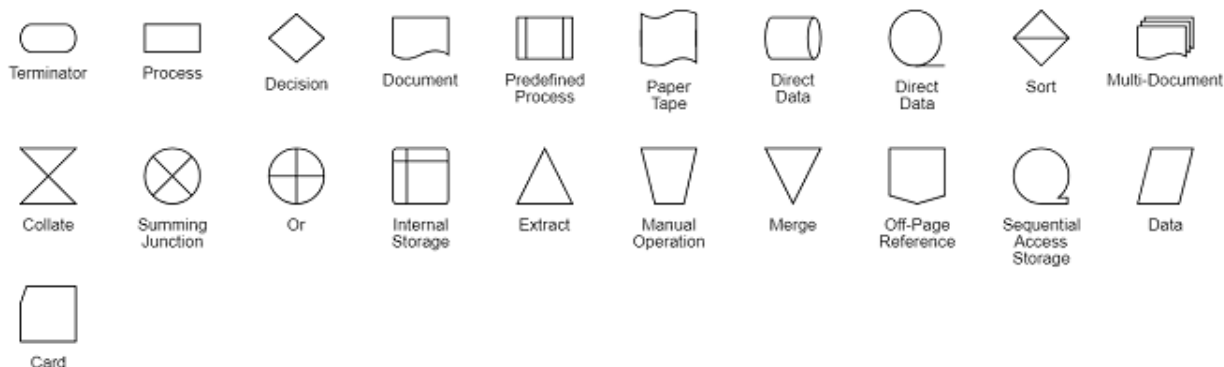
  <div id="container">
    <div id="element"></div>
  </div>
</body>
</html>
```

```

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

```

The list of flow shapes are as follows.



Bpmn shapes in EJ2 JavaScript 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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  //Sets type as Bpmn and shape as Event
  shape: {
    type: 'Bpmn',
    shape: 'Event',
    // set the event type as End
    event: {
      event: 'End'
    }
  },
};
// 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>

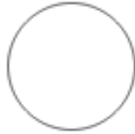
  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>
```

Note : The default value for the property `shape` is "event".

The list of BPMN shapes are as follows:

| Shape | Image |

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



| Event |



| Gateway |



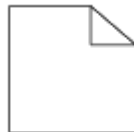
| Task |



| Message |



| DataSource |



| DataObject |



| Group |

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as event
    shape: {
        type: 'Bpmn',
        shape: 'Event',
        // Sets event as End and trigger as None
        event: {
            event: 'End',
            trigger: 'None'
        }
    },
};
// 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>

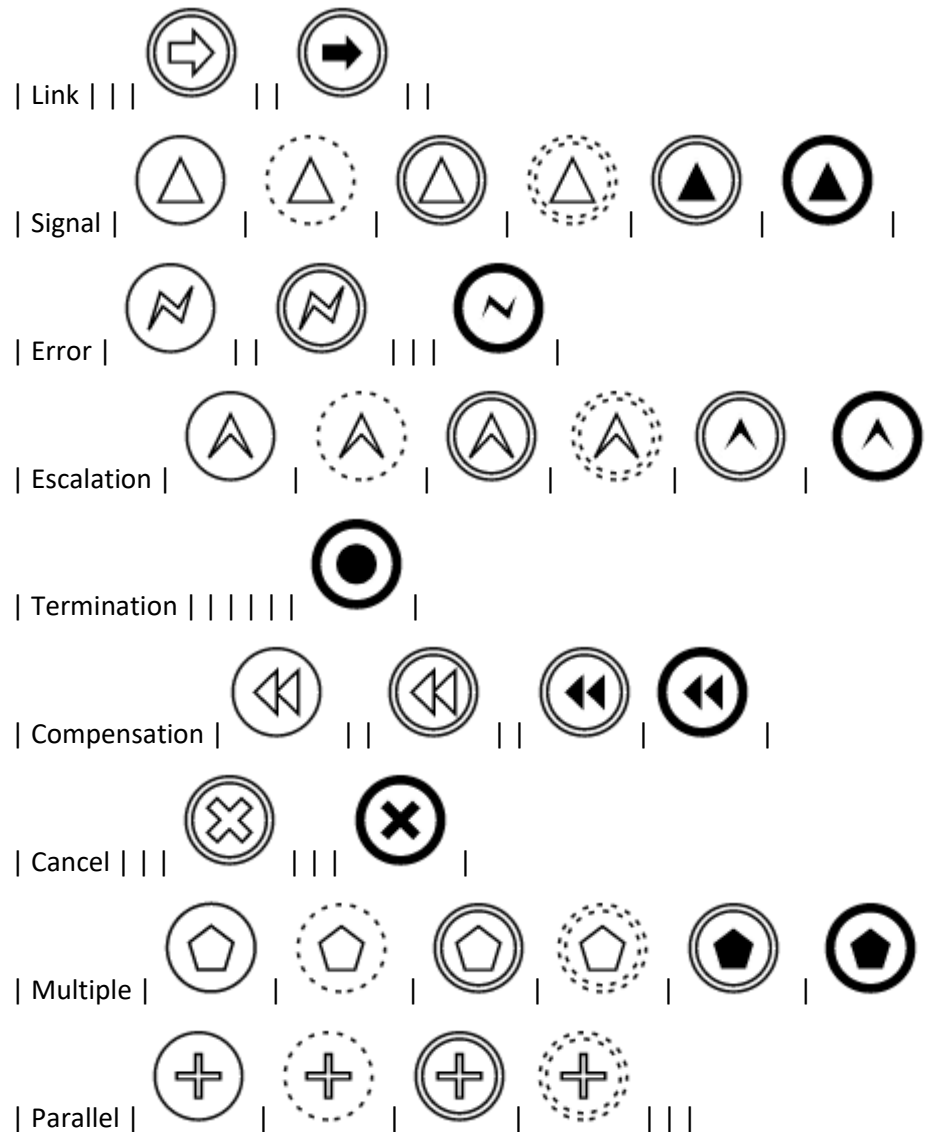
    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

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 |

	-----	-----	-----	-----	-----	-----	-----
None							
Message							
Timer							
Conditional							



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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
//Sets type as Bpmn and shape as Gateway
shape: {
  type: 'Bpmn',
  shape: 'Gateway',
  //Sets type of the gateway as None
  gateway: {
    type: 'None'
  }
  as BpmnGatewayModel
},
};
// 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>
```

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as Task
    activity: {
      activity: 'Task'
    }
  },
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as Task
        activity: {
            activity: 'Task',
            //Sets the type of the task as Send
            task: {
                type: 'Send'
            }
        },
    },
};
// 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>
```


The various types of BPMN tasks are tabulated as follows.

Shape Image
----- -----

		
Service		

		
Send		

		
Receive		

		
Instantiating Receive		



| Manual |



| Business Rule |



| User |



| Script |

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
```

```

        //Sets activity as Subprocess and collapsed of subprocess as true
        activity: {
            activity: 'SubProcess',
            subProcess: {
                collapsed: true
            }
        } as BpmnSubProcessModel
    },
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets the collapsed as true and type as Event
      subProcess: {
        collapsed: true,
        type: 'Event',
        //Sets event as Start and trigger as Message
        event: {
          event: 'Start',
          trigger: 'Message'
        }
      }
    },
    as BpmnSubProcessModel
  },
},
};
// 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>
```

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets collapsed as true and type as Transition
      subprocess: {
        collapsed: true,
        type: 'Transition',
        //Sets event as Intermediate and trigger as Cancel
        event: [{
          event: 'Intermediate',
          trigger: 'Cancel',
          offset: {
            x: 0.25,
            y: 1
          }
        },
        {
          event: 'Intermediate',
          trigger: 'Error',
          offset: {
            x: 0.25,
```



```

        y: 1
      },
    ],
  },
  as BpmnSubProcessModel
},
},
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as Task
    activity: {
      activity: 'Task',
      //Sets loop of the task as Standard
      task: {
        loop: 'Standard'
      }
    },
  },
};
let node2: NodeModel = {
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets Activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets collapsed as true and loop as Standard

```

```

        subprocess: {
            collapsed: true,
            loop: 'Standard'
        }
        as BpmnSubProcessModel
    },
},
});
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node, node2]
});
// 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">

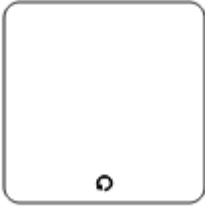
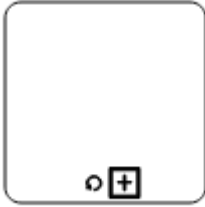


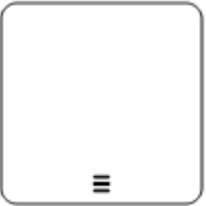
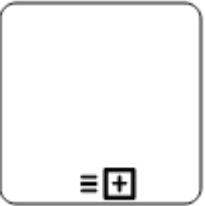
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
    <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>
```

The following table contains various types of BPMN loops.

Loops	Task	Subprocess
-----	-----	-----
Standard		
SequenceMultiInstance		
ParallelMultiInstance		

Compensation
[Compensation](#) is triggered, when operation is partially failed and enabled it with the compensation property of the task and the subprocess.

INDEX.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
```

```

        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as Task
        activity: {
            activity: 'Task',
            //Sets compensation of the task as true
            task: {
                compensation: true
            }
        },
    },
};
let node2: NodeModel = {
    // Position of the node
    offsetX: 300,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as SubProcess
        activity: {
            activity: 'SubProcess',
            //Set the collapsed as true and compensation as true
            subprocess: {
                collapsed: true,
                compensation: true
            }
        }
        as BpmnSubProcessModel
    },
};
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node, node2]
});
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets the activity as task
        activity: {

```

```

        activity: 'Task',
        //Sets the call of the task as true
        task: {
            call: true
        }
    },
},
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as SubProcess
        activity: {
            activity: 'SubProcess',
            //Sets collapsed as true and adhoc as true
            subprocess: {
                collapsed: true,
                adhoc: true
            }
        },
        as BpmnSubProcessModel
    },
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',

```

```

        shape: 'Activity',
        //Sets activity as SubProcess
        activity: {
            activity: 'SubProcess',
            //Sets collapsed as true and boundary as Call
            subprocess: {
                collapsed: true,
                boundary: 'Call'
            }
            as BpmnSubProcessModel
        },
    },
};
// 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>

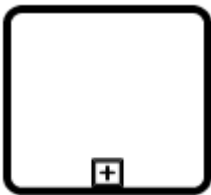
    <div id="container">
        <div id="element"></div>
    </div>
```

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

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnShapeModel,
BpmnDiagrams, BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from
'@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
    //Sets type as Bpmn and shape as DataObject
    shape: {
        type: 'Bpmn',
        shape: 'DataObject',
        //Sets collection as true and type as Input
        dataObject: {
            collection: true,
            type: 'Input'
        }
    }
    as BpmnShapeModel,
};
// 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">

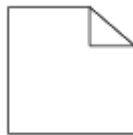
  <script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
type="text/javascript"></script>
  <script src="https://cdn.syncfusion.com/ej2/syncfusion-helper.js" type
="text/javascript"></script>
</head>
<body>
```

```
<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>
```

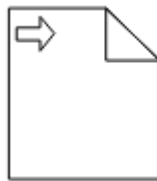
The following table contains various representation of BPMN data object.

| Boundary | Image |

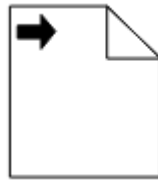
| ----- | ----- |



| 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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel, BpmnDiagrams,
BpmnActivityModel, BpmnFlowModel, BpmnGatewayModel } from '@syncfusion/ej2-
diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,  
//Sets type as Bpmn and shape as DataSource  
shape: {  
    type: 'Bpmn',  
    shape: 'DataSource',  
}  
};  
// 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>
```

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.TS

```
import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel,
BpmnShapeModel, BpmnDiagrams, BpmnActivityModel, BpmnFlowModel,
BpmnGatewayModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as DataObject
  shape: {
    type: 'Bpmn',
    shape: 'DataObject',
    //Sets collection as true and type as Input
    dataObject: {
      collection: true,
      type: 'Input'
    },
  },
};
```

```
//Sets the id, angle, length and text for the annotation
annotations: [{
    id: 'left',
    angle: 45,
    length: 150,
    text: 'Left',
}]
}
as BpmnShapeModel,
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, BpmnShape, BpmnSubProcessModel,
BpmnShapeModel, BpmnDiagrams, BpmnActivityModel, BpmnFlowModel,
BpmnGatewayModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// 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,
  // Sets type as Bpmn and shape as Group
  shape: {
    type: 'Bpmn',
    shape: 'Group',
  }
};
// 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>

```

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.TS

```

import { Diagram, NodeModel, ConnectorModel, BpmnShape, BpmnSubProcessModel,
BpmnShapeModel, BpmnDiagrams, BpmnActivityModel, BpmnFlowModel,
BpmnGatewayModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// A node is created and stored in nodes array.
let connector: ConnectorModel = {
    // Position of the node
    sourcePoint: {

```

```

        x: 100,
        y: 200
    },
    targetPoint: {
        x: 300,
        y: 200
    },
    //Sets type of the connector as Orthogonal
    type: 'Orthogonal',
    //Sets type as Bpmn, shflowape as Association and association as
    BiDirectional
    shape: {
        type: 'Bpmn',
        flow: 'Association',
        association: 'BiDirectional'
    },
};
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add Connectors
    connectors: [connector]
});
// 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>
</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.TS

```

import { Diagram, NodeModel, ConnectorModel, BpmnShape, BpmnSubProcessModel,
BpmnShapeModel, BpmnDiagrams, BpmnActivityModel, BpmnFlowModel,
BpmnGatewayModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams);
// A node is created and stored in nodes array.
let connector: ConnectorModel = {
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
};

```

```

    type: 'Orthogonal',
    //Sets type as Bpmn, flow as Sequence, and sequence as Conditional
    shape: {
        type: 'Bpmn',
        flow: 'Sequence',
        sequence: 'Conditional'
    },
};
// initialize diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    connectors: [connector]
});
// 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>


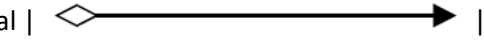
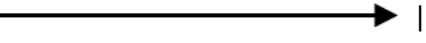
    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
}
```

```

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

```

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.TS

```

import { Diagram, NodeModel, ConnectorModel, BpmnShape, BpmnSubProcessModel,
BpmnShapeModel, BpmnDiagrams, BpmnActivityModel, BpmnFlowModel,
BpmnGatewayModel } from '@syncfusion/ej2-diagrams';
Diagram.Inject(BpmnDiagrams)
// A node is created and stored in nodes array.
let connector: ConnectorModel = {
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
  type: 'Orthogonal',
  // Sets type as Bpmn, flow as Message, and message as InitiatingMessage
  shape: {
    type: 'Bpmn',
    flow: 'Message',
    message: 'InitiatingMessage'
  },
};
// initialize diagram component
let diagram: Diagram = new Diagram({

```

```

    width: '100%',
    height: '600px',
    // Add node
    connectors: [connector]
  });
  // 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>

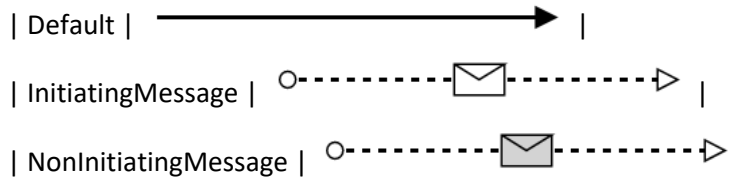
  <div id="container">
    <div id="element"></div>
  </div>
  <script>
var ele = document.getElementById('container');
if(ele) {
  ele.style.visibility = "visible";
}
  </script>
  <script src="index.js" type="text/javascript"></script>
</body></html>

```

The following table contains various representation of message flows.

| Message | Image |

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



Note: The default value for the property `message` is **default**.

UML diagram in EJ2 JavaScript 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.TS

```
import {
  Diagram,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-diagrams";
let node: NodeModel = {
  id: "Patient",
  style: {
    fill: '#26A0DA',
  },
  //Position of the node
  offsetX: 200,
  offsetY: 200,
  shape: {
    type: "UmlClassifier",
```



```

//Define class object
classShape: {
  name: "Patient",
  //Define class attributes
  attributes: [{ name: "accepted", type: "Date" }],
  //Define class methods
  methods: [{ name: "getHistory", type: "getHistory" }]
},
classifier: "Class"
} as UmlClassifierShapeModel
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  //Add node
  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-
  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>
</html>

```

```
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.TS

```
import {
  Diagram,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-diagrams";
let node: NodeModel = {
  id: "Patient",
  //Position of the node
  offsetX: 200,
  offsetY: 200,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: "UmlClassifier",
    //Define interface object
    interfaceShape: {
      name: "Patient",
      //Define interface attributes
      attributes: [{ name: "owner", type: "String[*]" }],
      //Define interface methods
      methods: [
        {
          name: "deposit",
          parameters: [
            {

```

```

        name: "amount",
        type: "Dollars"
    }
    ]
    }
    ],
    classifier: "Interface"
} as UmlClassifierShapeModel
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    //Add node
    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-
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.TS

```

import {
  Diagram,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-diagrams";
let node: NodeModel = {
  id: "Patient",
  offsetX: 200,
  offsetY: 200,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: "UmlClassifier",
    //Define enumeration object
    enumerationShape: {
      name: "AccountType",
      //set the members of enumeration
      members: [
        {
          name: "Checking Account",
        },
        {
          name: "Savings Account"
        },
        {
          name: "Credit Account"
        }
      ]
    },
    classifier: "Enumeration"
  } as UmlClassifierShapeModel
};
//Initializes diagram control

```

```
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  //Add node
  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-
  popups/styles/material.css" rel="stylesheet">

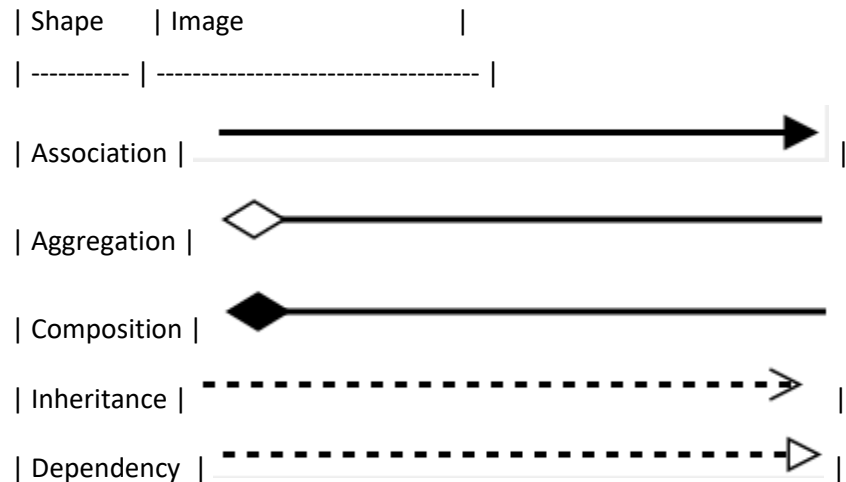
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
  popups/styles/material.css" rel="stylesheet">
  <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.TS

```
import {
  Diagram,
  ConnectorModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Association",
    //Define type of association
    association: "BiDirectional"
  }
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
```

```
//Add connector
connectors: [connector]
});
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-
  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.TS

```
import {
    Diagram,
    ConnectorModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
    id: "connector",
    //Define connector start and end points
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        //Set an relationship for connector
        relationship: "Aggregation"
    }
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    connectors: [connector]
});
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-
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>
</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.TS

```
import {
  Diagram,
  ConnectorModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Composition"
  }
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  //Add connector
  connectors: [connector]
});
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-
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.TS

```

import {
    Diagram,
    ConnectorModel,
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
    id: "connector",
    //Define connector start and end points
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",

```

```
//Set relationship for connector
relationship: "Dependency"
}
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: [connector]
});
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-
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>
```

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.TS

```
import {
  Diagram,
  ConnectorModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //set an relation of connector
    relationship: "Inheritance"
  }
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: [connector]
});
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-
  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.TS

```

import {
    Diagram,
    ConnectorModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
    id: "connector",
    //Define connector start and end points

```

```

sourcePoint: { x: 100, y: 100 },
targetPoint: { x: 300, y: 300 },
type: "Straight",
shape: {
  type: "UmlClassifier",
  relationship: "Dependency",
  multiplicity: {
    //Set multiplicity type
    type: "OneToMany",
    //Set source label to connector
    source: {
      optional: true,
      lowerBounds: 89,
      upperBounds: 67
    },
    //Set target label to connector
    target: {
      optional: true,
      lowerBounds: 78,
      upperBounds: 90
    }
  }
}
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: [connector]
});
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-
  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];

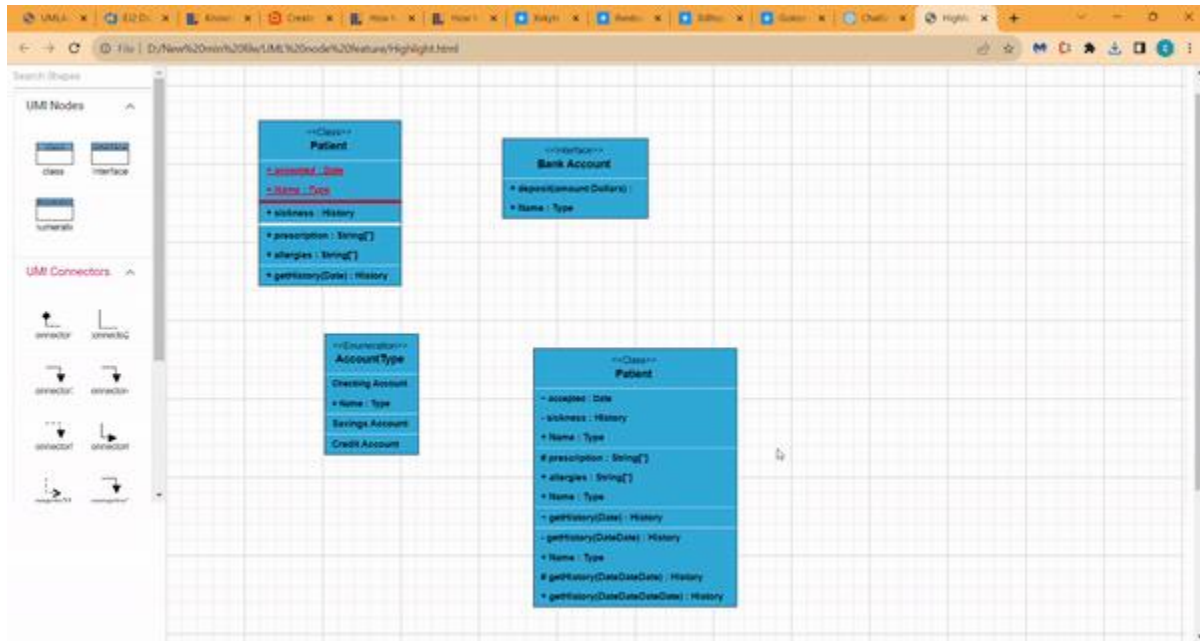
let members = { name: 'Checking new', style: { color: "red", }, isSeparator: true };

```

```
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.TS

```
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo
} from '@syncfusion/ej2-diagrams';
let diagram: Diagram = new Diagram({
  width: '100%', height: '500px'
});
diagram.appendTo('#diagram');
//Initialize the basicshapes for the symbol palette
export function getUmlShapes(): NodeModel[] {
  let umlShapes: NodeModel[] = [
    {
      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 umlShapes;
}
function setPaletteNodeDefaults(node:NodeModel) {
    node.width = 100;
    node.height = 100;
}
let palette: SymbolPalette = new SymbolPalette({

```

```

    palettes: [
      { id: 'UML', expanded: true, symbols: getUmlShapes(), title: 'UMLClass
Nodes' },
    ],
    width: '100%', height: '100%', symbolHeight: 90, symbolWidth: 90,
    getNodeDefaults: setPaletteNodeDefaults,
    symbolMargin: { left: 12, right: 12, top: 12, bottom: 12 },
    //Defines the symbol description for the symbols in the palette
    getSymbolInfo: (symbol: NodeModel): SymbolInfo => {
      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>

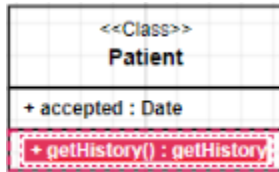
```

```
</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

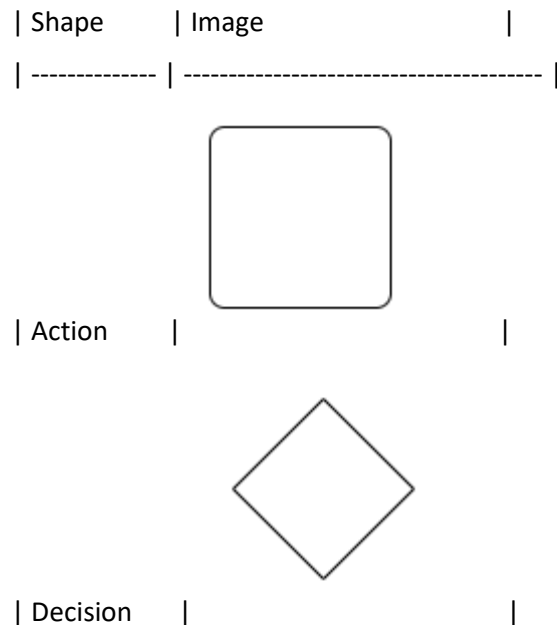
An Activity diagram functions as a visual flowchart, illustrating the progression from one activity to the next within a system. Each activity corresponds to a system operation, providing a clear depiction of the sequential flow in a dynamic process..

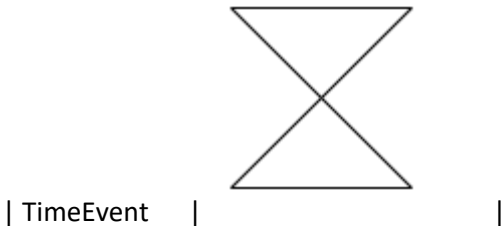
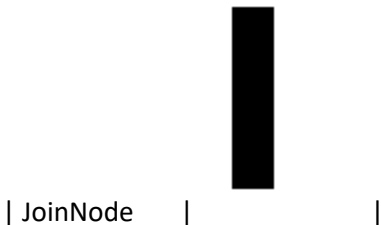
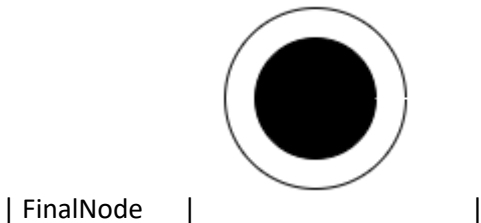
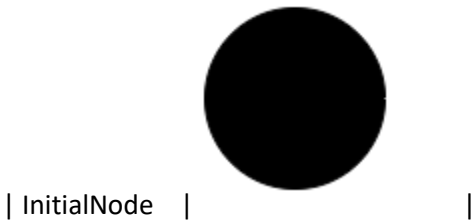
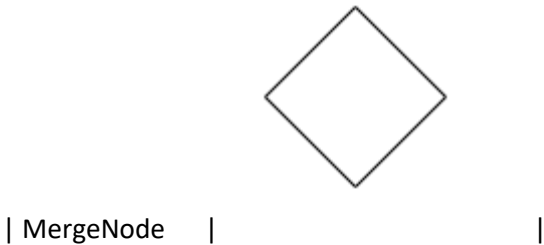
The purpose of an activity diagram can be described as follows.

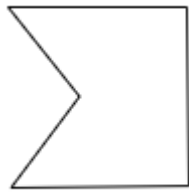
1. Draw the activity flow of a system.
2. Describe the sequence from one activity to another.
3. Describe the parallel, branched, and concurrent flow of the system.

UML Activity diagram Shapes

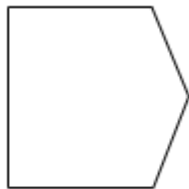
To create a UmlActivity, define the type as "UmlActivity" and the list of built-in shapes as demonstrated as follows and it should be set in the "shape" property.



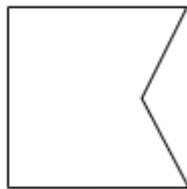




| AcceptingEvent |



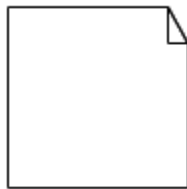
| SendSignal |



| ReceiveSignal |



| StructuredNode |



| Note |

The following code illustrates how to create a UmlActivity shapes.

INDEX.TS

```
import {  
    Diagram,  
    NodeModel,  
    UmlClassifierShapeModel
```

```
} from "@syncfusion/ej2-diagrams";
let node: NodeModel = {
  id: "UmlDiagram",
  //Set node size
  width: 100,
  height: 100,
  //position the node
  offsetX: 200,
  offsetY: 200,
  shape: {
    type: "UmlActivity",
    //Define UmlActivity shape
    shape: "Action"
  }
};
//Initializes diagram control
let diagram: Diagram = new 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-
  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>

```

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.TS

```

import {
    Diagram,
    ConnectorModel,
    UmlClassifierShapeModel
} from "@syncfusion/ej2-diagrams";
let connector: ConnectorModel = {
    id: 'connector',
    type: 'Straight',
    //Define connector start and end points
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 200, y: 200 },
    shape: { type: 'UmlActivity', flow: 'Exception' }
};
//Initializes diagram control
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    connectors: [connector]
});
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-
    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>

```

Connectors in EJ2 JavaScript Diagram control

Connectors are objects used to create link between two points, nodes or ports to represent the relationships between them.

Create connector

Connector can be created by defining the source and target point of the connector. The path to be drawn can be defined with a collection of segments. To explore the properties of a [connector](#), refer to [Connector Properties](#).

Add connectors through connectors collection

The [sourcePoint](#) and [targetPoint](#) properties of connector allow you to define the end points of a connector.

The following code example illustrates how to add a connector through connector collection.

INDEX.JS

```

var connectors = [{
    id: "connector1",
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {

```



```

        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
}
]]
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: connectors
}, '#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 connector at runtime

Connectors can be added at runtime by using public method, `diagram.add` and can be removed at runtime by using public method, `diagram.remove`.

The following code example illustrates how to add connector at runtime.

INDEX.JS

```

var connectors = {
  // Unique name for the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
}, '#element');
diagram.appendTo('#element');
diagram.add(connectors)

```

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>

```

Connectors from palette

Connectors can be predefined and added to the symbol palette. You can drop those connectors into the diagram, when required.

For more information about adding connectors from symbol palette, refer to [Symbol Palette](#).

Draw connectors

Connectors can be interactively drawn by clicking and dragging on the diagram surface by using [drawingObject](#).

For more information about drawing connectors, refer to [Draw Connectors](#).

Update connector at runtime

Various connector properties such as [sourcePoint](#), [targetPoint](#), [style](#), [sourcePortID](#), [targetPortID](#), etc., can be updated at the runtime.

The following code example illustrates how to update a connector's source point, target point, styles properties at runtime.

INDEX.JS

```

var connectors = [{
    // Unique name for the connector
    id: "connector1",
    sourcePoint: {
        x: 100,

```

```
        y: 100
      },
      targetPoint: {
        x: 200,
        y: 200
      }
    }
  }];
  var diagram = new ej.diagrams.Diagram({
    width: 1500,
    height: 1500,
    connectors: connectors
  }, '#element');
  diagram.appendTo('#element');
  diagram.connectors[0].style.strokeColor = '#6BA5D7';
  diagram.connectors[0].style.fill = '#6BA5D7';
  diagram.connectors[0].style.strokeWidth = 2;
  diagram.connectors[0].targetDecorator.style.fill = '#6BA5D7';
  diagram.connectors[0].targetDecorator.style.strokeColor = '#6BA5D7';
  diagram.connectors[0].sourcePoint.x = 150;
  diagram.connectors[0].targetPoint.x = 150;
  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>
</body>
</html>
```

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

Connect nodes

- The [sourceID](#) and [targetID](#) properties allow to define the nodes to be connected.
- The [connectorSpacing](#) property allows you to define the distance between the source node and the connector. It is the minimum distance the connector will re-route or the new segment will create.
- The following code example illustrates how to connect two nodes.

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",
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
```

```

        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourceID: "Start",
    targetID: "Init",
    connectorSpacing: 7,
    type: 'Orthogonal'
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    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>

```

- When you remove NodeConstraints [InConnect](#) from Default, the node accepts only an outgoing connection to dock in it. Similarly, when you remove NodeConstraints [OutConnect](#) from Default, the node accepts only an incoming connection to dock in it.
- When you remove both InConnect and OutConnect NodeConstraints from Default, the node restricts connector to establish connection in it.
- The following code illustrates how to disable InConnect constraints.

```

`javascript
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
  nodes:[
    {
      id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 150,
      shape: { type: 'Basic', shape: 'Rectangle' },
      //Disable InConnect constraints
      constraints: NodeConstraints.Default & ~NodeConstraints.InConnect,
    }
  ],'#diagram');
`

```

Connections with ports

The [sourcePortID](#) and [targetPortID](#) properties allow to create connections between some specific points of source/target nodes.

The following code example illustrates how to create port to port connections.

INDEX.JS

```

var port1 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}

```

```
    }
  }
  port1.shape = 'Circle';
  port1.id = 'nodeportnew';
  port1.visibility = ej.diagrams.PortVisibility.Visible;
  port1.id = 'port';
  port1.offset = {
    x: 1,
    y: 1
  };
  var port2 = {
    style: {
      strokeColor: '#366F8C',
      fill: '#366F8C'
    }
  };
  port2.offset = {
    x: 1,
    y: 0.5
  };
  port2.id = 'port1';
  port2.visibility = ej.diagrams.PortVisibility.Visible;
  port2.shape = 'Circle';
  var port3 = {
    style: {
      strokeColor: '#366F8C',
      fill: '#366F8C'
    }
  };
  port3.offset = {
    x: 0,
    y: 1
  };
  port3.id = 'newnodeport1';
  port3.visibility = ej.diagrams.PortVisibility.Visible;
  port3.shape = 'Circle';
  var nodes = [{
    id: 'node',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    ports: [port1]
  },
  {
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    ports: [port2, port3]
  },
  ];
  var connectors = {
    id: "connector1",
    sourcePoint: {
      x: 100,
```



```

        y: 100
      },
      type: 'Orthogonal',
      targetPoint: {
        x: 200,
        y: 200
      },
      sourceID: 'node',
      targetID: 'node1',
      sourcePortID: 'port',
      targetPortID: 'port1'
    }
  }
  var diagram = new ej.diagrams.Diagram({
    width: 900,
    height: 900,
    nodes: nodes,
    connectors: [connectors],
    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>
```

Similarly, the `sourcePortID` or `targetPortID` can be changed at the runtime by changing the port [sourcePortID](#) or [targetPortID](#).

INDEX.JS

```
var port1 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}
port1.shape = 'Circle';
port1.id = 'nodeportnew'
port1.visibility = ej.diagrams.PortVisibility.Visible
port1.id = 'port';
port1.offset = {
  x: 1,
  y: 1
};
var port2 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
port2.offset = {
  x: 1,
  y: 0.5
};
port2.id = 'port1';
port2.visibility = ej.diagrams.PortVisibility.Visible
port2.shape = 'Circle';
var port3 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
port3.offset = {
  x: 0,
  y: 1
};
port3.id = 'newnodeport1';
port3.visibility = ej.diagrams.PortVisibility.Visible
```

```

port3.shape = 'Circle';
var nodes = [{
    id: 'node',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    ports: [port1]
},
{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    ports: [port2, port3]
},
];
var connectors = {
    id: "connector1",
    sourcePoint: {
        x: 100,
        y: 100
    },
    type: 'Orthogonal',
    targetPoint: {
        x: 200,
        y: 200
    },
    sourceID: 'node',
    targetID: 'node1',
    sourcePortID: 'port',
    targetPortID: 'port1'
}
var diagram = new ej.diagrams.Diagram({
    width: 900,
    height: 900,
    nodes: nodes,
    connectors: [connectors],
    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>

```

- When you set PortConstraints to [InConnect](#), the port accepts only an incoming connection to dock in it. Similarly, when you set PortConstraints to [OutConnect](#), the port accepts only an outgoing connection to dock in it.
- When you set PortConstraints to None, the port restricts connector to establish connection in it.

```

`javascript
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
nodes:[
{
id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 150,
shape: { type: 'Basic', shape: 'Rectangle' },
ports: [
//Enable portConstraints Inconnect

```

```
{ id: 'port', height: 10, width: 10, offset: { x: 1, y: 0.5 }, constraints: PortConstraints.InConnect },
]
}
]
},'#diagram');
`
```

Segments

The path of the connector is defined with a collection of segments. There are three types of segments.

Straight

To create a straight line, specify the [type](#) of the segment as **straight** and add a straight segment to [segments](#) collection and need to specify [type](#) for the connector. The following code example illustrates how to create a default straight segment.

INDEX.JS

```
var connectors = {
  // Unique name for the connector
  id: "connector1",
  segments: [{
    type: 'Straight'
  }],
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: [connectors]
}, '#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 [point](#) property of straight segment allows you to define the end point of it. The following code example illustrates how to define the end point of a straight segment.

INDEX.JS

```

var connectors = {
    // Unique name for the connector
    id: "connector1",
    type: 'Straight',
    segments: [{
        type: 'Straight',
        point: {
            x: 100,
            y: 150
        }
    }],
    style: {
        strokeColor: '#6BA5D7',
    }
}

```

```

        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 300,
        y: 200
    }
}
}
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: [connectors]
}, '#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>

```

Orthogonal

Orthogonal segments is used to create segments that are perpendicular to each other.

Set the segment [type](#) as orthogonal to create a default orthogonal segment and need to specify [type](#). The following code example illustrates how to create a default orthogonal segment.

Multiple segments can be defined one after another. To create a connector with multiple segments, define and add the segments to [connector.segments](#) collection. The following code example illustrates how to create a connector with multiple segments.

The property [maxSegmentThumb](#) is used to limit the number of segment thumb in the connector.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var connectors = [{
    id: "connector1",
    type: 'Orthogonal',
    segments: [{
        type: 'Orthogonal',
        // Defines the direction for the segment lines
        direction: 'Right',
        // Defines the length for the segment lines
        length: 50
    }],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
},
{

```



```

    id: "connector2",
    type: 'Orthogonal',
    // Defines multile segments for the connectors
    segments: [{
        type: 'Orthogonal',
        direction: 'Bottom',
        length: 150
    },
    {
        type: 'Orthogonal',
        direction: 'Right',
        length: 150
    }
    ],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 300,
        y: 100
    },
    targetPoint: {
        x: 400,
        y: 200,
    },
    maxSegmentThumb: 3,
    constraints: ej.diagrams.ConnectorConstraints.Default |
    ej.diagrams.ConnectorConstraints.DragSegmentThumb
    }
]
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: connectors,
}, '#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">
```

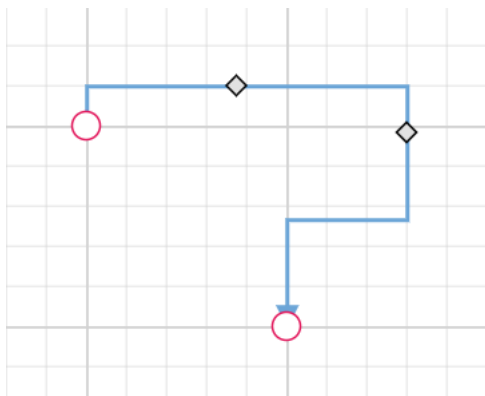
```

<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 [length](#) and [direction](#) properties allow to define the flow and length of segment. The following code example illustrates how to create customized orthogonal segments.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var connectors = [{
    id: "connector1",
    type: 'Orthogonal',
    segments: [{
        type: 'Orthogonal',
        // Defines the direction for the segment lines
        direction: 'Right',

```

```

        // Defines the length for the segment lines
        length: 50
    }],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
},
{
    id: "connector2",
    type: 'Orthogonal',
    // Defines multile segemnts for the connectors
    segments: [{
        type: 'Orthogonal',
        direction: 'Bottom',
        length: 150
    },
    {
        type: 'Orthogonal',
        direction: 'Right',
        length: 150
    }
    ],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 300,
        y: 100
    },
    targetPoint: {
        x: 400,
        y: 200,
    },
},

```

```

        maxSegmentThumb: 3,
        constraints: ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.DragSegmentThumb
    }
]
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: connectors,
}, '#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: You need to mention the segment type as same as what you mentioned in connector type. There should be no contradiction between connector type and segment type.

Avoid overlapping

Orthogonal segments are automatically re-routed, in order to avoid overlapping with the source and target nodes. The following preview illustrates how orthogonal segments are re-routed.

INDEX.JS

```
var nodeport = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
nodeport.shape = 'Circle';
nodeport.visibility = ej.diagrams.PortVisibility.Visible;
nodeport.id = 'port';
nodeport.offset = {
  x: 0,
  y: 0.5
};
var port2 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}
port2.offset = {
  x: 0,
  y: 0.5
};
port2.id = 'port1';
port2.visibility = ej.diagrams.PortVisibility.Visible;
port2.shape = 'Circle';
var nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  ports: [nodeport]
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  ports: [port2]
},
];
var connectors = {
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    type: 'Orthogonal',
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    },
    sourceID: 'node',
    targetID: 'node1',
    sourcePortID: 'port',
    targetPortID: 'port1'
}
var diagram = new ej.diagrams.Diagram({
    width: 900,
    height: 900,
    nodes: nodes,
    connectors: [connectors],
    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>

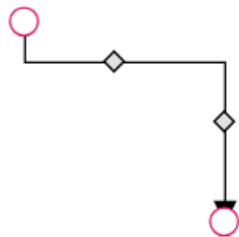
```

How to customize Orthogonal Segment Thumb Shape

The orthogonal connector can have any number of segments in between the source and the target point. Segments are rendered with the rhombus shape by default. The [segmentThumbShape](#) property allows you to change the default shape of the segment thumb. The following predefined shapes are provided:

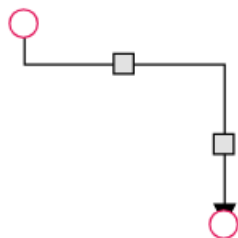
| Shape name | Shape |

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



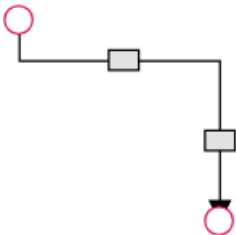
| Rhombus |

|

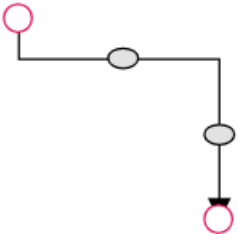


| Square |

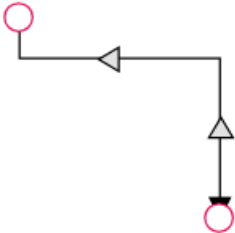
|



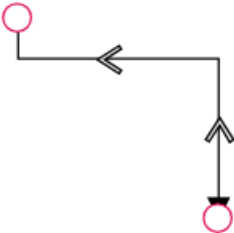
| Rectangle |



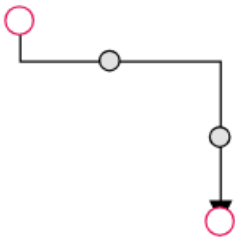
| Ellipse |



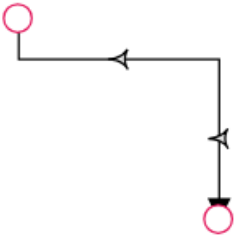
| Arrow |



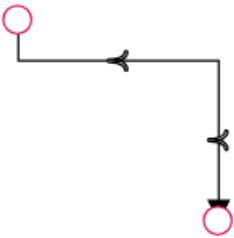
| OpenArrow |



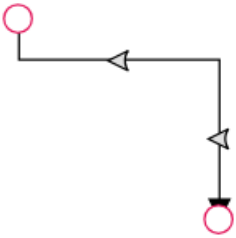
| Circle |



| Fletch |

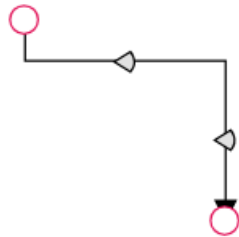


| OpenFetch |



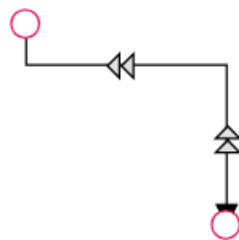
| IndentedArrow |





| OutdentedArrow |

|



| DoubleArrow |

|

You can customize the style of the thumb shape by overriding the class e-orthogonal-thumb.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var connector1 = {
  id: 'connector1',
  type: 'Orthogonal',
  sourcePoint: {x: 250, y: 250},
  targetPoint: {x: 350, y: 350},
  segments: [
    {
      type: 'Orthogonal',
      direction: "Right",
      length: 70
    },
    {
      type: 'Orthogonal',
      direction: "Bottom",
      length: 20
    }
  ],
  constraints: ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.DragSegmentThumb
};
var diagram = new ej.diagrams.Diagram({
  width: 1500, height: 1000,
  connectors: [connector1]
}, '#element');
```

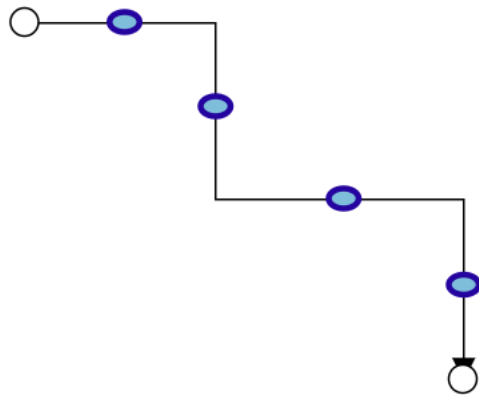
INDEX.HTML

```
<!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>
```



Use the following CSS to customize the segment thumb shape.

```
`scss
.e-orthogonal-thumb {
stroke:#24039e;
fill:rgb(126, 190, 219);
stroke-width: 3px;
}
`
```

Bezier

Bezier segments are used to create curve segments and the curves are configurable either with the control points or with vectors.

To create a bezier segment, the [segment.type](#) is set as `bezier` and need to specify [type](#) for the connector. The following code example illustrates how to create a default bezier segment.

INDEX.JS

```
var connectors = [
  {
    id: 'connector1',
    type: 'Bezier',
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        },
    },
    segments: [{
        type: 'Bezier',
    }],
    sourcePoint: {
        x: 50,
        y: 100
    },
    targetPoint: {
        x: 150,
        y: 200
    },
},
],
];
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: connectors
}, '#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 [point1](#) and [point2](#) properties of bezier segment enable you to set the control points. The following code example illustrates how to configure the bezier segments with control points.

INDEX.JS

```
var connectors = [
    {
        id: 'connector3',
        type: 'Bezier',
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7',
            strokeWidth: 2
        },
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        },
        segments: [{
            type: 'Bezier',
            point1: {
                x: 100,
                y: 100
            },
            point2: {
                x: 200,
                y: 200
            }
        }],
        sourcePoint: {
            x: 100,
            y: 200
        },
        targetPoint: {
            x: 200,
            y: 100
        }
    },
];
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    connectors: connectors
}, '#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 [vector1](#) and [vector2](#) properties of bezier segment enable you to define the vectors. The following code illustrates how to configure a bezier curve with vectors.

INDEX.JS

```
var connectors = [
  {
    id: 'connector2',
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
  },
];
```

```

    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    type: 'Bezier',
    segments: [{
      type: 'Bezier',
      vector1: {
        distance: 100,
        angle: 90
      },
      vector2: {
        distance: 45,
        angle: 270
      }
    }],
    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    }
  },
];
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: connectors
}, '#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>

```

Avoid overlapping with bezier

By default, when there are no segments defined for a bezier connector, the bezier segments will be created automatically and routed in such a way that avoids overlapping with the source and target nodes.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var nodes = [{
    id: 'Start',
    offsetX: 250,
    offsetY: 150,
    annotations: [{ content: 'Start' }]
},
{
    id: 'End',
    offsetX: 450,
    offsetY: 200,
    annotations: [{ content: 'End' }]
}];
let connectors = [{
    id: "connector1",
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: { shape: 'None' },
    // ID of the source and target nodes
    sourceID: "Start",
    sourcePortID: "StartPort",
    targetID: "End",
    targetPortID: "EndPort",
    type: 'Bezier',
}];
var diagram = new ej.diagrams.Diagram({

```

```

width: '100%',
height: '600px',
nodes: nodes,
connectors: connectors,
// Defines the default properties for the node
getNodeDefaults: (node) => {
    node.height = 100;
    node.width = 100;
    node.shape = { type: 'Basic', shape: 'Rectangle' }
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
},
getConnectorDefaults: (connector) => {
    connector.constraints = ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.DragSegmentThumb;
    return connector;
}
});
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>

```

Also, the intermediate point of two adjacent bezier segments can be edited interactively based on the `bezierSettings.segmentEditOrientation` property of the connector class.

How to interact with the bezier segments efficiently

While interacting with multiple bezier segments, maintain their control points at the same distance and angle by using the `bezierSettings.smoothness` property of the connector class.

| BezierSmoothness value | Description |

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

| SymmetricDistance | Both control points of adjacent segments will be at the same distance when any one of them is editing. |

| SymmetricAngle | Both control points of adjacent segments will be at the same angle when any one of them is editing. |

| Default | Both control points of adjacent segments will be at the same angle and same distance when any one of them is editing. |

| None | Segment's control points are interacted independently from each other. |

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var nodes = [{
    id: 'Start',
    offsetX: 250,
    offsetY: 150,
    annotations: [{ content: 'Start' }],
    ports: [{
        id: 'StartPort',
        visibility: ej.diagrams.PortVisibility.Visible,
        shape: 'Circle',
        offset: { x: 1, y: 0.5 },
        style: { strokeColor: '#366F8C', fill: '#366F8C' }
    }]
},
{
    id: 'End',
    offsetX: 450,
    offsetY: 200,
    annotations: [{ content: 'End' }],
    ports: [{
        id: 'EndPort',
        visibility: ej.diagrams.PortVisibility.Visible,
        shape: 'Circle',
        offset: { x: 0, y: 0.5 },
        style: { strokeColor: '#366F8C', fill: '#366F8C' }
    }]
}]

```

```

    }];
    let connectors = [{
        id: "connector1",
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7',
            strokeWidth: 2
        },
        targetDecorator: { shape: 'None' },
        // ID of the source and target nodes
        sourceID: "Start",
        sourcePortID: "StartPort",
        targetID: "End",
        targetPortID: "EndPort",
        type: 'Bezier',

        // Configuring settings for bezier interactions
        bezierSettings : { smoothness:
ej.diagrams.BezierSmoothness.SymmetricAngle }
    }];
    var diagram = new ej.diagrams.Diagram({
        width: '100%',
        height: '600px',
        nodes: nodes,
        connectors: connectors,
        // Defines the default properties for the node
        getNodeDefaults: (node) => {
            node.height = 100;
            node.width = 100;
            node.shape = { type: 'Basic', shape: 'Rectangle' }
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
        },
        getConnectorDefaults: (connector) => {
            connector.constraints = ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.DragSegmentThumb;
            return connector;
        }
    });
    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>

```

Decorator

- Starting and ending points of a connector can be decorated with some customizable shapes like arrows, circles, diamond, or path. The connection end points can be decorated with the [sourceDecorator](#) and [targetDecorator](#) properties of the connector.
- The [shape](#) property of [sourceDecorator](#) allows to define the shape of the decorators. Similarly, the [shape](#) property of [targetDecorator](#) allows to define the shape of the decorators.
- To create custom shape for source decorator, use [pathData](#) property. Similarly, to create custom shape for target decorator, use [pathData](#) property.
- The following code example illustrates how to create decorators of various shapes.

INDEX.JS

```

var connectors = {
    id: "connector1",
    type: 'Straight',
    // Decorator shape- circle
    sourceDecorator: {
        shape: 'Circle',
        // Defines the style for the sourceDecorator
        style: {
            // Defines the strokeWidth for the sourceDecorator
            strokeWidth: 3,
            // Defines the strokeColor for the sourceDecorator
            strokeColor: 'red'
        }
    },

```

```

    },
    // Decorator shape - Diamond
    targetDecorator: {
        // Defines the custom shape for the connector's target decorator
        shape: 'Custom',
        // Defines the path for the connector's target decorator
        pathData: 'M80.5,12.5 C80.5,19.127417 62.59139,24.5 40.5,24.5
C18.40861,24.5 0.5,19.127417 0.5,12.5' +
        'C0.5,5.872583 18.40861,0.5 40.5,0.5 C62.59139,0.5 80.5,5.872583
80.5,12.5 z',
        // defines the style for the target decorator
        style: {
            // Defines the strokeWidth for the targetDecorator
            strokeWidth: 3,
            // Defines the strokeColor for the sourceDecorator
            strokeColor: 'green',
            // Defines the opacity for the sourceDecorator
            opacity: .8
        },
    },
},
sourcePoint: {
    x: 100,
    y: 100
},
targetPoint: {
    x: 200,
    y: 200
}
};
var connectors2 = {
    id: "connectors2",
    type: 'Straight',
    // Decorator shape - IndentedArrow
    sourceDecorator: {
        shape: 'IndentedArrow',
        style: {
            strokeWidth: 3,
            strokeColor: 'blue'
        },
    },
},
// Decorator shape - OutdentedArrow
targetDecorator: {
    shape: 'OutdentedArrow',
    style: {
        strokeWidth: 3,
        strokeColor: 'yellow'
    },
},
sourcePoint: {
    x: 400,
    y: 100
},
targetPoint: {
    x: 300,
    y: 200
}
}
}

```

```
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: [connectors, connectors2],
}, '#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>
```

Padding

Padding is used to leave the space between the Connector's end point and the object to where it is connected.

- The [sourcePadding](#) property of connector defines space between the source point and the source node of the connector.

- The [targetPadding](#) property of connector defines space between the end point and the target node of the connector.
- The following code example illustrates how to leave space between the connection end points and source and target nodes.

INDEX.JS

```
var nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
}
];
var connector = {
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourceID: 'node',
  targetID: 'node1',
  // Set Source Padding value
  sourcePadding:20,
  // Set Target Padding value
  targetPadding:20
};
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '350px',
  getNodeDefaults: function(node) {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  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>

```

Hit padding

- The [hitPadding](#) property enables you to define the clickable area around the connector path. The default value for hitPadding is 10.
- The following code example illustrates how to specify hit padding for connector.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.ConnectorEditing);
var connectors = [{
  id: "connector1",
  type: "Orthogonal",
  hitPadding: 40,

```

```

    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    constraints: ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.DragSegmentThumb
  }
]
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: connectors,
}, '#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>
```

Flip

The diagram Provides support to flip the connector. The [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip

[Horizontal](#) is used to interchange the connector source and target x points.

- VerticalFlip

[Vertical](#) is used to interchange the connector source and target y points.

- Both

[Both](#) is used to interchange the source point as target point and target point as source point

.

INDEX.JS

```
var nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
}
];
var connector = {
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourceID: 'node',
    targetID: 'node1',
    // Set Source Padding value
    sourcePadding:20,
    // Set Target Padding value
    targetPadding:20
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '350px',
    getNodeDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    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 flip is not applicable when the connectors connect in nodes

Bridging

Line bridging creates a bridge for lines to smartly cross over the other lines, at points of intersection. By default, [bridgeDirection](#) is set to top. Depending upon the direction given bridging direction appears.

Bridging can be enabled/disabled either with the `connector.constraints` or `diagram.constraints`. The following code example illustrates how to enable line bridging.

INDEX.JS

```
var node1 = {
  id: 'Transaction',
  width: 150,
  height: 60,
  offsetX: 300,
  offsetY: 60,
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  },
  annotations: [{
    id: 'label1',
    content: 'Start Transaction',
    offset: {
      x: 0.5,
      y: 0.5
    }
  }],
};
var node2 = {
  id: 'Verification',
  width: 150,
  height: 60,
  offsetX: 300,
  offsetY: 250,
  shape: {
    type: 'Flow',
    shape: 'Process'
  },
  annotations: [{
    id: 'label2',
    content: 'Verification',
    offset: {
```

```
        x: 0.5,
        y: 0.5
    }
    ]]
};
var connectors = {
    id: 'connector1',
    type: 'Straight',
    sourceID: 'Transaction',
    targetID: 'Verification'
};
var connectors2 = {
    id: 'connector2',
    type: 'Straight',
    sourcePoint: {
        x: 200,
        y: 130
    },
    targetPoint: {
        x: 400,
        y: 130
    }
};
var connector3 = {
    id: 'connector3',
    type: 'Straight',
    sourcePoint: {
        x: 200,
        y: 170
    },
    targetPoint: {
        x: 400,
        y: 170
    }
};
//Enables bridging for every connector added in the model
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    getConnectorDefaults: (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;
    },
    getNodeDefaults: (node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    nodes: [node1, node2],
    height: '600px',
    //enables the bridging constraints for the connector
```

```
constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.Bridging,
connectors: [connectors, connectors2, connector3]
}, '#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: You need to inject connector bridging module into the diagram.

The [bridgeSpace](#) property of connectors can be used to define the width for line bridging.

Limitation: Bezier segments do not support bridging.

Corner radius

Corner radius allows to create connectors with rounded corners. The radius of the rounded corner is set with the [cornerRadius](#) property.

INDEX.JS

```
var nodes = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
},
{
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 350,
},
]
var connectors = [{
    id: "connector1",
    type: 'Orthogonal',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    cornerRadius: 10,
    sourceID: 'node1',
    targetID: 'node2',
    segments: [{
        type: 'Orthogonal',
        direction: 'Right',
        length: 50
    }],
}],
}]
var diagram = new ej.diagrams.Diagram({
    width: 1500,
    height: 1500,
    getConnectorDefaults: function(node) {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    connectors: connectors,
    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>
```

Appearance

- The connector's [strokeWidth](#), [strokeColor](#), [strokeDashArray](#), and [opacity](#) properties are used to customize the appearance of the connector segments.
- The [visible](#) property of the connector enables or disables the visibility of connector.
- Default values for all the connectors can be set using the `getConnectorDefaults` properties. For example, if all connectors have the same type or having the same property then such properties can be moved into `getConnectorDefaults`.

Segment appearance

The following code example illustrates how to customize the segment appearance.

INDEX.JS

```
var connectors = [{
  id: "connector1",
  targetDecorator: {
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    }
  },
  style: {
    // Stroke color
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    // Stroke width of the line
    strokeWidth: 2,
    // Line style
    strokeDashArray: '2,2'
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  segments: [{
    type: 'Orthogonal',
    direction: 'Right',
    length: 50
  }],
},
{
  id: "connector2",
  // Set the visibility of the connector to false
  visible: false,
  targetDecorator: {
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    }
  },
  style: {
    // Stroke color
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    // Stroke width of the line
    strokeWidth: 2,
    // Line style
    strokeDashArray: '2,2'
  },
},
]
```

```

        sourcePoint: {
            x: 300,
            y: 300
        },
        targetPoint: {
            x: 400,
            y: 400
        },
        segments: [{
            type: 'Orthogonal',
            direction: 'Right',
            length: 50
        }],
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '600px',
    // Define the default values for all the connector. Similary we can add
    all the properties
    getConnectorDefaults: (obj) => {
        obj.type = 'Orthogonal'
        return obj;
    },
    connectors: connectors,
}, '#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>
```

Decorator appearance

- The source decorator's [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#) properties are used to customize the color, width, and appearance of the decorator.
- To set the border stroke color, stroke width, and stroke dash array for the target decorator, use [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#).
- To set the size for source and target decorator, use width and height property.

The following code example illustrates how to customize the appearance of the decorator.

INDEX.JS

```
var connectors = [{
  id: "connector1",
  type: 'Straight',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  bridgeSpace: 20,
  // Customize the target decorator
  targetDecorator: {
    style: {
      // Fill color of the decorator
      fill: '#6BA5D7',
      // Stroke color of the decorator
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
```

```
connectors: connectors,
}, '#element');
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>
```

Interaction

- Diagram allows to edit the connectors at runtime. To edit the connector segments at runtime, refer to [Connection Editing](#).

Automatic line routing

Diagram provides additional flexibility to re-route the diagram connectors. A connector will frequently re-route itself when a shape moves next to it. The following screenshot illustrates how the connector automatically re-routes the segments.

- Dependency LineRouting module should be injected to the application as the following code snippet.

```
`javascript
```

```
/
```

- Injecting the automatic line routing module.

```
*/
```

```
ej.diagrams.Diagram.Inject(ej.diagrams.LineRouting);
```

```
,
```

- Now, the line routing constraints must be included to the default diagram constraints to enable automatic line routing support like below.

```
`javascript
```

```
/
```

- Initialize the Diagram

```
*/
```

```
var diagram = new ej.diagrams.Diagram({
```

```
//Add Line routing constraints to diagram.
```

```
constraints: ej.diagrams.DiagramConstraints.Default |
```

```
ej.diagrams.DiagramConstraints.LineRouting
```

```
});
```

```
diagram.appendTo('#diagram');
```

```
,
```

- The following code block shows how to create the diagram with specifying nodes, connectors, constraints, and necessary modules for line routing.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.LineRouting);  
var diagram;  
var nodes = [  
  { id: 'shape1', offsetX: 100, offsetY: 100, width: 120, height: 50 },
```

```

    { id: 'shape2', offsetX: 300, offsetY: 300, width: 120, height: 50 },
    { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 }
  ];
  var connectors = [
    { id: 'connector', sourceID: 'shape1', targetID: 'shape2', type:
    'Orthogonal' }
  ];

  var diagram = new ej.diagrams.Diagram({
    width: '100%', height: 900, nodes: nodes, connectors: connectors,

    constraints: ej.diagrams.DiagramConstraints.Default |
    ej.diagrams.DiagramConstraints.LineRouting,
    getNodeDefaults: function (node) {
      node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } }
      return node;
    },
    snapSettings: {
      constraints: ej.diagrams.SnapConstraints.None,
    }
  });
  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>
```

- In some situations, automatic line routing enabled diagram needs to ignore a specific connector from automatic line routing. So, in this case, auto routing feature can be disabled to the specific connector using the [constraints](#) property of the connector like the following code snippet.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.LineRouting);
var diagram;
var nodes = [
  { id: 'shape1', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 350, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 },
  { id: 'shape4', offsetX: 300, offsetY: 200, width: 120, height: 50 }
];
var connectors = [
  { id: 'connector', sourceID: 'shape1', targetID: 'shape2', type:
'Orthogonal', annotations: [{offset:.7, content: ' Routing \n
enabled', style:{fill:"white"}}]},
  { id: 'connector2', sourceID: 'shape1', targetID:
'shape2', annotations: [{offset:.6, content: ' Routing \n
disabled', style:{fill:"white"}}], type: 'Orthogonal', constraints:
ej.diagrams.ConnectorConstraints.Default
&~ej.diagrams.ConnectorConstraints.InheritLineRouting }
];
/**
 * Initialize the Diagram
 */
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: 900, nodes: nodes, connectors: connectors,
  //Add Line routing constraints to diagram.
  constraints: ej.diagrams.DiagramConstraints.Default |
ej.diagrams.DiagramConstraints.LineRouting,
  getNodeDefaults: function (node) {
    node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } }
    return node;
  },
  snapSettings: {
    constraints: ej.diagrams.SnapConstraints.None,
  }
});
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>

```

Constraints

- The [constraints](#) property of connector allows to enable/disable certain features of connectors.
- To enable or disable the constraints, refer [constraints](#).

The following code illustrates how to disable selection.

INDEX.JS

```

var connectors = {
  id: "connector1",
  constraints: ej.diagrams.ConnectorConstraints.Default &
~ej.diagrams.ConnectorConstraints.Select,
  type: 'Straight',
  style: {

```

```

        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
};
var diagram = new ej.diagrams.Diagram({
    width: 1500,
    height: 1500,
    connectors: [connectors]
}, '#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>
```

Custom properties

- The [addInfo](#) property of connectors allow you to maintain additional information to the connectors.

```
`javascript
var connectors = {
id: 'connector1',
// Defines the information about the connector
addInfo:'centralconnector',
type: 'Straight',
sourceID: 'Transaction',
targetID: 'Verification'
};
`
```

Stack order

The connectors [zIndex](#) property specifies the stack order of the connector. A connector with greater stack order is always in front of a connector with a lower stack order.

The following code illustrates how to render connector based on the stack order.

INDEX.JS

```
var connectors = {
    id: 'connector1',
    addInfo: 'connec',
    zIndex: 2,
    type: 'Straight',
    sourcePoint: {
        x: 300,
        y: 100
    },
    targetPoint: {
        x: 300,
        y: 200
    }
};
```

```

var connectors2 = {
  id: 'connector2',
  type: 'Straight',
  zIndex: 1,
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
};
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  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;
  },
  height: '600px',
  connectors: [ connectors, connectors2]
}, '#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>

```

Enable Connector Splitting

The connectors are used to create a link between two points, ports, or nodes to represent the relationship between them. Split the connector between two nodes when dropping a new node onto an existing connector and create a connection between the new node and existing nodes by setting the [enableConnectorSplit](#) as true. The default value of the [enableConnectorSplit](#) is false

The following code illustrates how to split the connector and create a connection with new node.

INDEX.JS

```

var node1 = {
  id: 'node1',
  width: 100, height: 100,
  offsetX: 150, offsetY: 150,
  annotations: [{ content: 'node1' } ] ,
};
var node2= {
  id: 'node2',
  width: 100, height: 100,
  offsetX: 650, offsetY: 150,
  annotations: [{ content: 'node2' } ] ,
};
var node3 = {
  id: 'node3',
  width: 100, height: 100,
  offsetX: 490, offsetY: 290,
  annotations: [{ content: 'node3' } ] ,
};
var connector1 = {
  id: 'connector1',sourceID:"node1",targetID:"node2",
  constraints: ej.diagrams.ConnectorConstraints.Default |
ej.diagrams.ConnectorConstraints.AllowDrop
};
var diagram = new ej.diagrams.Diagram({
  width: 1500, height: 1000,
  enableConnectorSplit:true,
  nodes: [node1, node2, node3],
  connectors: [connector1]
}, '#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>
```

![Enable Connector Split](../images/EnableSplit.gif)

See Also

- [How to add annotations to the connector](#)
- [How to enable/disable the behavior of the node](#)
- [How to add connectors to the symbol palette](#)
- [How to perform the interaction on the connector](#)
- [How to create diagram connectors using drawing tools](#)

Group in EJ2 JavaScript 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.TS

```
import {Diagram,NodeModel,} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle1'
    }]
}, {
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle2'
    }]
},
{
    id: "rectangle3",
    offsetX: 400,
    offsetY: 300,
    width: 100,
    height: 100,
    style: { fill: 'darkCyan', strokeWidth: 2 },
    annotations: [{ content: 'rectangle2' }]
}
// Grouping node 1 and node 2 into a single group
{
    id: 'group',
    children: ['rectangle1', 'rectangle2'],
    padding:{left:10,right:10,top:10,bottom:10}
},
];
let diagram: Diagram = new Diagram({
    width: '100%',
```

```

    height: '600px',
    nodes: nodes,
    getNodeDefaults: (node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
  });
  diagram.appendTo('#element');
  diagram.selectAll();
  // Adding the third node into the existing group
  diagram.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>

```


The following code illustrates how a ungroup at runtime.

INDEX.TS

```
import {Diagram,NodeModel,} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle1'
    }]
}, {
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle2'
    }]
}, {
    id: 'group',
    children: ['rectangle1', 'rectangle2']
}];
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    nodes: nodes,
    getNodeDefaults: (node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
});
diagram.appendTo('#element');
diagram.selectAll();
// Ungroup the selected group into nodes
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.TS

```

import {Diagram,NodeModel,} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle1'
    }]
}, {
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,

```

```
        annotations: [{
            content: 'rectangle2'
        }]
    }, ];
let group: NodeModel = {
    id: 'group2',
    children: ['rectangle1', 'rectangle2']
};
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    getNodeDefaults: (node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    },
    nodes: nodes,
});
diagram.appendTo('#element');
// Add the group into the diagram
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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  id: 'node1', width: 150, height: 100, offsetX: 100, offsetY: 100,
  annotations: [{ content: 'Node1' }]
};
var node2: NodeModel = {
  id: 'node2', width: 80, height: 130, offsetX: 200, offsetY: 200,
  annotations: [{ content: 'Node2' }]
};
var group: NodeModel = {
  id: 'group1', children: ['node1', 'node2']
};
var node3: NodeModel = {
  id: 'node3', width: 100, height: 100, offsetX: 300, offsetY: 300,
  annotations: [{ content: 'Node3' }]
};
let diagram: Diagram = new Diagram({
```

```

width: '100%',
height: 900,
nodes: [node, node2, node3, group],
getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
}
});
diagram.appendTo('#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.TS

```
import {Diagram, DiagramElement, Canvas} from '@syncfusion/ej2-diagrams';
let diagram: Diagram;
let canvas: Canvas;
let child1: DiagramElement;
let child2: DiagramElement;
canvas = new Canvas();
canvas.pivot = {
    x: 0,
    y: 0
};
canvas.offsetX = 75;
canvas.offsetY = 75;
canvas.style.fill = '#6BA5D7';
child1 = new DiagramElement();
child1.width = 100;
child1.height = 100;
child1.margin.left = child1.margin.top = 10;
child2 = new DiagramElement();
child2.width = 100;
child2.height = 100;
child2.margin.left = 190;
child2.margin.top = 190;
canvas.children = [child1, child2];
diagram = new Diagram({
    mode: 'SVG',
    width: '100%',
    height: '600px',
    // Defines the basic elements
    basicElements: [canvas]
});
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>
```

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.TS

```
import {Diagram,NodeModel,StackPanel,TextElement,} from '@syncfusion/ej2-
diagrams';
```

```

let nodes: NodeModel[] = [{
  id: 'node5',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  annotations: [{
    content: 'Custom Template',
    offset: {
      y: 1
    },
    verticalAlignment: 'Top'
  }]
}, ];

let getTextElement: Function = (text: string) => {
  let textElement: TextElement = new TextElement();
  textElement.width = 50;
  textElement.height = 20;
  textElement.content = text;
  return textElement;
};

let addRows: Function = (column: StackPanel) => {
  column.children.push(getTextElement('Row1'));
  column.children.push(getTextElement('Row2'));
  column.children.push(getTextElement('Row3'));
  column.children.push(getTextElement('Row4'));
};

let diagram: Diagram = new Diagram({
  width: '100%',
  height: 900,
  nodes: nodes,
  getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  setNodeTemplate: (obj: NodeModel, diagram: Diagram): StackPanel => {
    if (obj.id.indexOf('node5') !== -1) {
      // It will be replaced with grid panel
      let table: StackPanel = new StackPanel();
      table.orientation = 'Horizontal';
      table.padding.left
      let column1: StackPanel = new StackPanel();
      column1.children = [];
      column1.children.push(getTextElement('Column1'));
      addRows(column1);
      let column2: StackPanel = new StackPanel();
      column2.children = [];
      column2.children.push(getTextElement('Column2'));
      addRows(column2);
      table.children = [column1, column2];
    }
  }
});

```



```

        return table;
    }
    return null;
},
});
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>
```

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 EJ2 JavaScript 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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let model: NodeModel = {
  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
    let diagram: Diagram = new Diagram({
        width: '100%',
        height: '600px',
        // Add node
        nodes: [node1]
    });
    // 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>
```

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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node1: NodeModel = {
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Initialize 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
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  // Add node
  nodes: [node1]
});
// 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>

```

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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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
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>
</body>
<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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  // Add node
  nodes: [node]
});

```

```
// render initialized Diagram
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">

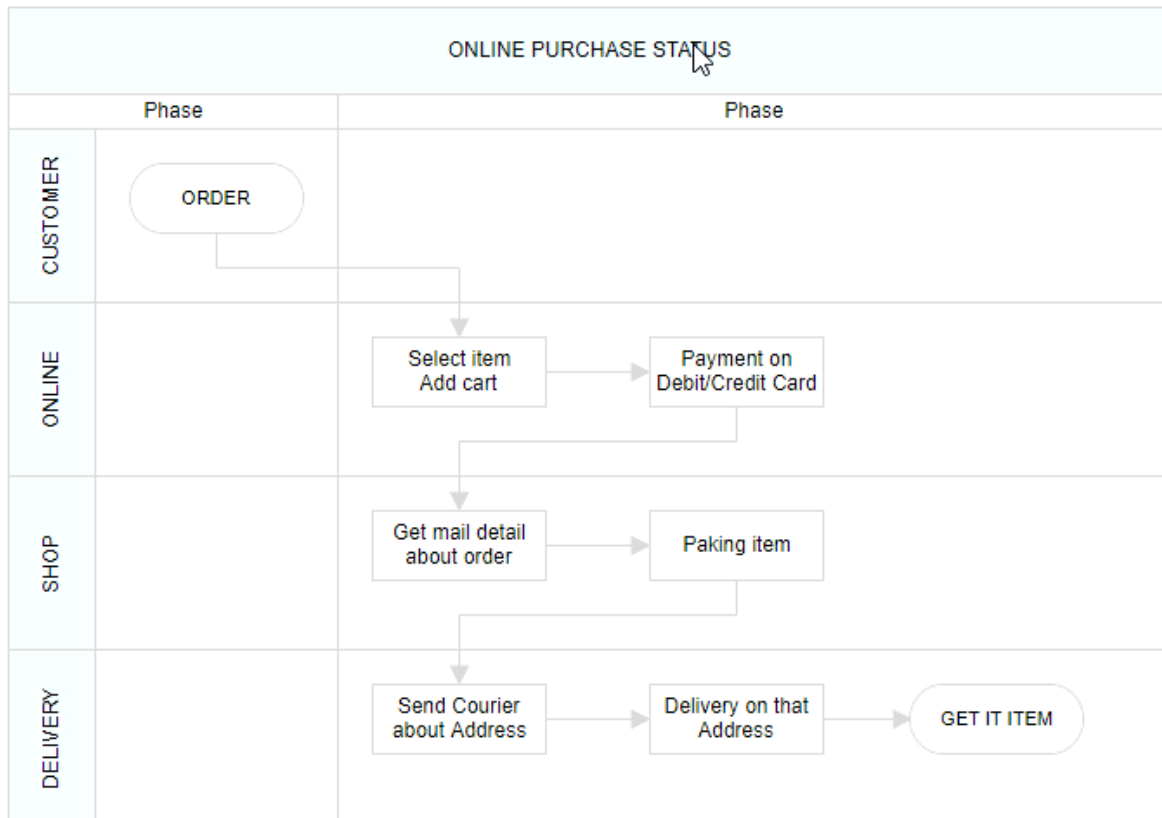
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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,
            // set the lane info
            addInfo:{name:'lane1'}
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// 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>

```

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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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
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>

```

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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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
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>
```

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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');
let lane : nodeModel = 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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  // Add node
  nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');
let 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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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 }
                },
                // 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
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>

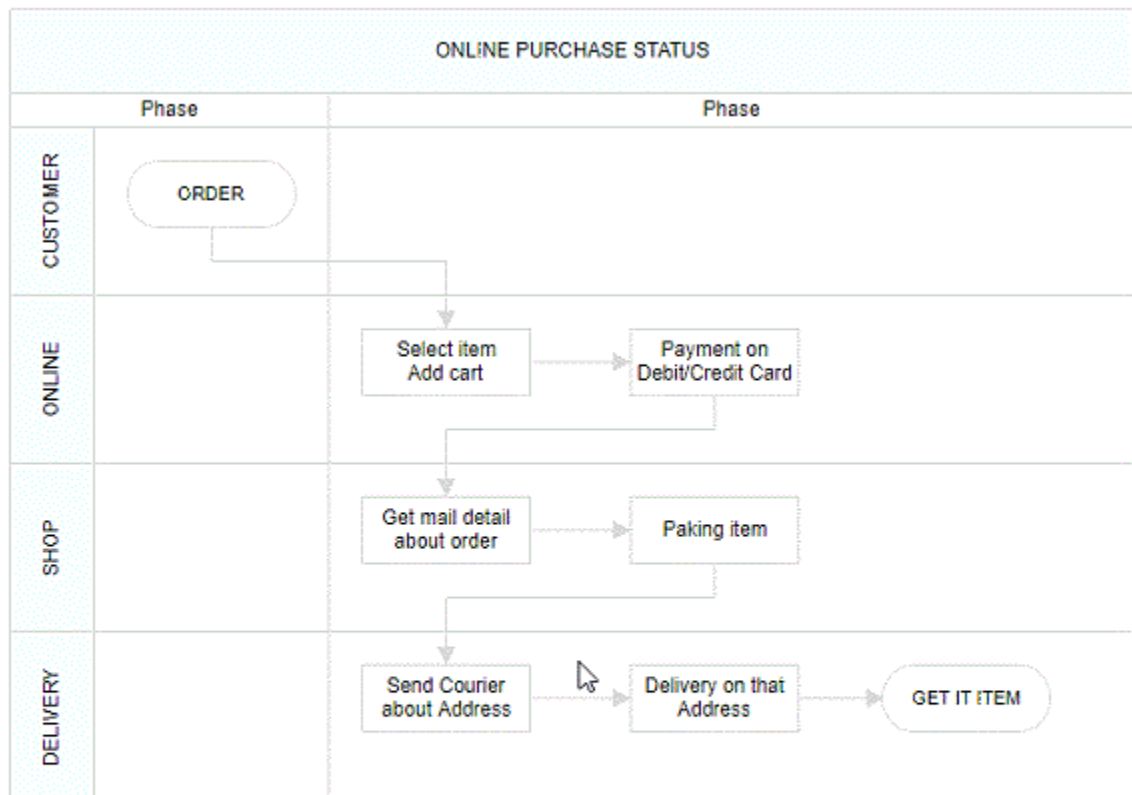
    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>
```

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

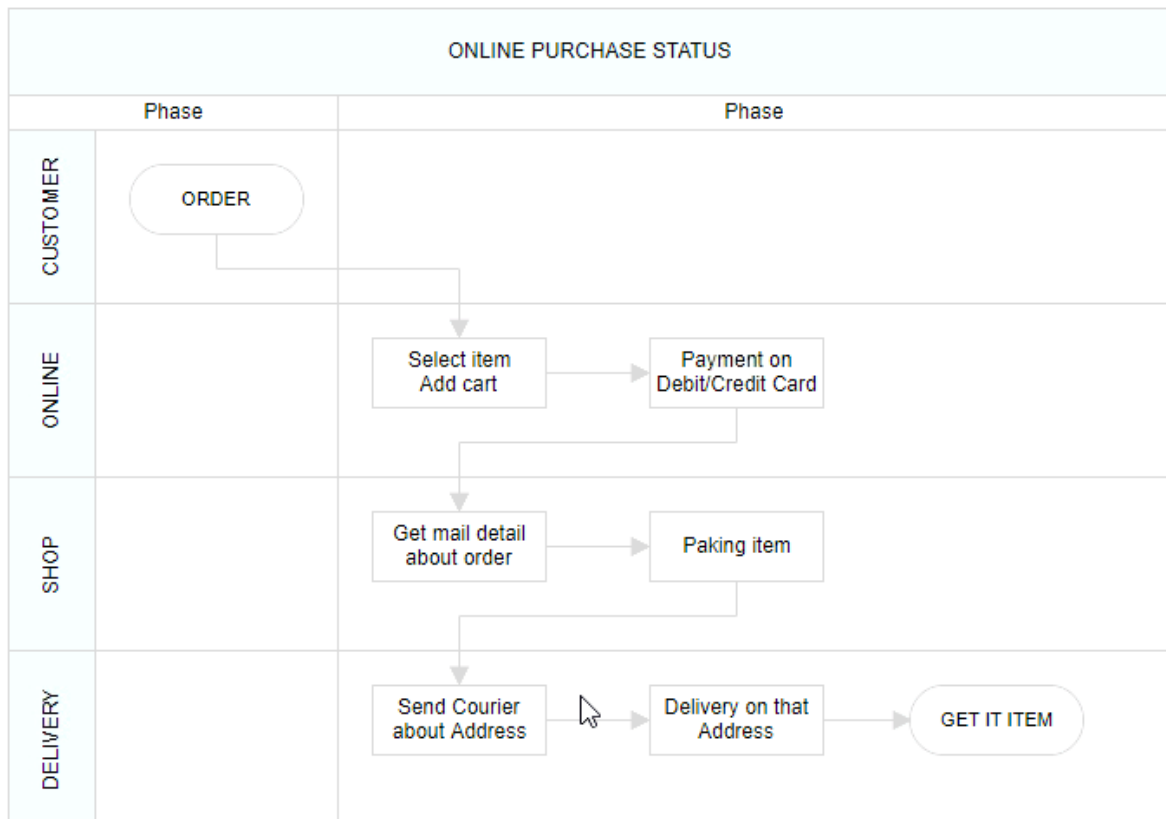
illustrates how resize the lane.



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.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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, fill: 'red' }
    },
    canMove: false,
},
],
phases: [
    {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
    },
],
    phaseSize: 20,
},
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
};
// initialize Diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');
let 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>
```

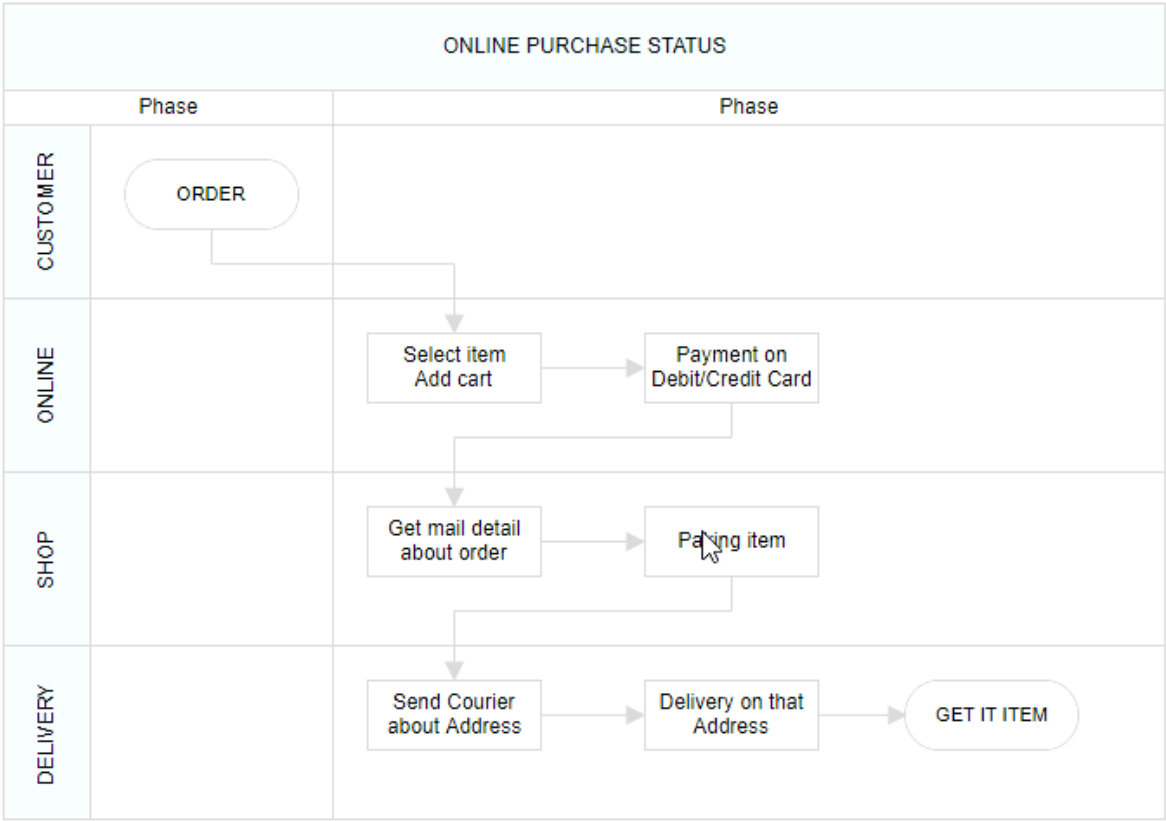
Resize helper

- 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.

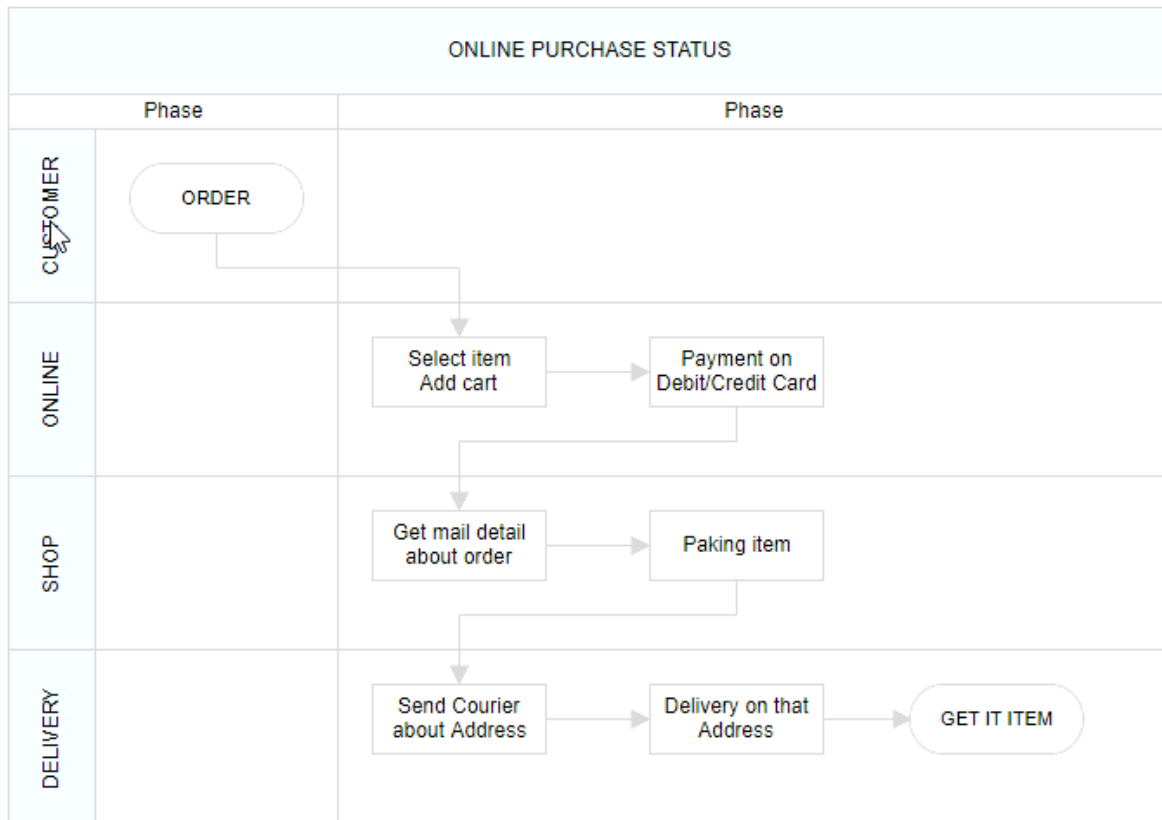
The following image illustrates children interaction in lane.



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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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
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>

```

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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    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
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node
    nodes: [node]
});
// render initialized Diagram
diagram.appendTo('#element');
let 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>
</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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  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: '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: 120,
            // set the phase info
            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
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>

```

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.

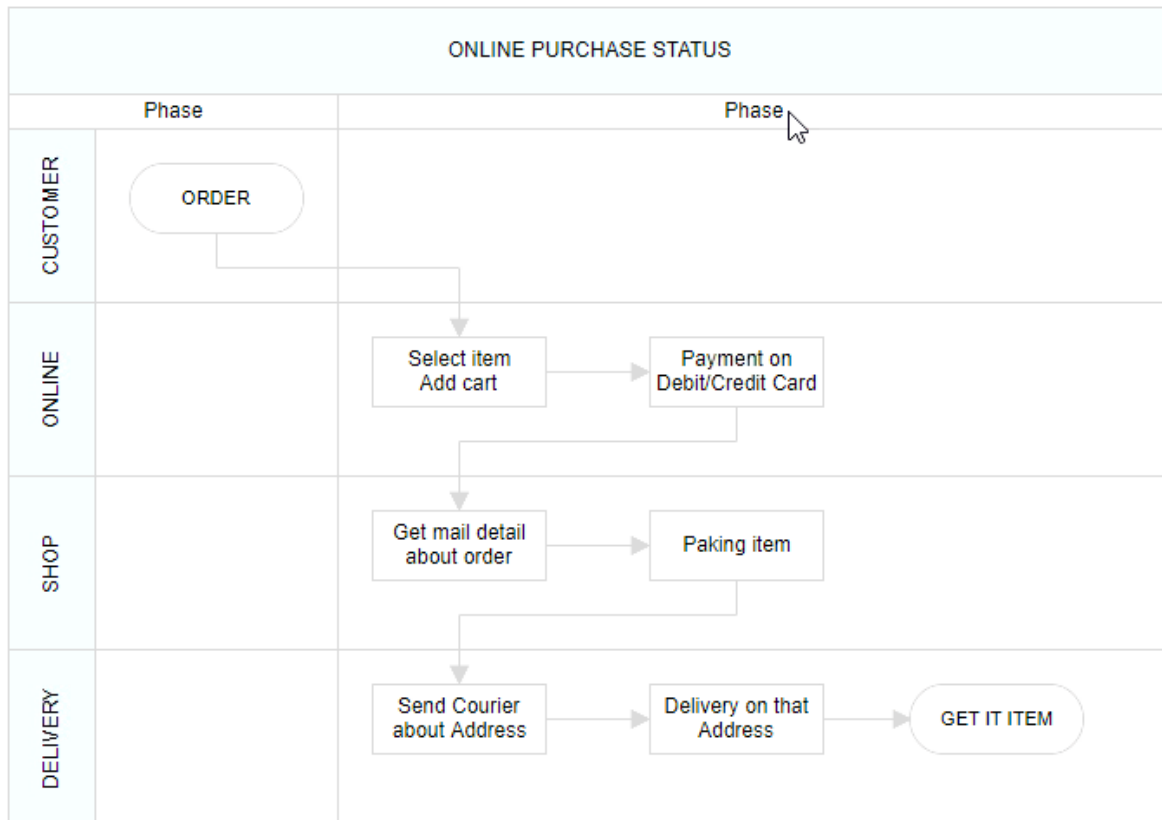
Resizing helper

- 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.TS

```

import {
    Diagram,
    NodeModel,
    SymbolPalette,
    SymbolInfo
} from '@syncfusion/ej2-diagrams';
//Initialize the flowshapes for the symbol palette
export function getswimlaneShapes(): NodeModel[] {
    let swimlaneShapes : NodeModel[] = [
        {
            id: 'stackCanvas1',
            shape: {
                type: 'SwimLane', lanes: [
                    {
                        id: 'lane1',
                        style: { strokeColor: 'black' }, height: 60,
                        header: { width: 50, height: 50, style: {
                            strokeColor: 'black', fontSize: 11 } },
                    }
                ],
            },
            width: 150,
        },
    ],

```

```

        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
}
    ];
    return swimlaneShapes;
}
function setPaletteNodeDefaults(node: NodeModel): void {
    node.width = 100;
    node.height = 100;
    node.style.strokeColor = '#3A3A3A';
}

```

```
//Initializes the symbol palette
let palette: SymbolPalette = new SymbolPalette({
  expandMode: 'Multiple',
  palettes: [{
    id: 'swimlane',
    expanded: true,
    symbols: getswimlaneShapes(),
    title: 'Swimlane Shapes'
  }, ],
  symbolPreview: {
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5
    }
  },
  symbolMargin: {
    left: 12,
    right: 12,
    top: 12,
    bottom: 12
  },
  //Returns the default properties of node
  getNodeDefaults: setPaletteNodeDefaults,
  getSymbolInfo: (symbol: NodeModel): SymbolInfo => {
    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">

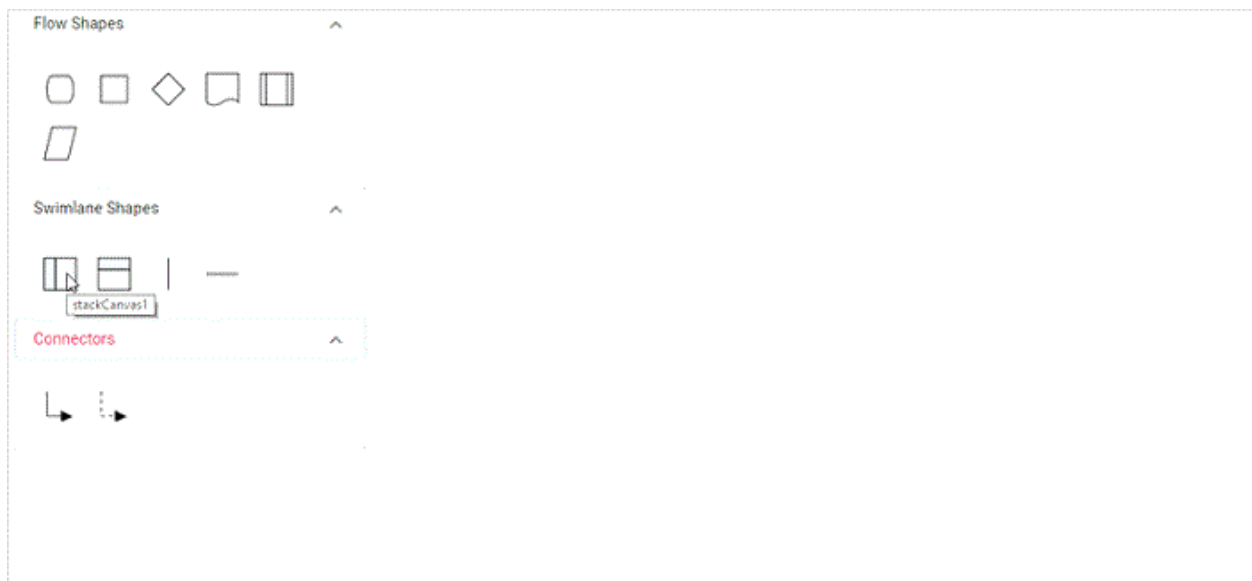
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
  <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
popups/styles/material.css" rel="stylesheet">
  <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 as 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.

Labels in EJ2 JavaScript Diagram control

[Annotation](#) is a block of text that can be displayed over a node or connector. Annotation is used to textually represent an object with a string that can be edited at runtime. Multiple annotations can be added to a node/connector.

<!-- markdownlint-disable MD033 -->

Create annotation

An annotation can be added to a node/connector by defining the annotation object and adding that to the annotation collection of the node/connector. The [content](#) property of annotation defines the text to be displayed. The following code illustrates how to create a annotation.

INDEX.JS

```
var node = {
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  annotations: [{ template: '<div><input type="button"
value="Submit"></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>

    <div id="container">
        <div id="element"></div>
    </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 annotations at runtime

- Annotations can be added at runtime by using the client-side method [addLabels](#). The following code illustrates how to add a annotation to a node.
- The annotation's [ID](#) property is used to define the name of the annotation and its further used to find the annotation at runtime and do any customization.

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' },
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
});
diagram.appendTo('#element');
var annotation = [{ id: 'label1', content: 'Annotation' }];
diagram.addLabels(diagram.nodes[0], annotation);
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>

```

Remove annotation

A collection of annotations can be removed from the node by using client-side method [removeLabels](#). The following code illustrates how to remove a annotation to a node.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    sstyle: { fill: '#6BA5D7', strokeColor: 'white' },
    annotations: [{ content: 'Annotation' }]
};

// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');
diagram.nodes[0].annotations[0].content = 'Updated Annotation';

```

```
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>
```

Update annotation at runtime

You can change any annotation properties at runtime and update it through the client-side method [dataBind](#).

The following code example illustrates how to change the annotation properties.

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
```



```

// Size of the node
width: 100,
height: 100,
sstyle: { fill: '#6BA5D7', strokeColor: 'white' },
annotations: [{ content: 'Annotation' }]
    };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node]
}, '#element');
diagram.nodes[0].annotations[0].content = 'Updated Annotation';
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>

```

Alignment

Annotation can be aligned relative to the node boundaries. It has [margin](#), [offset](#), horizontal, and vertical alignment settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset

The offset property of annotation is used to align the annotations based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Set the size for a nodes annotation by using [width](#) and [height](#) properties.

The following code shows the relationship between the annotation position (black color circle) and offset (fraction values).

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  sstyle: { fill: '#6BA5D7', strokeColor: 'white' },
  annotations: [{ content: 'Annotation' }]
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node]
}, '#element');
diagram.nodes[0].annotations[0].content = 'Updated Annotation';
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>




```

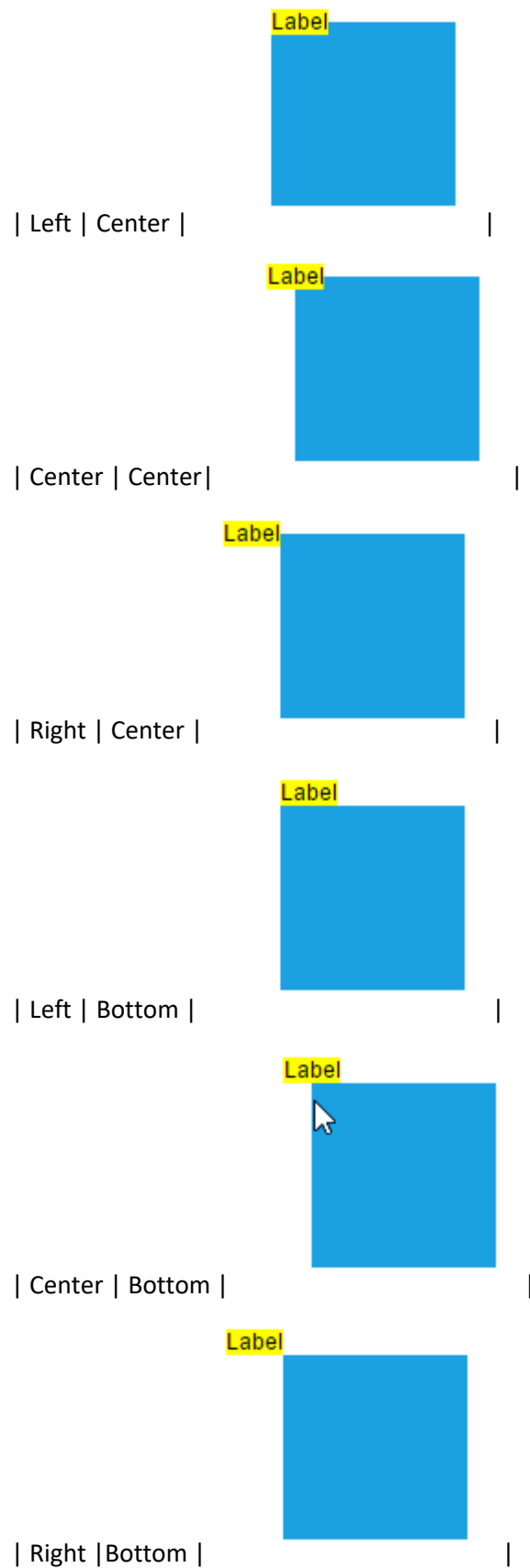
Horizontal and vertical alignment

The [horizontalAlignment](#) property of annotation is used to set how the annotation is horizontally aligned at the annotation position determined from the fraction values. The [verticalAlignment](#) property is used to set how annotation is vertically aligned at the annotation position.

The following tables illustrates all the possible alignments visually with 'offset (0, 0)'.

Horizontal Alignment	Vertical Alignment	Output with Offset(0,0)
Left	Top	
Center	Top	
Right	Top	

Horizontal Alignment	Vertical Alignment	Output with Offset(0,0)
Left	Top	
Center	Top	
Right	Top	



The following codes illustrates how to align annotations.

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' },
  annotations: [{ content: 'Annotation', horizontalAlignment: 'Left' ,
verticalAlignment: 'Top' }]
};
// 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>

```

Annotation alignment with respect to segments

The offset and alignment properties of annotation allows you to align the connector annotations with respect to the segments.

The following code example illustrates how to align connector annotations.

INDEX.JS

```

var node = {
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  annotations: [{ content: 'Task1' }]
};
var node2 = {
  id: 'node2',
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  annotations: [{ content: 'Task2' }]
};
var connector = {
  sourceID: 'node1', targetID: 'node2', type: 'Orthogonal', style: {
    strokeColor: '#6BA5D7', strokeWidth: 2 },
  annotations: [{ content: '0', offset: 0 }, { content: '1', offset: 1}]
};
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '600px', nodes: [node, node2], 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>

```

Margin

[Margin](#) is an absolute value used to add some blank space in any one of its four sides. The annotations can be displaced with the margin property. The following code example illustrates how to align a annotation based on its `offset`, `horizontalAlignment`, `verticalAlignment`, and `margin` values.

INDEX.JS

```

var nodes = [
    {
        id: 'Start', width: 100, height: 100, offsetX: 100, offsetY: 100,
        style: { fill: '#6BA5D7', strokeColor: 'white' },
        annotations: [{ content: 'Task1', margin: { top: 10},
horizontalAlignment: 'Center' , verticalAlignment: 'Top', offset: { x: 0.5,
y: 1} }]}
    ],
];
// initialize Diagram component
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>

```

Text align

The [textAlign](#) property of annotation allows you to set how the text should be aligned (left, right, center, or justify) inside the text block. The following codes illustrate how to set `textAlign` for an annotation.

INDEX.JS

```

var node = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  annotations: [{ content: 'Text align is set as Left', style: {textAlign:
'Left'} }]
}

```



```

    };
    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>

```

Hyperlink

Diagram provides a support to add a [hyperlink](#) for the nodes/connectors annotation. It can also be customized.

A User can open the hyperlink in the new window, the same tab and the new tab by using the [hyperlinkOpenState](#) property

INDEX.JS

```
var node = {
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: { fill: '#6BA5D7', strokeColor: 'white' },
  annotations: [{ hyperlink: { link:
'https://hr.syncfusion.com/home',hyperlinkOpenState:'CurrentTab' } }}
  ];
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>
```

Template Support for Annotation

Diagram provides template support for annotation. you should define a SVG/HTML content as string in the annotation's [template](#) property.

The following code illustrates how to define a template in node's annotation. similarly, you can define it in connectors.

INDEX.JS

```
var node = {
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  annotations: [{ template: '<div><input type="button"
value="Submit"></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>

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

```

Wrapping

When text overflows node boundaries, you can control it by using [text wrapping](#). So, it is wrapped into multiple lines. The wrapping property of annotation defines how the text should be wrapped. The following code illustrates how to wrap a text in a node.

INDEX.JS

```

var node = {
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: { fill: '#6BA5D7', strokeColor: 'white' },
    annotations: [{ content: 'Annotation Text Wrapping', style: {
textWrapping: 'Wrap' } } ]
};
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>

```

Value	Description	Image
-----	-----	-----
		
No Wrap	Text will not be wrapped.	
Wrap	Text-wrapping occurs, when the text overflows beyond the available node width.	

| WrapWithOverflow (Default) | Text-wrapping occurs, when the text overflows beyond the available node width. However, the text may overflow beyond the node width in the case of a very long word. |



Text overflow

The label's [TextOverflow](#) property is used control whether to display the overflowed content in node or not.

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' },
    annotations: [{ content: 'Annotation Text', style: { textOverflow:
'Ellipsis' } }]
};
// 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>

```

Appearance

- You can change the font style of the annotations with the font specific properties (fontSize, fontFamily, color). The following code illustrates how to customize the appearance of the annotation.
- The label's [bold](#), [italic](#), and [textDecoration](#) properties are used to style the label's text.
- The label's [fill](#), [strokeColor](#), and [strokeWidth](#) properties are used to define the background color and border color of the annotation and the [opacity](#) property is used to define the transparency of the annotations.
- The [visible](#) property of the annotation enables or disables the visibility of annotation.

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' },
    annotations: [{ content: 'Annotation Text', style: { color: 'black',
bold: true, italic: true, fontSize: '12', fontFamily: 'TimesNewRoman' } }]
};
// 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 fill, border, and opacity appearances of the text can also be customized with appearance specific properties of annotation. The following code illustrates how to customize background, opacity, and border of the annotation.

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' },
    annotations: [{ content: 'Annotation Text', style: { color: 'black',
fill: 'white', opacity: 0.7 , strokeColor: 'black', strokeWidth: 2 } }]
};
// 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>
```

Interaction

Diagram allows annotation to be interacted by selecting, dragging, rotating, and resizing. Annotation interaction is disabled, by default. You can enable annotation interaction with the [constraints](#) property of annotation. You can also curtail the services of interaction by enabling either selecting, dragging, rotating, or resizing individually with the respective constraints property of annotation. The following code illustrates how to enable annotation interaction.

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' },
annotations: [{ content: 'Annotation Text', constraints:
ej.diagrams.AnnotationConstraints.ReadOnly}]
    };
// 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>

```

Edit

Diagram provides support to edit an annotation at runtime, either programmatically or interactively. By default, annotation is in view mode. But it can be brought to edit mode in two ways;

- Programmatically

By using [startTextEdit](#) method, edit the text through programmatically.

- Interactively
 1. By double-clicking the annotation.
 2. By selecting the item and pressing the F2 key.

Double-clicking any annotation will enables editing and the node enables first annotation editing. When the focus of editor is lost, the annotation for the node is updated. When you double-click on the node/connector/diagram model, the [doubleClick](#) event gets triggered.

Read-only annotations

Diagram allows to create read-only annotations. You have to set the read-only property of annotation to enable/disable the read-only [constraints](#). The following code illustrates how to enable read-only mode.

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' },
  annotations: [{ content: 'Annotation Text', constraints:
ej.diagrams.AnnotationConstraints.ReadOnly}]
};
// 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>

```

Drag Limit

- The diagram control now supports defining the [dragLimit](#) to the label while dragging from the connector and also update the position to the nearest segment offset.
- You can set the value to dragLimit [left](#), [right](#), [top](#), and [bottom](#) properties which allow the dragging of connector labels to a certain limit based on the user defined values.
- By default, drag limit will be disabled for the connector. It can be enabled by setting connector constraints as drag.
- The following code illustrates how to set a dragLimit for connector annotations.

INDEX.JS

```

var connector = {
    id: 'connector', type: 'Orthogonal',
    sourcePoint: { x: 200, y: 200 },
    targetPoint: { x: 300, y: 300 },
    annotations: [
        {
            content: 'connector1', offset: 0.5,
            constraints: ej.diagrams.AnnotationConstraints.Interaction |
ej.diagrams.AnnotationConstraints.Drag,
            dragLimit: { left: 20, right: 20, top: 20, bottom: 20 }
        }
    ]
};
// 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>
```

Multiple annotations

You can add any number of annotations to a node or connector. The following code illustrates how to add multiple annotations to a node.

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' },
  annotations: [{ content: 'Left', offset: { x: 0.12, y: 0.1 } },
    { content: 'Center', offset: { x: 0.5, y: 0.5 } },
    { content: 'Right', offset: { x: 0.82, y: 0.9 } } ]
};
// 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>

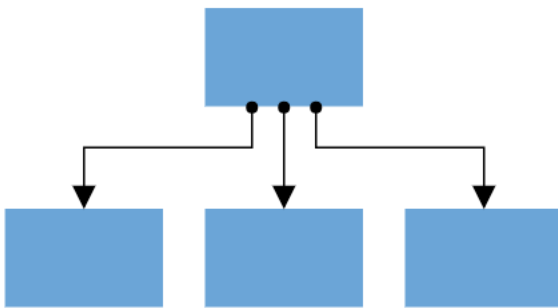
```

Constraints

The constraints property of annotation allows you to enable/disable certain annotation behaviors. For instance, you can disable annotation editing.

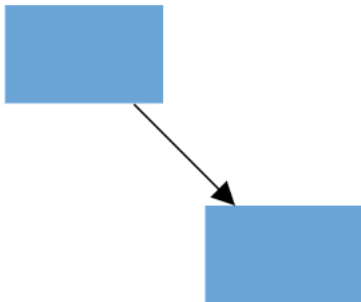
Ports in EJ2 JavaScript Diagram control

Diagram provides support to define custom ports for making connections.

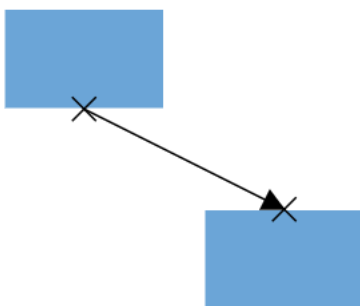


<!-- markdownlint-disable MD033 -->

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>
<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](#).

Constraints in EJ2 JavaScript Diagram control

Constraints are used to enable/disable certain behaviors of the diagram, nodes and connectors. Constraints are provided as flagged enumerations, so that multiple behaviors can be enabled/disabled using Bitwise operators (&, |, ~, <<, etc.).

To know more about Bitwise operators, refer to [Bitwise Operations](#).

Diagram constraints

Diagram constraints allow to enable or disable the following behaviors:

- Page editing
- Bridging
- Zoom and pan
- Undo/redo
- Tooltip

The following example illustrates how to disable page editing using the diagram constraints.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 600,
constraints: DiagramConstraints.Default & ~DiagramConstraints.PageEditable
});
`
```

For more information about diagram constraints, refer to [DiagramConstraints](#).

Node constraints

Node constraints allows to enable or disable the following behaviors of node. They are as follows:

- Selection
- Deletion
- Drag
- Resize
- Rotate
- Connect
- Shadow
- Tooltip

The following example illustrates how to disable rotation using the node constraints.

```
`ts
```

```
var nodes = [{ id: 'node', offsetX: 100, offsetY: 100, constraints: NodeConstraints.Default & ~NodeConstraints.Rotate }];
```

```
var diagram = new ej.diagrams.Diagram({  
width: '100%', height: 600, nodes: nodes,  
}, '#element');
```

```
`
```

For more information about node constraints, refer to [NodeConstraints](#).

Connector constraints

Connector constraints allow to enable or disable certain behaviors of connectors.

- Selection
- Deletion
- Drag
- Segment editing
- Tooltip
- Bridging

The following code illustrates how to disable selection by using connector constraints.

```
`javascript
```

```
var connectors = [{  
id: 'connector1',  
type: 'Straight',  
sourcePoint: { x: 100, y: 100 },  
targetPoint: { x: 200, y: 200 },  
constraints: ConnectorConstraints.Default & ~ConnectorConstraints.Select  
}];
```

```
var diagram = new ej.diagrams.Diagram({  
width: '100%', height: 600, connectors: connectors,  
}, '#element');
```

```
`
```

For more information about connector constraints, refer to [ConnectorConstraints](#).

Port constraints

You can enable or disable certain behaviors of port. They are as follows:

- Connect
- ConnectOnDrag

The following code illustrates how to disable creating connections with a port.

```
`javascript
var nodes = [
{
id: 'node',
offsetX: 100,
offsetY: 100,
ports: [
{
constraints: PortConstraints.None
}
]
}
];
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 600, nodes: nodes,
});
`
```

For more information about port constraints, refer to [PortConstraints](#).

Annotation constraints

You can enable or disable read-only mode for the annotations by using the annotation constraints.

The following code illustrates how to enable read-only mode for the annotations.

```
`javascript
var nodes = [
{
id: 'node',
offsetX: 100,
offsetY: 100,
annotations: [
{
id: 'anotation_1',
content: 'annotation',
constraints: AnnotationConstraints.ReadOnly,
}
]
}
];
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 600, nodes: nodes,
});
`
```

```
}  
]  
}  
];  
var diagram = new ej.diagrams.Diagram({  
width: '100%', height: 600, nodes: nodes,  
});  
`
```

For more details about annotation constraints, refer to [AnnotationConstraints](#).

Selector constraints

Selector visually represents the selected elements with certain editable thumbs. The visibility of the thumbs can be controlled with selector constraints. The part of selector is categorized as follows:

- Resizer
- Rotator
- User handles

The following code illustrates how to hide rotator.

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
width: '100%', height: 600,  
selectedItems: { constraints: SelectorConstraints.All & ~SelectorConstraints.Rotate}  
});  
`
```

For more information about selector constraints, refer to [SelectorConstraints](#).

Snap constraints

Snap constraints control the visibility of gridlines and enable/disable snapping. Snap constraints allow to set the following behaviors.

- Show only horizontal or vertical gridlines.
- Show both horizontal and vertical gridlines.
- Snap to either horizontal or vertical gridlines.
- Snap to both horizontal and vertical gridlines.

The following code illustrates how to show only horizontal gridlines.

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
width: '100%', height: 600,
```

```
snapSettings: {  
  constraints: SnapConstraints.ShowHorizontalLines  
}  
});  
`
```

For more information about snap constraints, refer to [SnapConstraints](#).

Boundary constraints

Boundary constraints defines a boundary for the diagram inside which the interaction should be done. Boundary constraints allow to set the following behaviors.

- Infinite boundary
- Diagram sized boundary
- Page sized boundary

The following code illustrates how to limit the interaction done inside a diagram within a page.

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
  width: '100%', height: 600,  
  pageSettings: {  
    boundaryConstraints: 'Page'  
  }  
});  
`
```

For more information about selector constraints, refer to [BoundaryConstraints](#).

Inherit behaviors

Some of the behaviors can be defined through both the specific object (node/connector) and diagram. When the behaviors are contradictorily defined through both, the actual behavior is set through inherit options.

The following code example illustrates how to inherit the line bridging behavior from the diagram model.

```
`ts  
ej.diagrams.Inject(ej.diagrams.ConnectorBridging);  
var connectors = [{  
  id: 'connector1',  
  type: 'Straight',  
  sourcePoint: { x: 100, y: 100 },  
  targetPoint: { x: 200, y: 200 },  
}
```

```
constraints: ConnectorConstraints.Default & ConnectorConstraints.InheritBridging
});
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 600, connectors: connectors,
constraints: DiagramConstraints.Default | DiagramConstraints.Bridging
});
`
```

Bitwise operations

Bitwise operations are used to manipulate the flagged enumerations [enum]. In this section, Bitwise operations are illustrated by using node constraints. The same is applicable while working with node constraints, connector constraints, or port constraints.

Add operation

You can add or enable multiple values at a time by using Bitwise '|' (OR) operator.

```
`javascript
node.constraints = NodeConstraints.Select | NodeConstraints.Rotate;
`
```

In the previous example, you can do both the selection and rotation operation.

Remove Operation

You can remove or disable values by using Bitwise '&~' (XOR) operator.

```
`javascript
node.constraints = node.constraints & ~(NodeConstraints.Rotate);
`
```

In the previous example, rotation is disabled but other constraints are enabled.

Check operation

You can check any value by using Bitwise '&' (AND) operator.

```
`javascript
if ((node.constraints & (NodeConstraints.Rotate)) == (NodeConstraints.Rotate));
`
```

In the previous example, check whether the rotate constraints are enabled in a node. When node constraints have rotate constraints, the expression returns a rotate constraint.

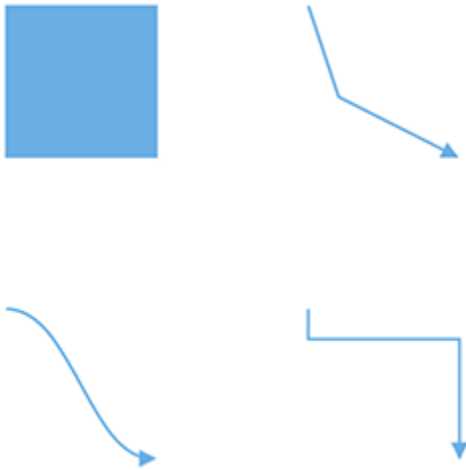
Interaction in EJ2 JavaScript Diagram control

Selection

Selector provides a visual representation of selected elements. It behaves like a container and allows to update the size, position, and rotation angle of the selected elements through interaction and by using program. Single or multiple elements can be selected at a time.

Single selection

An element can be selected by clicking that element. During single click, all previously selected items are cleared. The following image shows how the selected elements are visually represented.



- While selecting the diagram elements, the following events can be used to do your customization.
- When selecting/unselecting the diagram elements, the [selectionChange](#) event gets triggered.

Selecting a group

When a child element of any group is clicked, its contained group is selected instead of the child element. With consecutive clicks on the selected element, selection is changed from top to bottom in the hierarchy of parent group to its children.

Multiple selection

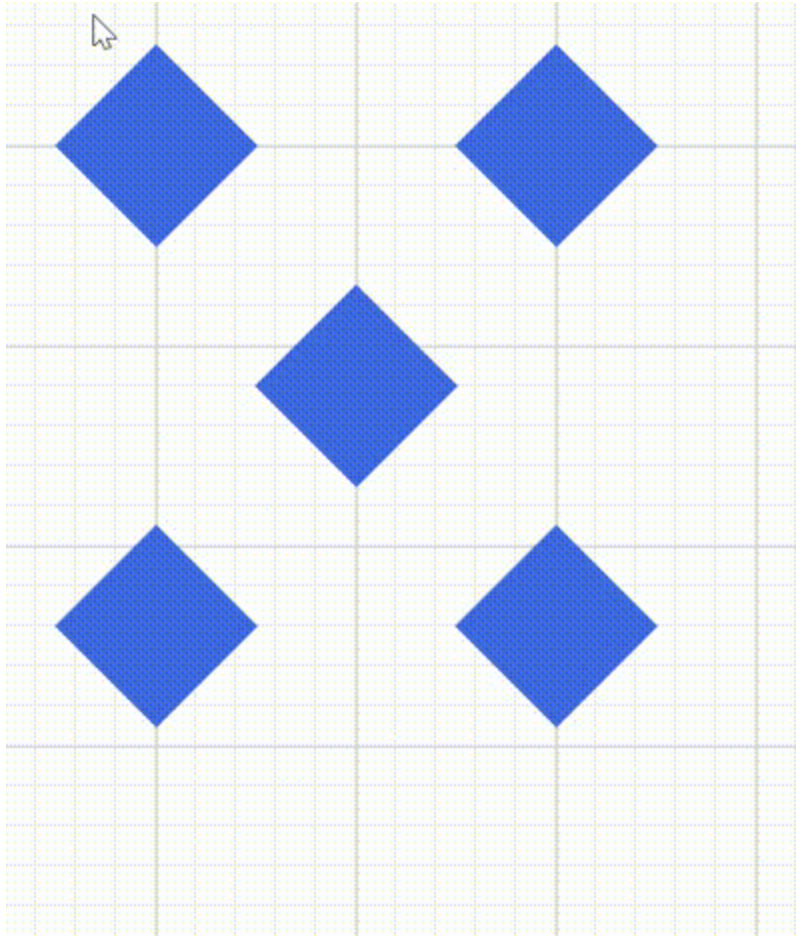
Multiple elements can be selected with the following ways:

- Ctrl+Click

During single click, any existing item in the selection list be cleared, and only the item clicked recently is there in the selection list. To avoid cleaning the old selected item, Ctrl key must be on hold when clicking.

- Selection rectangle/rubber band selection

Clicking and dragging the diagram area allows to create a rectangular region. The elements that are covered under the rectangular region are selected at the end.



Select/Unselect elements using program

The client-side methods [select](#) and [clearSelection](#) help to select or clear the selection of the elements at runtime. The following code example illustrates how to select or clear the selection of an item using program.

Get the current selected items from the [nodes](#) and [connectors](#) collection of the [selectedItems](#) property of the diagram model.

Select entire elements in diagram programmatically

The client-side method [selectAll](#) used to select all the elements such as nodes/connectors in the diagram. Refer to the following link which shows how to use [selectAll](#) method on the diagram.

Drag

- An object can be dragged by clicking and dragging it. When multiple elements are selected, dragging any one of the selected elements move every selected element.
- When you drag the elements in the diagram, the [positionChange](#) event gets triggered and to do customization in this event.



Resize

- Selector is surrounded by eight thumbs. When dragging these thumbs, selected items can be resized.
- When one corner of the selector is dragged, opposite corner is in a static position.
- When a node is resized, the [sizeChange](#) event gets triggered.



Note: While dragging and resizing, the objects are snapped towards the nearest objects to make better alignments. For better alignments, refer to [Snapping](#).

Customize the resize-thumb

You can change the size of the node resize thumb and the connector end point handle by using the `handleSize` property. The appearance such as fill, stroke, and stroke width of the node resize thumb and connector end point handle can be customized by overriding the `e-diagram-resize-handle` and `e-diagram-endpoint-handle` classes respectively.

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' },
  };
  // initialize Diagram component
  var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '600px', nodes: [node],
    selectedItems : {handleSize : 40}
  }, '#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>

```



Rotate

- A rotate handler is placed above the selector. Clicking and dragging the handler in a circular direction lead to rotate the node.
- The node is rotated with reference to the static pivot point.
- Pivot thumb (thumb at the middle of the node) appears while rotating the node to represent the static point.



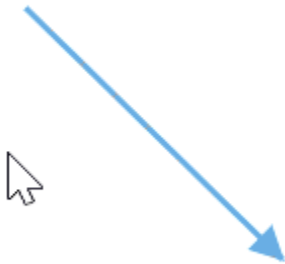
Connection editing

- Each segment of a selected connector is editable with some specific handles/thumbs.

Note: For connector editing, you have to inject the [ConnectorEditing](#) module.

End point handles

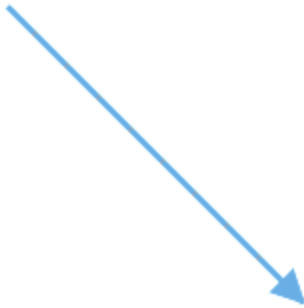
Source and target points of the selected connectors are represented with two handles. Clicking and dragging those handles help you to adjust the source and target points.



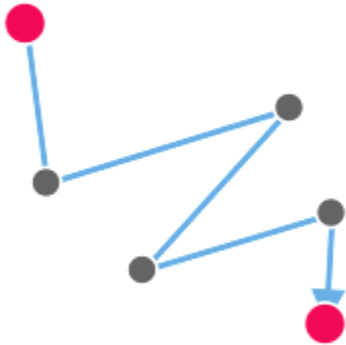
- If you drag the connector end points, then the following events can be used to do your customization.
- When the connector source point is changed, the [sourcePointChange](#) event gets triggered.
- When the connector target point is changed, the [targetPointChange](#) event gets triggered.
- When you connect connector with ports/node or disconnect from it, the [connectionChange](#) event gets triggered.

Straight segment editing

- End point of each straight segment is represented by a thumb that enables to edit the segment.
- Any number of new segments can be inserted into a straight line by clicking, when Shift and Ctrl keys are pressed (Ctrl+Shift+Click).

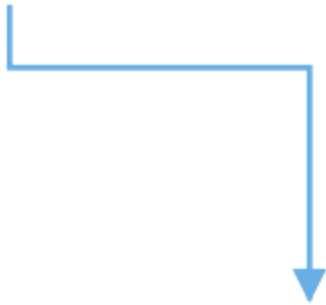


- Straight segments can be removed by clicking the segment end point, when Ctrl and Shift keys are pressed (Ctrl+Shift+Click).



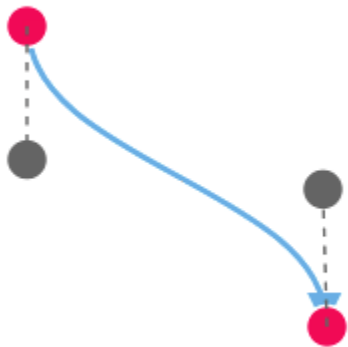
Orthogonal thumbs

- Orthogonal thumbs allow you to adjust the length of adjacent segments by clicking and dragging it.
- When necessary, some segments are added or removed automatically, when dragging the segment. This is to maintain proper routing of orthogonality between segments.



Bezier thumbs

- Bezier segments are annotated with two thumbs to represent the control points. Control points of the curve can be configured by clicking and dragging the control thumbs.



Drag and drop nodes over other elements

Diagram provides support to drop a node/connector over another node/connector. The [drop](#) event is raised to notify that an element is dropped over another one and it is disabled, by default. It can be enabled with the `constraints` property.

User handles

- 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

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

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.

Notification for the mouse button clicked

The diagram component notifies the mouse button clicked. For example, when the right mouse button is clicked, the clicked button is notified as right. The mouse click is notified with,

Notification	Description
Left	When the left mouse button is clicked, left is notified
Middle	When the mouse wheel is clicked, middle is notified
Right	When the right mouse button is clicked, right is notified

```
`javascript
```

```
var diagram = new ej.diagrams.Diagram({
width: '100%', height: 900
}, '#diagram');

diagram.click = function (args: IClickEventArgs) {
// Obtains the mouse button clicked
let clickedButtons = args.button
};
```

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];
    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>

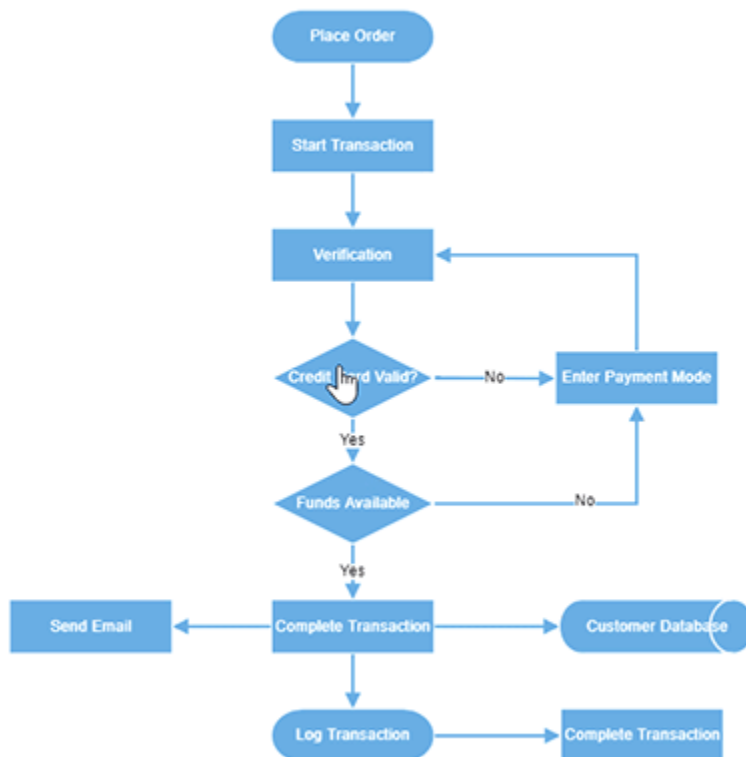
    <div id="container">
        <div id="element"></div>
    </div>
<script>
var ele = document.getElementById ('container');
if (ele) {
    ele.style.visibility = "visible";
}
</script>

```

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

Zoom pan

- When a large diagram is loaded, only certain portion of the diagram is visible. The remaining portions are clipped. Clipped portions can be explored by scrolling the scrollbars or panning the diagram.
- Diagram can be zoomed in or out by using Ctrl + mouse wheel.
- When the diagram is zoomed or panned, the [scrollChange](#) event gets triggered.



Zoom pan status

Diagram provides the support to notify the pan status of the zoom pan tool. Whenever the diagram is panning [scrollChange](#) event is triggered and hence the pan status can be obtained. The pan status is notified with Start, Progress, and Completed.

Pan Status	Description
Start	When the mouse is clicked and dragged the status is notified as start.
Progress	When the mouse is in motion the status is notified as progress.
Completed	When panning is stopped the status is notified with completed.

`ts

```

let diagram: Diagram = new Diagram({
width: '100%', height: 900
tool: DiagramTools.ZoomPan, scrollChange:function (args:IScrollChangeEventArgs |
IBlazorScrollChangeEventArgs) {
// Obtains the zoom pan status
let mouseButton = args.panState
}
});
diagram.appendTo('#element');
`

```

Keyboard

Diagram provides support to interact with the elements with key gestures. By default, some in-built commands are bound with a relevant set of key combinations.

The following table illustrates those commands with the associated key values.

Shortcut Key	Command	Description
-----	-----	-----
Ctrl + A	selectAll	Select all nodes/connectors in the diagram.
Ctrl + C	copy	Copy the diagram selected elements.
Ctrl + V	paste	Pastes the copied elements.
Ctrl + X	cut	Cuts the selected elements.
Ctrl + Z	undo	Reverses the last editing action performed on the diagram.
Ctrl + Y	redo	Restores the last editing action when no other actions have occurred since the last undo on the diagram.
Delete	delete	Deletes the selected elements.
Ctrl/Shift + Click on object		Multiple selection (Selector binds all selected nodes/connectors).
Up Arrow	nudge("up")	nudgeUp : Moves the selected elements towards up by one pixel.
Down Arrow	nudge("down")	nudgeDown : Moves the selected elements towards down by one pixel.
Left Arrow	nudge("left")	nudgeLeft : Moves the selected elements towards left by one pixel.
Right Arrow	nudge("right")	nudgeRight : Moves the selected elements towards right by one pixel.
Ctrl + MouseWheel	zoom	Zoom (Zoom in/Zoom out the diagram).
F2	startLabelEditing	Starts to edit the label of selected element.
Esc	endLabelEditing	Sets the label mode as view and stops editing.

See Also

- [How to create diagram nodes using drawing tools](#)
- [How to create diagram connectors using drawing tools](#)
- [How to disable the diagram interaction](#)
- [How to control the diagram history](#)
- [How to create overview control to the diagram](#)

Tools in EJ2 JavaScript Diagram control

Drawing tools

Drawing tool allows you to draw any kind of node/connector during runtime by clicking and dragging on the diagram page.

Shapes

To draw a shape, set the JSON of that shape to the drawType property of the diagram and activate the drawing tool by using the [tool](#) property. The following code example illustrates how to draw a rectangle at runtime.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: 700, height: 700, created: () => {
    var drawingshape = { type: 'Basic', shape: 'Rectangle' };
    var node = {
      shape: drawingshape
    };
    diagram.drawingObject = node;
    diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
    diagram.dataBind();
  }
});
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>

```

The following code example illustrates how to draw a path.

INDEX.JS

```

var diagram = new ej.diagrams.Diagram({
    width: 700, height: 700, created: () => {
        var node = {
            id: "Path",
            style: { fill: "#fbe172" },
            annotations: [{
                content: "Path"
            }],
            shape: {
                type: 'Path',
                data: 'M13.560 67.524 L 21.941 41.731 L 0.000 25.790 L
27.120 25.790 L 35.501 0.000 L 43.882 25.790 L 71.000 25.790 L 49.061 41.731
L 57.441 67.524 L 35.501 51.583 z'
            }
        };
        diagram.drawingObject = node;
        diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
        diagram.dataBind();
    }
});
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>

```

Connectors

To draw connectors, set the JSON of the connector to the drawType property. The drawing [tool](#) can be activated by using the tool property. The following code example illustrates how to draw a straight line connector.

INDEX.JS

```

var diagram = new ej.diagrams.Diagram({
    width: 700, height: 700, created: () => {
        var connectors = {
            id: 'connector1',
            type: 'Straight',
            segments: [{ type: "polyline" }]
        }
        diagram.drawingObject = connectors;
        diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
        diagram.dataBind();
    }
});
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>
```

Text

Diagram allows you to create a textNode, when you click on the diagram page. The following code illustrates how to draw a text.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: 700, height: 700, created: () => {
    var drawingshape = { type: 'Basic', shape: 'Rectangle' };
    var node = {
      shape: drawingshape
    };
    diagram.drawingObject = node;
    diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
```

```

        diagram.dataBind();
    }
});
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>
```

Once you activate the TextTool, perform label editing of a node/connector.

Polygon shape

Diagram allows to create the polygon shape by clicking and moving the mouse at runtime on the diagram page.

The following code illustrates how to draw a polygon shape.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: 700, height: 700, created: () => {
    var node = {
      shape: {
        type: 'Text',
      }
    };
    diagram.drawingObject = node;
    diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
    diagram.dataBind();
  }
});
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>
```

Polyline Connector

Diagram allows to create the polyline segments with straight lines and angled vertices at the control points by clicking and moving the mouse at runtime on the diagram page.

The following code illustrates how to draw a polyline connector.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: 700, height: 700, created: () => {
    var connector = { id: 'connector1', type: 'Polyline' };
    diagram.drawingObject = connector;
    diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
    diagram.dataBind();
  }
});
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>

```

Tool selection

There are some functionalities that can be achieved by clicking and dragging on the diagram surface. They are as follows.

- Draw selection rectangle: MultipleSelect tool
- Pan the diagram: Zoom pan
- Draw nodes/connectors: DrawOnce/DrawOnce

As all the three behaviors are completely different, you can achieve only one behavior at a time based on the tool that you choose.

When more than one of those tools are applied, a tool is activated based on the precedence given in the following table.

Precedence	Tools	Description
1st	ContinuesDraw	Allows you to draw the nodes or connectors continuously. Once it is activated, you cannot perform any other interaction in the diagram.
2nd	DrawOnce	Allows you to draw a single node or connector. Once you complete the DrawOnce action, SingleSelect, and MultipleSelect tools are automatically enabled.
3rd	ZoomPan	Allows you to pan the diagram. When you enable both the SingleSelect and ZoomPan tools, you can perform the basic interaction as the cursor hovers node/connector. Panning is enabled when cursor hovers the diagram.
4th	MultipleSelect	Allows you to select multiple nodes and connectors. When you enable both the MultipleSelect and ZoomPan tools, cursor hovers the diagram. When panning is enabled, you cannot select multiple nodes.
5th	SingleSelect	Allows you to select individual nodes or connectors.
6th	None	Disables all tools.

Set the desired [tool](#) to the tool property of the diagram model. The following code illustrates how to enable Zoom pan in the diagram

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,
        annotations: [{
            id: 'label2',
            content: 'End'
        }],
        shape: {
            type: 'Flow',
            shape: 'process'
        },
        annotations: [{
            content: 'var i = 0;'
        }]
    }
];
var connector = {
    id: "connector1",
    id: "connector1",
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourceID: "Start",
    targetID: "Init",
    connectorSpacing: 7,
    type: 'Orthogonal'
};
var diagram = new ej.diagrams.Diagram({
    width: 700, height: 700,
    nodes: nodes,
    connectors: [connector], created: () => {
        tool:DiagramTools.DrawOnce | DiagramTools.ZoomPan;
    }
});
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>

```

Events

[elementDraw](#) event is triggered when node or connector is drawn using drawing tool.

INDEX.JS

```

var diagram = new ej.diagrams.Diagram({
    width: 700, height: 700, created: () => {
        var connectors = {
            id: 'connector1',
            type: 'Straight',
            segments: [{ type: "Straight" }]
        }
        diagram.drawingObject = connectors;
        diagram.tool = ej.diagrams.DiagramTools.ContinuousDraw;
        elementDraw : elementDraw;
        diagram.dataBind();
    }
});

```

```
});  
diagram.appendTo('#element');  
function elementDraw(args) {  
    alert("Event triggered");  
}
```

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>
```

Freehand Drawing

Diagram has support for free-hand drawing to draw anything on the diagram page independently. Free-hand drawing will be enabled by using the drawingObject property and setting its value to Freehand.

The following code illustrates how to draw a freehand drawing.

INDEX.JS

```
var diagram = new ej.diagrams.Diagram({
  width: 700, height: 700, created: () => {
    var connector = { id: 'connector1', type: 'Freehand' };
    diagram.drawingObject = connector;
    diagram.tool = ej.diagrams.DiagramTools.DrawOnce;
    diagram.dataBind();
  }
});
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>
```

Grid lines in EJ2 JavaScript 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.TS

```
import {Diagram, SnapConstraints, SnapSettingsModel, Snapping} from
 '@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
    // Display both Horizontal and Vertical gridlines
    constraints: SnapConstraints.ShowLines };
let diagram: Diagram = new Diagram({
    width: '100%', height: '500px',
    snapSettings: snapSettings
});
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>
</html>
```



```
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.TS

```
import {Diagram, SnapConstraints, SnapSettingsModel, Snapping} from
 '@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
  // Define the Constraints for gridlines and snapping
  constraints: SnapConstraints.ShowLines,
  // Defines the horizontalGridlines for SnapSettings
  horizontalGridlines: {
    // Sets the line color of gridlines
    lineColor: 'blue',
    // Defines the lineDashArray of gridlines
    lineDashArray: '2 2'
  },
  // Defines the verticalGridlines for SnapSettings
  verticalGridlines: {
    lineColor: 'blue',
    lineDashArray: '2 2'
  }
};
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '500px',
  // Define the snap setting for the diagram
  snapSettings: snapSettings
});
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>

```

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.TS

```

import {Diagram, SnapConstraints, SnapSettingsModel, Snapping} from
'@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
    constraints: SnapConstraints.ShowLines,
    horizontalGridlines: {
        // Sets the lineIntervals of Gridlines

```

```

        lineIntervals: [1.25, 14, 0.25, 15, 0.25, 15, 0.25, 15, 0.25, 15],
        lineColor: 'blue',
        lineDashArray: '2 2'
    },
    verticalGridlines: {
        // Sets the lineIntervals of Gridlines
        lineIntervals: [1.25, 14, 0.25, 15, 0.25, 15, 0.25, 15, 0.25, 15],
        lineColor: 'blue',
        lineDashArray: '2 2'
    }
};
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '500px',
    snapSettings: snapSettings
});
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>
<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.TS

```
import {Diagram, SnapConstraints, SnapSettingsModel, Snapping, NodeModel} from
 '@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
  // Enables the object to snap with both horizontal and Vertical
  gridlines
  constraints: SnapConstraints.SnapToLines | SnapConstraints.ShowLines
};
let nodes: NodeModel[] = [{
  id: 'node1',
  style:{fill: '#6BA5D7', strokeColor: '#6BA5D7'},
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
}];
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '500px',
  nodes: nodes,
  snapSettings: snapSettings
});
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 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.TS

```

import {Diagram, SnapConstraints, SnapSettingsModel, Snapping, NodeModel} from
'@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
    horizontalGridlines: {
        // Defines the snap interval for object
        snapIntervals: [10]
    },
    verticalGridlines: {
        snapIntervals: [10]
    },
    constraints: SnapConstraints.All
};
let nodes: NodeModel[] = [{
    id: 'node1',
    width: 100,
    style:{fill: '#6BA5D7',strokeColor: '#6BA5D7'},
    height: 100,
    offsetX: 100,
    offsetY: 100
}];
let diagram: Diagram = new Diagram({
    width: '100%',

```

```

        height: '500px',
        nodes: nodes,
        snapSettings: snapSettings
    });
    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>

```

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.TS

```
import {Diagram, SnapConstraints, SnapSettingsModel, Snapping, NodeModel} from
 '@syncfusion/ej2-diagrams';
Diagram.Inject(Snapping);
let nodes: NodeModel[] = [{
  id: 'node1',
  style:{fill:'#6BA5D7',strokeColor:'#6BA5D7'},
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100
},{
  id: 'node2',
  style:{fill:'#6BA5D7',strokeColor:'#6BA5D7'},
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100
}];
let snapSettings: SnapSettingsModel = {
  // Enable snap to object constraint
  constraints: SnapConstraints.SnapToObject|SnapConstraints.ShowLines,
  // Sets the Snap object distance
  snapObjectDistance: 10,
  // Snap Angle for object
  snapAngle: 10,
  // Set the Snapline color
  snapLineColor: 'red'
};
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '500px',
  nodes: nodes,
  snapSettings: snapSettings
});
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>

```

Page settings in EJ2 JavaScript 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.TS

```

import {Diagram,ConnectorModel,NodeModel} from '@syncfusion/ej2-diagrams';
let diagram: Diagram;
let connector: ConnectorModel = {
    id: 'connector1',
    style: { strokeColor: '#6BA5D7', fill: '#6BA5D7', strokeWidth: 2 },
    targetDecorator: { style: { fill: '#6BA5D7', strokeColor: '#6BA5D7' } },
    sourceID: 'node1',
    targetID: 'node2',

```



```
};
let node: NodeModel = {
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
};
let node2: NodeModel = {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node3'
  }]
};
diagram = new Diagram({
  width: '1000px',
  height: '500px',
  getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  },
  nodes: [node, node2],
  connectors: [connector],
  // Defines the pageSettings for the diagram
  pageSettings: {
    // Sets the PageOrientation for the diagram to page
    orientation: 'Landscape',
    // Sets the Page Break for diagram
    showPageBreaks: true,
    // Defines the background color and image of diagram
    background: {
      color: 'grey'
    },
    // Sets the width for the Page
    width: 300,
    // Sets the height for the Page
    height: 300,
    // Sets the space to be left between an annotation and its parent
    // node/connector
    margin: {
      left: 10,
      top: 10,
      bottom: 10
    },
  },
});
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>

```

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.TS

```

import {Diagram} from '@syncfusion/ej2-diagrams';
let diagram: Diagram;
diagram = new Diagram({
  width: '1000px',

```

```

height: '500px',
pageSettings: {
  orientation: 'Landscape',
  showPageBreaks: true,
  // Defines the background Image source
  background: {
    source: 'https://www.w3schools.com/images/w3schools_green.jpg',
    // Defines the scale values for the background image
    scale: 'Meet',
    // Defines the align values for the background image
    align: 'XMinYMin'
  },
  width: 300,
  height: 300,
  margin: {
    left: 10,
    top: 10,
    bottom: 10
  },
},
});
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>
```

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.TS

```
import {Diagram, ConnectorModel, NodeModel} from '@syncfusion/ej2-diagrams';
let diagram: Diagram;
let connector: ConnectorModel = {
  id: 'connector1',
  style: { strokeColor: '#6BA5D7', fill: '#6BA5D7', strokeWidth: 2 },
  targetDecorator: { style: { fill: '#6BA5D7', strokeColor: '#6BA5D7' } },
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
};
let node: NodeModel = {
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
};
let node2: NodeModel = {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node2'
  }]
};
diagram = new Diagram({
  width: '1000px',
```

```

height: '500px',
nodes: [node, node2],
connectors: [connector],
getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
},
pageSettings: {
    orientation: 'Landscape',
    // Sets the Multiple page for diagram
    multiplePage: true,
    // Sets the Page Break for diagram
    showPageBreaks: true,
    width: 300,
    height: 300,
    margin: {
        left: 10,
        top: 10,
        bottom: 10
    },
}
});
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>
```

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.TS

```
import {Diagram,ConnectorModel,NodeModel,BoundaryConstraints} from
 '@syncfusion/ej2-diagrams';
let diagram: Diagram;
let connector: ConnectorModel = {
  id: 'connector1',
  style: { strokeColor: '#6BA5D7', fill: '#6BA5D7', strokeWidth: 2 },
  targetDecorator: { style: { fill: '#6BA5D7', strokeColor: '#6BA5D7' } },
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 100
  }
};
let node: NodeModel = {
  id: 'node1',
  width: 150,
  height: 100,
  offsetX: 100,
  offsetY: 100,
};
let node2: NodeModel = {
  id: 'node2',
  width: 80,
  height: 130,
  offsetX: 200,
  offsetY: 200,
};
diagram = new Diagram({
  width: 800,
  height: 800,
  nodes: [node, node2],
```

```

connectors: [connector],
getNodeDefaults: (node: NodeModel) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
},
pageSettings: {
  // Sets the BoundaryConstraints to page
  boundaryConstraints: 'Page',
  background: {
    color: 'grey'
  },
  width: 400,
  height: 400,
  showPageBreaks: 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>
```

Scroll settings in EJ2 JavaScript 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.TS

```
import {
    Diagram, BasicShapeModel, NodeModel, DiagramTools
} from '@syncfusion/ej2-diagrams';
//Sets scroll status
let diagram: Diagram = new Diagram({
    width: '100%', height: 700, scrollSettings: {
        horizontalOffset: 100, verticalOffset: 50
    }
});
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>
```

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.TS

```
import {
    Diagram, BasicShapeModel, NodeModel, DiagramTools
} from '@syncfusion/ej2-diagrams';
let diagram: Diagram = new Diagram({
    width: '100%', height: 700
});
diagram.appendTo('#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.TS

```
import {
    Diagram, NodeModel, ConnectorModel
} from '@syncfusion/ej2-diagrams';
```

```

let nodes: NodeModel[] = [{
  id: 'Start',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Start'
  }],
}];
let connectors: ConnectorModel[] = [{
  id: 'connector1', sourcePoint: { x: 300, y: 100 }, targetPoint: { x:
450, y: 200 },
  style: {
    strokeColor: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
}];
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  nodes: nodes,
  connectors: connectors,
  // set the autoScroll
  scrollSettings:{canAutoScroll: true, scrollLimit: 'Infinity'},
  getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return 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>

```

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.TS

```

import {
    Diagram, NodeModel, ConnectorModel
} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'labell1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
let diagram: Diagram = new Diagram({
    width: '100%',

```

```

    height: '600px',
    nodes: nodes,
    // set the autoScrollBorder
    scrollSettings:{canAutoScroll: true, scrollLimit: 'Infinity',
    autoScrollBorder:{left:100,right:100,top:100,bottom:100}},
    getNodeDefaults: (node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return 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>

```

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.TS

```
import {
  Diagram, NodeModel, ConnectorModel
} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  nodes: nodes,
  // set the autoScrollBorder
  scrollSettings: {
    canAutoScroll: true,
    //Sets the scroll limit
    scrollLimit: 'infinity'
  },
  getNodeDefaults: (node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return 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>

```

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.TS

```

import {
  Diagram, NodeModel, ConnectorModel
} from '@syncfusion/ej2-diagrams';
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 100, height: 100,

```

```
offsetX: 350, offsetY: 350,
shape: {
  type: 'Flow',
  shape: 'Terminator'
}
}];
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  nodes: nodes,
  scrollSettings: {
    canAutoScroll: true,
    //Sets the scroll padding
    padding: { right: 50, bottom: 50 }
  }
},
});
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>
```

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.TS

```
import {  
    Diagram, NodeModel, ConnectorModel  
} from '@syncfusion/ej2-diagrams';  
let nodes: NodeModel[] = [{  
    id: 'Start',  
    width: 140,  
    height: 50,  
    offsetX: 300,  
    offsetY: 50,  
    annotations: [{  
        id: 'label1',  
        content: 'Start'  
    }],  
    shape: {  
        type: 'Flow',  
        shape: 'Terminator'  
    }  
}];  
let diagram: Diagram = new 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: NodeModel) => {  
        node.height = 100;  
        node.width = 100;  
        node.style.fill = '#6BA5D7';  
        node.style.strokeColor = 'white';  
        return 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>
```

UpdateViewport

The [updateViewPort](#) method is used to update the diagram page and view size at runtime.

Data binding in EJ2 JavaScript Diagram control

- Diagram can be populated with the **nodes** and **connectors** based on the information provided from an external data source.
- Diagram exposes its specific data-related properties allowing you to specify the data source fields from where the node information has to be retrieved from.
- The [dataManager](#) property is used to define the data source either as a collection of objects or as an instance of **DataManager** that needs to be populated in the diagram.
- The [ID](#) property is used to define the unique field of each JSON data.

- The [parentId](#) property is used to defines the parent field which builds the relationship between ID and parent field.
- The [root](#) property is used to define the root node for the diagram populated from the data source.
- To explore those properties, see [DataSourceSettings](#).
- Diagram supports two types of data binding. They are:
 1. Local data
 2. Remote data

Local data

Diagram can be populated based on the user defined JSON data (Local Data) by mapping the relevant data source fields.

To map the user defined JSON data with diagram, configure the fields of [dataSourceSettings](#). The following code example illustrates how to bind local data with the diagram.

INDEX.JS

```
var species = [
  { 'Name': 'Species', 'fillColor': '#3DD94A' },
  { 'Name': 'Plants', 'Category': 'Species' },
  { 'Name': 'Fungi', 'Category': 'Species' },
  { 'Name': 'Lichens', 'Category': 'Species' },
  { 'Name': 'Animals', 'Category': 'Species' },
  { 'Name': 'Mosses', 'Category': 'Plants' },
  { 'Name': 'Ferns', 'Category': 'Plants' },
  { 'Name': 'Gymnosperms', 'Category': 'Plants' },
  { 'Name': 'Dicotyledens', 'Category': 'Plants' },
  { 'Name': 'Monocotyledens', 'Category': 'Plants' },
  { 'Name': 'Invertebrates', 'Category': 'Animals' },
  { 'Name': 'Vertebrates', 'Category': 'Animals' },
  { 'Name': 'Insects', 'Category': 'Invertebrates' },
  { 'Name': 'Molluscs', 'Category': 'Invertebrates' },
  { 'Name': 'Crustaceans', 'Category': 'Invertebrates' },
  { 'Name': 'Others', 'Category': 'Invertebrates' },
  { 'Name': 'Fish', 'Category': 'Vertebrates' },
  { 'Name': 'Amphibians', 'Category': 'Vertebrates' },
  { 'Name': 'Reptiles', 'Category': 'Vertebrates' },
  { 'Name': 'Birds', 'Category': 'Vertebrates' },
  { 'Name': 'Mammals', 'Category': 'Vertebrates' }
];
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
//Initializes diagram control
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: 490,
  //Configures data source
  dataSourceSettings: {
    id: 'Name', parentId: 'Category', dataManager: new
ej.data.DataManager(species),
    //binds the external data with node
    doBinding: function (nodeModel, data, diagram) {
      nodeModel.annotations = [{
        content: data.Name, margin: { top: 10, left: 10, right:
10, bottom: 0 },
```

```

        style: { color: 'black' }
    };
    nodeModel.style = { fill: '#ffeec7', strokeColor: '#f5d897',
strokeWidth: 1 };
    },
    //Configures HierarchicalTree layout
    layout: {
        type: 'HierarchicalTree', horizontalSpacing: 15,
verticalSpacing: 50,
        margin: { top: 10, left: 10, right: 10, bottom: 0 },
    },
    //Sets the default values of nodes
    getNodeDefaults: function (obj, diagram) {
        obj.shape = { type: 'Basic', shape: 'Rectangle' };
        obj.style = { strokeWidth: 1 };
        obj.width = 95;
        obj.height = 30;
    },
    //Sets the default values of connectors.
    getConnectorDefaults: function (connector, diagram) {
        connector.type = 'Orthogonal';
        connector.style.strokeColor = '#4d4d4d';
        connector.targetDecorator.shape = 'None';
    },
    //Disables all interactions except zoom/pan
    tool: ej.diagrams.DiagramTools.ZoomPan,
    snapSettings: { constraints: 0 }
});
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>
```

Remote data

You can bind the diagram with remote data by using `[dataManager]`.

It uses two different classes: `DataManager` for processing and `Query` for serving data. `DataManager` communicates with data source and `Query` generates data queries that are read by the [dataManager](#).

To learn more about data manager, refer to [Data Manager](#).

To bind remote data to the diagram, configure the fields of [dataSourceSettings](#). The following code illustrates how to bind remote data to the diagram.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
//Initializes diagram control
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: 490,
    //Configures hierarchical tree layout
    layout: {
        type: 'HierarchicalTree', margin: { left: 0, right: 0, top: 100,
bottom: 0 },
        verticalSpacing: 40,
        getLayoutInfo: function (node, options) {
            if (options.level === 3) {
                node.style.fill = '#3c418d';
            }
            if (options.level === 2) {
                node.style.fill = '#108d8d';
                options.type = 'Center';
                options.orientation = 'Horizontal';
            }
            if (options.level === 1) {
                node.style.fill = '#822b86';
            }
        }
    },
    },
```

```

//Sets the default values of nodes
getNodeDefaults: function (obj) {
    obj.width = 80;
    obj.height = 40;
    obj.shape = { type: 'Basic', shape: 'Rectangle' };
    obj.style = { fill: '#048785', strokeColor: 'Transparent' };
},
//Sets the default values of connector
getConnectorDefaults: function (connector) {
    connector.type = 'Orthogonal';
    connector.style.strokeColor = '#048785';
    connector.targetDecorator.shape = 'None';
},
//Configures data source
dataSourceSettings: {
    id: 'Id',
    parentId: 'ParentId',
    dataSource: new ej.data.DataManager({
        url:
'https://services.syncfusion.com/js/production/api/RemoteData',
        crossDomain: true,
    }),
    //binds the external data with node
    doBinding: function (nodeModel, data, diagram) {
        nodeModel.annotations = [{
            content: data.Label,
            style: { color: 'white' }
        }];
    }
},
//Disables all interactions except zoom/pan
tool: ej.diagrams.DiagramTools.ZoomPan,
snapSettings: { constraints: 0 }
});
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>
```

CRUD

This feature allows you to read the data source and perform add or edit or delete the data in data source at runtime.

Read DataSource

- This feature allows you to define the nodes and connectors collection in the data source and connectionDataSource respectively.
- You can set the data collection in the model's dataSourceSettings [dataManager](#) property. The nodes will be generated based on the data specified in the data source.
- You can set the connector collection in the model's dataSourceSettings [connectionDataSource](#) property.
- The dataSourceSettings connectionDataSource [dataManager](#) property is used to set the data source for the connection data source items.
- If you have a data (data will be set in the dataSource property) with parent relationship in the database and also defined the connector in the connectionDataSource simultaneously, then the connectors set in the connectionDataSource will be considered as a priority to render the connector.
- The dataSourceSettings [crudAction's read](#) property specifies the method, which is used to read the data source and its populate the nodes in the diagram.
- The connectionDataSource crudAction's [read](#) specifies the method, which is used to read the data source and its populates the connectors in the diagram.
- The dataSourceSettings's [id](#) and connectionDataSource's [id](#) properties are used to define the unique field of each JSON data.
- The connectionDataSource's [sourceID](#) and [targetID](#) properties are used to set the sourceID and targetID for connection data source item.
- The connectionDataSource's [sourcePointX](#), [sourcePointY](#), [targetPointX](#), and [targetPointY](#) properties are used to define the sourcePoint and targetPoint values for connector from data source.

- The dataSourceSettings crudAction's [customFields](#) property is used to maintain the additional information for nodes.
- Similarly, connectionDataSource's crudAction's [customFields](#) is used to maintain the additional information for connectors.

How to perform Editing at runtime

- The dataSourceSettings crudAction object allows you to define the method, which is used to get the changes done in the data source defined for shapes from the client-side to the server-side.
- Similarly, the connectionDataSource crudAction object allows you to define the method, which is used to get the changes done in the data source defined for connectors from the client-side to the server-side.

InsertData

- The dataSourceSettings crudAction's [create](#) property specifies the method, which is used to get the nodes added from the client-side to the server-side.
- The connectionDataSource crudAction's [create](#) specifies the method, which is used to get the connectors added from the client-side to the server-side.
- The following code example illustrates how to send the newly added or inserted data from the client to server-side.

```
`ts
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
  dataSourceSettings: {
    crudAction:
    {
      //Url which triggers the server side AddNodes method
      create: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/AddNodes',
    },
  },
  connectionDataSource: {
    crudAction: {
      //Url which triggers the server side AddConnectors method
      create: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/AddConnectors',
    },
  },
}, '#diagram');
```


//Sends the newly added nodes/connectors from client side to the server side through the URL which is specified in server side.

```
diagram.insertData();
```

```
,
```

UpdateData

- The dataSourceSettings crudAction's [update](#) property specifies the method, which is used to get the modified nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [update](#) specifies the method, which is used to get the modified connectors from the client-side to the server-side.
- The following code example illustrates how to send the updated data from the client to the server side.

```
`ts
```

```
//Initialize diagram
```

```
var diagram = new ej.diagrams.Diagram({
```

```
dataSourceSettings: {
```

```
crudAction:
```

```
{
```

```
//Url which triggers the server side UpdateNodes method
```

```
update: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/UpdateNodes',
```

```
},
```

```
}
```

```
connectionDataSource: {
```

```
crudAction: {
```

```
//Url which triggers the server side UpdateConnectors method
```

```
update: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/UpdateConnectors',
```

```
}
```

```
}
```

```
}, '#diagram');
```

//Sends the updated nodes/connectors from client side to the server side through the URL which is specified in server side.

```
diagram.updateData();
```

```
,
```

DeleteData

- The dataSourceSettings crudAction's [destroy](#) property specifies the method, which is used to get the deleted nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [destroy](#) specifies the method, which is used to get the deleted connectors from the client-side to the server-side.

`ts

```
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
  dataSourceSettings: {
    crudAction:
    {
      //Url which triggers the server side DeleteNodes method
      destroy: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/DeleteNodes',
    },
  },
  connectionDataSource: {
    crudAction: {
      //Url which triggers the server side DeleteConnectors method
      destroy: 'https://ej2services.syncfusion.com/development/web-services/api/Crud/DeleteConnectors',
    }
  },
  '#diagram');
//Sends the deleted nodes/connectors from client side to the server side through the URL which is
specified in server side.
diagram.removeData();
`
```

See Also

- [How to arrange the diagram nodes and connectors using varies layout](#)

Automatic layout in EJ2 JavaScript Diagram control

Diagram provides support to auto-arrange the nodes in the diagram area that is referred as **Layout**. It includes the following layout modes:

Layout modes

- Hierarchical layout

- Organization chart
- Radial tree
- Symmetric layout
- Mind Map layout
- Complex hierarchical tree layout

Hierarchical layout

The hierarchical tree layout arranges nodes in a tree-like structure, where the nodes in the hierarchical layout may have multiple parents. There is no need to specify the layout root. To arrange the nodes in a hierarchical structure, specify the layout [type](#) as hierarchical tree. The following example shows how to arrange the nodes in a hierarchical structure.

Note: If you want to use hierarchical tree layout in diagram, you need to inject HierarchicalTree in the diagram.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [{
    Name: "Steve-Ceo"
},
{
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo"
},
{
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo"
},
{
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager"
},
{
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager"
},
{
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager"
},
{
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager"
}
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '650px',
    //Uses layout to auto-arrange nodes on the diagram page
    layout: {
        //Sets layout type
        type: 'HierarchicalTree'
    }
});
```

```

}, //Configures data source for diagram
dataSourceSettings: {
  id: 'Name',
  parentId: 'ReportingPerson',
  dataManager: items
}, //Sets the default properties for nodes
getNodeDefaults: (obj) => {
  obj.shape = {
    type: 'Text',
    content: (obj.data.Name)
  };
  obj.style = {
    fill: 'None',
    strokeColor: 'none',
    strokeWidth: 2,
    bold: true,
    color: 'white'
  };
  obj.borderColor = 'white';
  obj.width = 100;
  obj.height = 40;
  obj.backgroundColor = '#6BA5D7';
  obj.borderWidth = 1;
  (obj.shape).margin = {
    left: 5,
    right: 5,
    top: 5,
    bottom: 5
  };
  return obj;
}, //Sets the default properties for and connectors
getConnectorDefaults: (connector, diagram) => {
  connector.style = {
    strokeColor: '#6BA5D7',
    strokeWidth: 2
  };
  connector.targetDecorator.style.fill = '#6BA5D7';
  connector.targetDecorator.style.strokeColor = '#6BA5D7';
  connector.type = 'Orthogonal';
  return connector;
}
});
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>

```

Radial tree layout

The radial tree layout arranges nodes on a virtual concentric circle around a root node. Sub-trees formed by the branching of child nodes are located radially around the child nodes. This arrangement result in an ever-expanding concentric arrangement with radial proximity to the root node indicating the node level in the hierarchy. The layout [root](#) property can be used to define the root node of the layout. When no root node is set, the algorithm automatically considers one of the diagram nodes as the root node.

To arrange nodes in a radial tree structure, set the [type](#) of the layout as **RadialTree**. The following code illustrates how to arrange the nodes in a radial tree structure.

Note: If you want to use radial tree layout in diagram, you need to inject DataBinding and RadialTree in the diagram.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
    {
        "Id": "parent", "Name": "Maria Anders", "Designation": "Managing
Director",
        "ImageUrl": "../content/images/radialtree/Clayton.png",
        "IsExpand": "true", "RatingColor": "#C34444"
    },

```

```

    {
        "Id": 1, "Name": "Ana Trujillo", "Designation": "Project
Manager",
        "ImageUrl": "../content/images/radialtree/Thomas.PNG",
        "IsExpand": "false",
        "RatingColor": "#68C2DE", "ReportingPerson": "parent"
    },
    {
        "Id": 2, "Name": "Lino Rodri", "Designation": "Project Manager",
        "ImageUrl": "../content/images/radialtree/Robin.PNG",
        "IsExpand": "true",
        "RatingColor": "#68C2DE", "ReportingPerson": "parent"
    },
    {
        "Id": 3, "Name": "Philip Cramer", "Designation": "Project
Manager",
        "ImageUrl": "../content/images/radialtree/Robin.PNG",
        "IsExpand": "true",
        "RatingColor": "#68C2DE", "ReportingPerson": "parent"
    },
    {
        "Id": 4, "Name": "Pedro Afonso", "Designation": "Project
Manager",
        "ImageUrl": "../content/images/radialtree/Paul.png", "IsExpand":
"true",
        "RatingColor": "#68C2DE", "ReportingPerson": 1
    },
    {
        "Id": 5, "Name": "Anto Moreno", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/image53.png",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 1
    },
    {
        "Id": 6, "Name": "Elizabeth Roel", "Designation": "Project
Lead",
        "ImageUrl": "../content/images/radialtree/Maria.png",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 2
    },
    {
        "Id": 7, "Name": "Aria Cruz", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Jenny.png",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 3
    },
    {
        "Id": 8, "Name": "Eduardo Roel", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/image55.png",
        "IsExpand": "true",
        "RatingColor": "#93B85A", "ReportingPerson": 1
    },
    {
        "Id": 9, "Name": "Howard Snyd", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Jenny.png",
        "IsExpand": "false",
        "RatingColor": "#68C2DE", "ReportingPerson": 1
    }

```

```

    },
    {
        "Id": 10, "Name": "Daniel Tonini", "Designation": "Project
Lead",
        "ImageUrl": "../content/images/radialtree/Thomas.png",
        "IsExpand": "true",
        "RatingColor": "#93B85A", "ReportingPerson": 1
    },
    {
        "Id": 11, "Name": "Nardo Batista", "Designation": "Project
Lead",
        "ImageUrl": "../content/images/radialtree/Maria.PNG",
        "IsExpand": "true",
        "RatingColor": "#68C2DE", "ReportingPerson": 1
    },
    {
        "Id": 12, "Name": "Michael Holz", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Thomas.PNG",
        "IsExpand": "true",
        "RatingColor": "#68C2DE", "ReportingPerson": 1
    },
    {
        "Id": 13, "Name": "Kloss Perrier", "Designation": "Project
Lead",
        "ImageUrl": "../content/images/radialtree/Clayton.png",
        "IsExpand": "None",
        "RatingColor": "#93B85A", "ReportingPerson": 1
    },
    {
        "Id": 14, "Name": "Liz Nixon", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Jenny.png",
        "IsExpand": "false",
        "RatingColor": "#68C2DE", "ReportingPerson": 1
    },
    {
        "Id": 15, "Name": "Paul Henriot", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Thomas.png",
        "IsExpand": "false",
        "RatingColor": "#D46E89", "ReportingPerson": 1
    },
    {
        "Id": 16, "Name": "Paula Parente", "Designation": "Project
Lead",
        "ImageUrl": "../content/images/radialtree/John.png", "IsExpand":
"None",
        "RatingColor": "#EBB92E", "ReportingPerson": 1
    },
    {
        "Id": 17, "Name": "Matti Kenna", "Designation": "Project Lead",
        "ImageUrl": "../content/images/radialtree/Jenny.png",
        "IsExpand": "None",
        "RatingColor": "#93B85A", "ReportingPerson": 3
    },
    {
        "Id": 18, "Name": "Laura Callahan", "Designation": "Project
Lead",

```

```

    "ImageUrl": "../content/images/radialtree/Robin.png",
    "IsExpand": "false",
    "RatingColor": "#D46E89", "ReportingPerson": 3
  },
  {
    "Id": 19, "Name": "Simon Roel", "Designation": "Project Lead",
    "ImageUrl": "../content/images/radialtree/Clayton.png",
    "IsExpand": "true",
    "RatingColor": "#93B85A", "ReportingPerson": 3
  },
  {
    "Id": 20, "Name": "Thomas Hardy", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/image57.png",
    "IsExpand": "false",
    "RatingColor": "#68C2DE", "ReportingPerson": 3
  },
  {
    "Id": 21, "Name": "Martín Kloss", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/Robin.png",
    "IsExpand": "false",
    "RatingColor": "#93B85A", "ReportingPerson": 3
  },
  {
    "Id": 22, "Name": "Maria Larsson", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/image51.png",
    "IsExpand": "false",
    "RatingColor": "#EBB92E", "ReportingPerson": 3
  },
  {
    "Id": 23, "Name": "Diego Roel", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/image55.png",
    "IsExpand": "false",
    "RatingColor": "#D46E89", "ReportingPerson": 3
  },
  {
    "Id": 24, "Name": "José Pedro ", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/Thomas.png",
    "IsExpand": "true",
    "RatingColor": "#D46E89", "ReportingPerson": 3
  },
  {
    "Id": 25, "Name": "Manu Pereira", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/image56.png",
    "IsExpand": "None",
    "RatingColor": "#D46E89", "ReportingPerson": 3
  },
  {
    "Id": 26, "Name": "Annette Roel", "Designation": "Senior S/w
Engg",
    "ImageUrl": "../content/images/radialtree/image55.png",
    "IsExpand": "false",

```



```

        "RatingColor": "#93B85A", "ReportingPerson": 2
    },
    {
        "Id": 27, "Name": "Catherine Kaff", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/image57.png",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 2
    },
    {
        "Id": 28, "Name": "Lúcia Carvalho", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/Robin.PNG",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 2
    },
    {
        "Id": 29, "Name": "Alej Camino", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/Thomas.PNG",
        "IsExpand": "false",
        "RatingColor": "#93B85A", "ReportingPerson": 2
    },
    {
        "Id": 30, "Name": "Liu Wong", "Designation": "Senior S/w Engg",
        "ImageUrl": "../content/images/radialtree/image57.png",
        "IsExpand": "None",
        "RatingColor": "#D46E89", "ReportingPerson": 2
    },
    {
        "Id": 31, "Name": "Karin Josephs", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/image55.png",
        "IsExpand": "None",
        "RatingColor": "#D46E89", "ReportingPerson": 2
    },
    {
        "Id": 32, "Name": "Ruby Anabela ", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/Thomas.png",
        "IsExpand": "None",
        "RatingColor": "#D46E89", "ReportingPerson": 2
    },
    {
        "Id": 33, "Name": "Pirkko King", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/Robin.png",
        "IsExpand": "None",
        "RatingColor": "#D46E89", "ReportingPerson": 2
    },
    {
        "Id": 34, "Name": "Karl Jablonski", "Designation": "Senior S/w
Engg",
        "ImageUrl": "../content/images/radialtree/image53.png",
        "IsExpand": "None",
        "RatingColor": "#D46E89", "ReportingPerson": 2
    },
    },

```

```

];
var items = new ej.data.DataManager(data , new ej.data.Query().take(5));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '590px',
  snapSettings: { constraints: 0 },
  layout: {
    type: 'RadialTree', horizontalSpacing: 30, verticalSpacing: 30,
  },
  root: 'parent'
}, {
  dataSourceSettings: {
    id: 'Id', parentId: 'ReportingPerson', dataManager: items
  },
  getNodeDefaults: (obj, diagram) => {
    obj.height = 15;
    obj.width = 15;
    obj.backgroundColor = 'lightgrey';
    obj.style = { fill: 'transparent', strokeWidth: 2 };
    return obj;
  },
  getConnectorDefaults: (connector, diagram) => {
    connector.targetDecorator.shape = 'None';
    connector.type = 'Straight';
    return connector;
  },
});
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>

```

Organizational Chart

An organizational chart is a diagram that displays the structure of an organization and relationships. To create an organizational chart, the [type](#) of layout should be set as an [OrganizationalChart](#).

The following code example illustrates how to create an organizational chart.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
  { Id: "parent", Role: "Project Management" },
  { Id: 1, Role: "R&D Team", Team: "parent" },
  { Id: 3, Role: "Philosophy", Team: "1" },
  { Id: 4, Role: "Organization", Team: "1" },
  { Id: 5, Role: "Technology", Team: "1" },
  { Id: 7, Role: "Funding", Team: "1" },
  { Id: 8, Role: "Resource Allocation", Team: "1" },
  { Id: 9, Role: "Targeting", Team: "1" },
  { Id: 11, Role: "Evaluation", Team: "1" },
  { Id: 156, Role: "HR Team", Team: "parent" },
  { Id: 13, Role: "Recruitment", Team: "156" },
  { Id: 113, Role: "Training", Team: "12" },
  { Id: 112, Role: "Employee Relation", Team: "156" },
  { Id: 14, Role: "Record Keeping", Team: "12" },
  { Id: 15, Role: "Compensations & Benefits", Team: "12" },
  { Id: 16, Role: "Compliances", Team: "12" },
  { Id: 17, Role: "Production & Sales Team", Team: "parent" },
  { Id: 119, Role: "Design", Team: "17" },
  { Id: 19, Role: "Operation", Team: "17" },
  { Id: 20, Role: "Support", Team: "17" },
  { Id: 21, Role: "Quality Assurance", Team: "17" },
  { Id: 23, Role: "Customer Interaction", Team: "17" },
  { Id: 24, Role: "Support and Maintenance", Team: "17" },
  { Id: 25, Role: "Task Coordination", Team: "17" }
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(5));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '550px',
  layout: { type: 'OrganizationalChart' },
  dataSourceSettings: { id: 'Id', parentId: 'Team', dataManager: items },
  getNodeDefaults: (obj) => {
    obj.shape = { type: 'Text', content: (obj.data).Role };
  }
});

```

```

    obj.style = { fill: 'None', strokeColor: 'none',strokeWidth:
2,bold:true,color:'white' };
    obj.borderColor = 'black';
    obj.backgroundColor = 'darkcyan';
    obj.width = 75;
    obj.height = 40;
    obj.borderWidth = 1;
    (obj.shape).margin = { left: 5, right: 5, top: 5, bottom: 5 };
    return obj;
}, getConnectorDefaults: (connector, diagram) => {
    connector.type = 'Orthogonal';
    return connector;
}
});
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>
```

Organizational chart layout starts parsing from root and iterate through all its child elements. The `getLayoutInfo` method provides necessary information of a node's children and the way to arrange (direction, orientation, offsets, etc.) them. The arrangements can be customized by overriding this function as explained.

GetLayoutInfo

Set chart orientations, chart types, and offset to be left between parent and child nodes by overriding the method, `diagram.layout.getLayoutInfo`. The `getLayoutInfo` method is called to configure every subtree of the organizational chart. It takes the following arguments.

- node: Parent node to that options are to be customized.
- options: Object to set the customizable properties.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
  {
    'Id': 'parent', 'Name': 'Maria Anders', 'Designation': 'Managing
Director',
    'IsExpand': 'true', 'RatingColor': '#C34444'
  },
  {
    'Id': 1, 'Name': 'Ana Trujillo', 'Designation': 'Project Manager',
    'IsExpand': 'false',
    'RatingColor': '#68C2DE', 'ReportingPerson': 'parent'
  },
  {
    'Id': 2, 'Name': 'Anto Moreno', 'Designation': 'Project Lead',
    'IsExpand': 'false',
    'RatingColor': '#93B85A', 'ReportingPerson': 'parent'
  }
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
diagram = new ej.diagrams.Diagram({
  snapSettings: { constraints: 0 },
  layout: {
    type: 'OrganizationalChart', margin: { top: 20 },
    getLayoutInfo: (node, option) => {
      if (!option.hasSubTree) {
        option.orientation = 'Vertical';
        option.type = 'Alternate';
      }
    }
  },
  dataSourceSettings: {
    id: 'Id', parentId: 'ReportingPerson', dataManager: items
  },
});
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>

```

The following table illustrates the properties that “options” argument takes.

Property	Description	Default Value
----------	-------------	---------------

Property	Description	Default Value
----------	-------------	---------------

options.assistants	By default, the collection is empty. When any of the child nodes have to be set as Assistant , you can remove from children collection and have to insert into assistants collection.	Empty array
--------------------	--	-------------

options.orientation	Gets or sets the organizational chart orientation.	SubTreeOrientation.Vertical
---------------------	--	-----------------------------

|options.type|Gets or sets the chart organizational chart type. |For horizontal chart orientation:SubTreeAlignments.Center and for vertical chart orientation:SubTreeAlignments.Alternate|

|options.offset|Offset is the horizontal space to be left between parent and child nodes. |20 pixels applicable only for vertical chart orientations. |

|options.hasSubTree|Gets whether the node contains subtrees. |Boolean|

|options.level|Gets the depth of the node from layout root. |Number|

|options.enableRouting|By default, connections are routed based on the chart type and orientations. This property gets or sets whether default routing is to be enabled or disabled. |true|

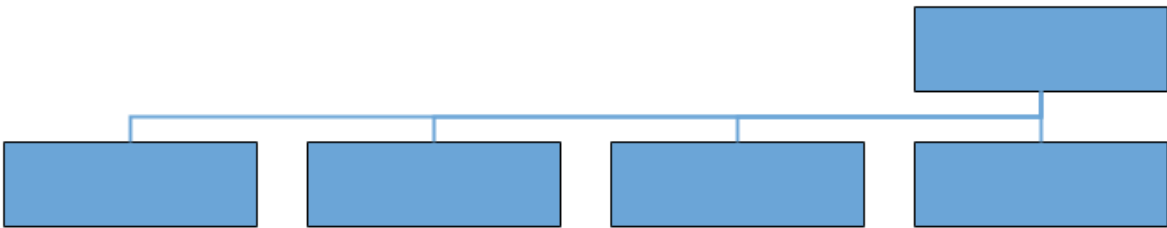
|options.rows|Sets the number of rows on which the child nodes will be arranged. Applicable only for balanced type horizontal tree. |Number|

The following table illustrates the different chart orientations and chart types.

|Orientation|Type|Description|Example|

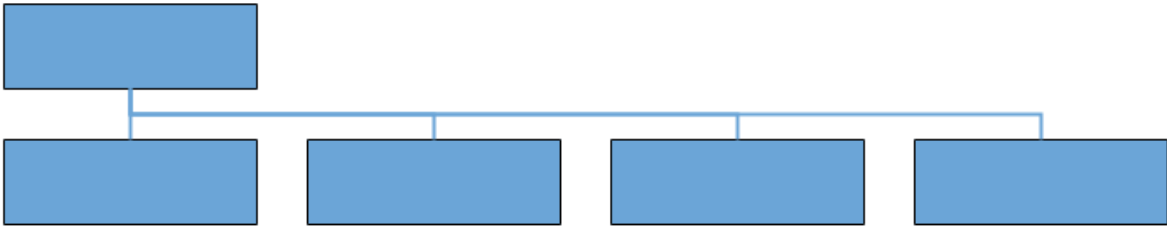
|-----|-----|-----|-----|

|Horizontal|Left|Arranges the child nodes horizontally at the left side of the parent. |



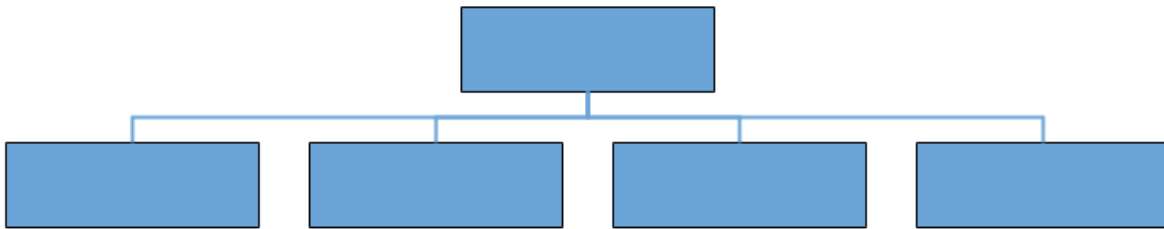
|

|Right|Arranges the child nodes horizontally at the right side of the parent. |



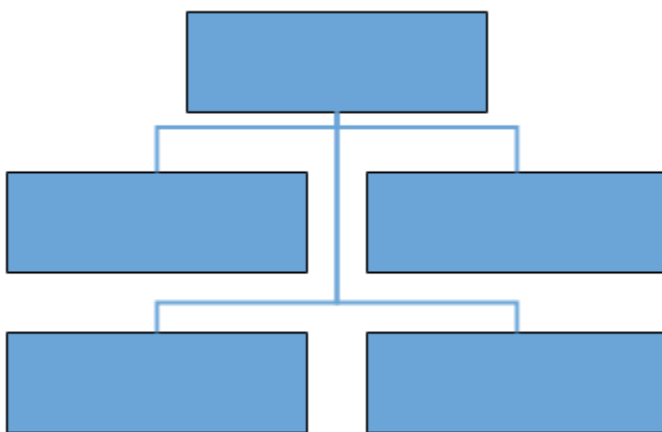
|

||Center|Arranges the children like standard tree layout orientation.|



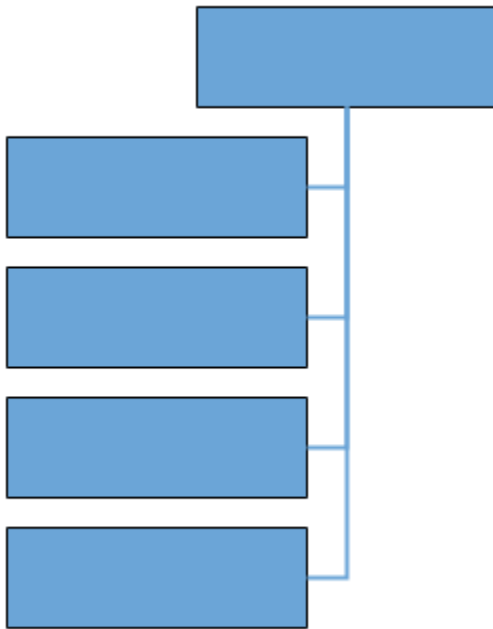
|

||Balanced|Arranges the leaf level child nodes in multiple rows.|



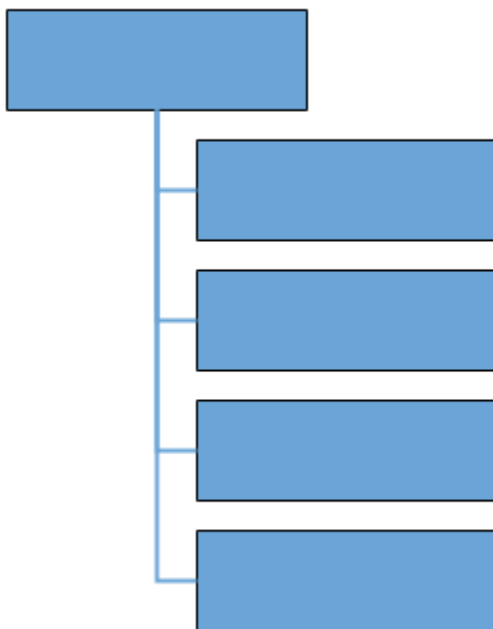
|

|Vertical|Left|Arranges the children vertically at the left side of the parent.|



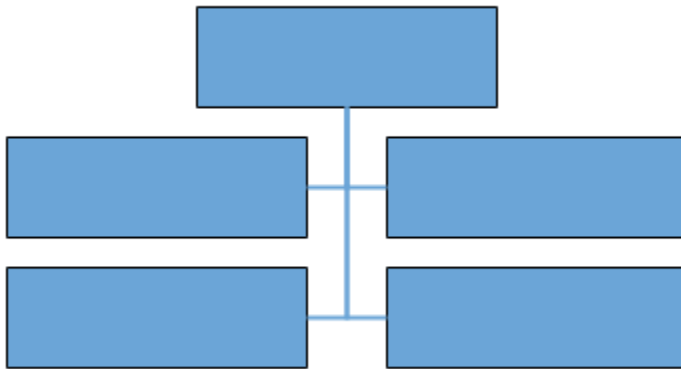
|

||Right|Arranges the children vertically at the right side of the parent.|



|

| |Alternate|Arranges the children vertically at both left and right sides of the parent.|



The following code example illustrates how to set the vertical right arrangement to the leaf level trees.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [{ Id: "parent", Role: "Board" },
  { Id: "1", Role: "General Manager", Manager: "parent" },
  { Id: "2", Role: "Human Resource Manager", Manager: "1" },
  { Id: "3", Role: "Trainers", Manager: "2" },
  { Id: "4", Role: "Recruiting Team", Manager: "2" },
  { Id: "5", Role: "Finance Asst. Manager", Manager: "2" },
  { Id: "6", Role: "Design Manager", Manager: "1" },
  { Id: "7", Role: "Design Supervisor", Manager: "6" },
  { Id: "8", Role: "Development Supervisor", Manager: "6" },
  { Id: "9", Role: "Drafting Supervisor", Manager: "6" },
  { Id: "10", Role: "Marketing Manager", Manager: "1" },
  { Id: "11", Role: "Oversea sales Manager", Manager: "10" },
  { Id: "12", Role: "Petroleum Manager", Manager: "10" },
  { Id: "13", Role: "Service Dept. Manager", Manager: "10" },
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '530px',
  snapSettings: { constraints: 0 },
  layout: {
    type: 'OrganizationalChar',
    getLayoutInfo: (node, options) => {
      if (node.data['Role'] === 'General Manager') {
        options.assistants.push(options.children[0]);
        options.children.splice(0, 1);
      }
      if (!options.hasSubTree) {
        options.type = 'Right';
        options.orientation = 'Vertical';
      }
    }
  },
  dataSourceSettings: {

```

```

        id: 'Id', parentId: 'Manager', dataManager: items
    },
    getNodeDefaults: (obj, diagram) => {
        obj.width = 150;
        obj.height = 50;
        obj.style.fill = 'darkcyan';
        obj.annotations = [{ content: obj.data['Role'], style: { color:
'white' } }];
        return obj;
    }, getConnectorDefaults: (connector, diagram) => {
        connector.targetDecorator.shape = 'None';
        connector.type = 'Orthogonal';
        return connector;
    }
});
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>
```

Assistant

Assistants are child item that have a different relationship with the parent node. They are laid out in a dedicated part of the tree. A node can be specified as an assistant of its parent by adding it to the `assistants` property of the argument "options".

The following code example illustrates how to add assistants to layout.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [{ Id: 1, Role: "General Manager" },
  { Id: 2, Role: "Assistant Manager", Team: 1 },
  { Id: 3, Role: "Human Resource Manager", Team: 1 },
  { Id: 4, Role: "Design Manager", Team: 1 },
  { Id: 5, Role: "Operation Manager", Team: 1 },
  { Id: 6, Role: "Marketing Manager", Team: 1 }
];
var items= new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '530px',
  snapSettings: { constraints: 0 },
  layout: {
    type: 'OrganizationalChart',
    getLayoutInfo: (node, options) => {
      if (node.data['Role'] === 'General Manager') {
        options.assistants.push(options.children[0]);
        options.children.splice(0, 1);
      }
      if (!options.hasSubTree) {
        options.type = 'Center';
        options.orientation = 'Horizontal';
      }
    }
  },
  dataSourceSettings: {
    id: 'Id', parentId: 'Team', dataManager: items
  },
  getNodeDefaults: (obj, diagram) => {
    obj.width = 150;
    obj.height = 50;
    obj.style.fill = 'darkcyan';
    obj.annotations = [{ content: obj.data['Role'], style: { color:
'white' } }];
    return obj;
  },
  getConnectorDefaults: (connector, diagram) => {
    connector.targetDecorator.shape = 'None';
    connector.type = 'Orthogonal';
    return connector;
  }
});
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>

```

Symmetric layout

The symmetric layout has been formed using nodes position by closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state.

The layout's [springLength](#) defined as how long edges should be, ideally. This will be the resting length for the springs. Edge attraction and vertex repulsion forces to be defined by using layout's [springFactor](#), the more sibling nodes repel each other. The relative positions do not change any more from one iteration to the next. The number of iterations can be specified by using layout's [maxIteration](#).

The following code illustrates how to arrange the nodes in a radial tree structure.

Note: If you want to use symmetric layout in diagram, you need to inject SymmetricLayout in the diagram.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var nodes = [];
var connectors = [];
function ConnectNodes(parentNode, childNode) {
    var connector = {
        id: parentNode.id + childNode.id,
        sourceID: parentNode.id,
        targetID: childNode.id,
        targetDecorator: { shape: 'None' }
    }
    return connector;
}
function GetRectangle(name) {
    var shape = { type: 'Basic', shape: 'Ellipse' };
    var node = { id: name, height: 25, width: 25, borderColor: '#5e5e5e',
borderWidth: 1, style: { fill: '#ff6329' }, shape: shape };
    return node;
}
function populateNodes() {
    var parentRect = GetRectangle("p");
    nodes.push(parentRect);
    for (var i = 0; i < 2; i++) {
        var childRect_i = GetRectangle("c" + i);
        nodes.push(childRect_i);
        for (var j = 0; j < 2; j++) {
            var childRect_j = GetRectangle("c" + i + j);
            nodes.push(childRect_j);
            for (var k = 0; k < 6; k++) {
                var childRect_k = GetRectangle("c" + i + j + k);
                nodes.push(childRect_k);
                connectors.push(ConnectNodes(childRect_j, childRect_k));
            }
            connectors.push(ConnectNodes(childRect_i, childRect_j));
        }
        connectors.push(ConnectNodes(parentRect, childRect_i));
    }
    return nodes;
}
populateNodes();
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'SymmetricalLayout', springLength: 80, springFactor:
0.8, maxIteration: 500, margin: { left: 20, top: 20 } },
    nodes: nodes, connectors: connectors,
});
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>

```

Mind Map layout

A mind map is a diagram that displays the nodes as a spider diagram organizes information around a central concept. To create mind map, the [type](#) of layout should be set as **MindMap**.

Tree Orientation in layout

An [Orientation](#) of a **MindMapTreeLayout** is used to arrange the tree layout according to a specific direction. By default, the orientation is set to Horizontal. The following table outlines the various orientation types available:

Orientation Type	Description
Horizontal	Aligns the tree layout from left to right
Vertical	Aligns the tree layout from top to bottom

The following code example illustrates how to create an mindmap layout.

Note: If you want to use mind map layout in diagram, you need to inject MindMap in the diagram.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var nodes = [];
var connectors = [];
function ConnectNodes(parentNode, childNode) {
    var connector = {
        id: parentNode.id + childNode.id,
        sourceID: parentNode.id,
        targetID: childNode.id,
        targetDecorator: { shape: 'None' }
    }
    return connector;
}
function GetRectangle(name) {
    var shape = { type: 'Basic', shape: 'Ellipse' };
    var node = { id: name, height: 25, width: 25, borderColor: '#5e5e5e',
borderWidth: 1, style: { fill: '#ff6329' }, shape: shape };
    return node;
}
function populateNodes() {
    var parentRect = GetRectangle("p");
    nodes.push(parentRect);
    for (var i = 0; i < 2; i++) {
        var childRect_i = GetRectangle("c" + i);
        nodes.push(childRect_i);
        for (var j = 0; j < 2; j++) {
            var childRect_j = GetRectangle("c" + i + j);
            nodes.push(childRect_j);
            for (var k = 0; k < 6; k++) {
                var childRect_k = GetRectangle("c" + i + j + k);
                nodes.push(childRect_k);
                connectors.push(ConnectNodes(childRect_j, childRect_k));
            }
            connectors.push(ConnectNodes(childRect_i, childRect_j));
        }
        connectors.push(ConnectNodes(parentRect, childRect_i));
    }
    return nodes;
}
populateNodes();
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'SymmetricalLayout', springLength: 80, springFactor:
0.8, maxIteration: 500, margin: { left: 20, top: 20 } },
    nodes: nodes, connectors: connectors,
});
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>

```

Complex hierarchical tree

Complex hierarchical tree layout is the extended version of the hierarchical tree layout. The child had been two or more parents. To create a complex hierarchical tree, the [type](#) of layout should be set as `ComplexHierarchicalTree`.

Note: If you want to use Complex hierarchical layout in diagram, you need to inject `ComplexHierarchicalTree` in the diagram.

The following code example illustrates how to create a complex hierarchical tree.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var nodes = [];
var connectors = [];
function ConnectNodes(parentNode, childNode) {
    var connector = {
        id: parentNode.id + childNode.id,
        sourceID: parentNode.id,

```

```

        targetID: childNode.id,
        targetDecorator: { shape: 'None' }
    }
    return connector;
}
function GetRectangle(name) {
    var shape = { type: 'Basic', shape: 'Ellipse' };
    var node = { id: name, height: 25, width: 25, borderColor: '#5e5e5e',
borderWidth: 1, style: { fill: '#ff6329' }, shape: shape };
    return node;
}
function populateNodes() {
    var parentRect = GetRectangle("p");
    nodes.push(parentRect);
    for (var i = 0; i < 2; i++) {
        var childRect_i = GetRectangle("c" + i);
        nodes.push(childRect_i);
        for (var j = 0; j < 2; j++) {
            var childRect_j = GetRectangle("c" + i + j);
            nodes.push(childRect_j);
            for (var k = 0; k < 6; k++) {
                var childRect_k = GetRectangle("c" + i + j + k);
                nodes.push(childRect_k);
                connectors.push(ConnectNodes(childRect_j, childRect_k));
            }
            connectors.push(ConnectNodes(childRect_i, childRect_j));
        }
        connectors.push(ConnectNodes(parentRect, childRect_i));
    }
    return nodes;
}
populateNodes();
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'SymmetricalLayout', springLength: 80, springFactor:
0.8, maxIteration: 500, margin: { left: 20, top: 20 } },
    nodes: nodes, connectors: connectors,
});
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>

```

Line Distribution

Line distribution is used to arrange the connectors without overlapping in automatic layout. In some cases, the automatic layout connectors connecting to the nodes will be overlapped with one another. So user can decide whether the segment of each connector from a single parent node should be same point or different point. The [connectionPointOrigin](#) property of layout is used to enable or disable the line distribution in layout. By default ConnectionPointOrigin will be [SamePoint](#).

The following code example illustrates how to create a complex hierarchical tree with line distribution.

Note: If you want to use line distribution in diagram layout, you need to inject `LineDistribution` module in the diagram.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.ComplexHierarchicalTree, ej.diagrams.LineDistribution );
var data = [
    { "Name": "node11" },
    { "Name": "node12", "ReportingPerson": ["node114"] },
    { "Name": "node13", "ReportingPerson": ["node12"] },
    { "Name": "node14", "ReportingPerson": ["node12"] },
    { "Name": "node15", "ReportingPerson": ["node12"] },
    { "Name": "node116", "ReportingPerson": ["node22", "node12"] },
    { "Name": "node16", "ReportingPerson": [] },
    { "Name": "node18", "ReportingPerson": [] },
    { "Name": "node21" },
    { "Name": "node22", "ReportingPerson": ["node114"] },
    { "Name": "node23", "ReportingPerson": ["node22"] },
    { "Name": "node24", "ReportingPerson": ["node22"] },
    { "Name": "node25", "ReportingPerson": ["node22"] },

```

```

        { "Name": "node26", "ReportingPerson": [] },
        { "Name": "node28", "ReportingPerson": [] },
        { "Name": "node31" },
        { "Name": "node114", "ReportingPerson": ["node11", "node21",
"node31"]}
    ];
    var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
    var diagram = new ej.diagrams.Diagram({
        width: '100%',
        height: '590px',
        //Uses layout to auto-arrange nodes on the diagram page
        layout: {
            //Sets layout type
            type: 'ComplexHierarchicalTree',
            connectionPointOrigin:
ej.diagrams.ConnectionPointOrigin.DifferentPoint,
            horizontalSpacing: 40, verticalSpacing: 40, horizontalAlignment:
"Left", verticalAlignment: "Top",
            margin: { left: 0, right: 0, top: 0, bottom: 0 },
            orientation: 'TopToBottom'
        }, //Configures data source for diagram
        dataSourceSettings: {
            id: 'Name',
            parentId: 'ReportingPerson',
            dataManager: items
        }, //Sets the default properties for nodes
        getNodeDefaults: (obj) => {
            obj.width = 40; obj.height = 40;
            obj.shape = { type: 'Basic', shape: 'Rectangle', cornerRadius: 7 };
            obj.style = { fill: '#6BA5D7', strokeColor: 'none', strokeWidth: 2
};

            obj.borderWidth = 1;
            obj.backgroundColor = '#6BA5D7';
            return obj;
        }, //Sets the default properties for and connectors
        getConnectorDefaults: (connector, diagram) => {
            connector.type = 'Orthogonal';
            connector.cornerRadius = 7;
            connector.targetDecorator.height = 7;
            connector.targetDecorator.width = 7;
            connector.style = { strokeColor: '#6BA5D7', strokeWidth: 1 };
            connector.targetDecorator.style.fill = '#6BA5D7';
            connector.targetDecorator.style.strokeColor = '#6BA5D7';
            return connector;
        }
    });
    diagram.appendTo('#element');
    diagram.fitToPage({ mode: 'Width' });

```

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>

```

Linear Arrangement

Linear arrangement is used to linearly arrange the child nodes in layout, which means the parent node is placed in the center corresponding to its children. When line distribution is enabled, linear arrangement is also activated by default. The [arrangement](#) property of layout is used to enable or disable the linear arrangement in layout. By default arrangement will be **Nonlinear**.

Note: If you want to use linear arrangement in diagram layout, you need to inject `LineDistribution` module in the diagram. Linear arrangement is applicable only for complex hierarchical tree layout.

The following code illustrates how to allow a linear arrangement in diagram layout.

```

`ts
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
width: '100%', height: '590px',
layout: {
type: 'ComplexHierarchicalTree',

```

```
//To arrange a child nodes in a linear manner
arrangement: ChildArrangement.Linear,
horizontalSpacing: 40, verticalSpacing: 40,
orientation: 'TopToBottom',
},
},'#diagram');
`ts
```

Prevent connectors overlay

The below constraints prevents the connector segments overlapping nodes with a complex hierarchical layout.

```
`ts
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
width: '100%', height: '590px',
layout: {
//this prevents connector segments overlapping
enableRouting: true,
},
},'#diagram');
`ts
```

Customize layout

Orientation, spacings, and position of the layout can be customized with a set of properties.

To explore layout properties, refer to [Layout Properties](#).

Layout bounds

Diagram provides support to align the layout within any custom rectangular area. For more information about bounds, refer to [bounds](#).

Layout alignment

The layout can be aligned anywhere over the layout bounds/viewport using the [horizontalAlignment](#) and [verticalAlignment](#) properties of the layout.

The following code illustrates how to align the layout at the top-left of the layout bounds.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.ComplexHierarchicalTree, ej.diagrams.LineDistribution );
var data = [
  { "Name": "node11" },
  { "Name": "node12", "ReportingPerson": ["node114"] },
  { "Name": "node13", "ReportingPerson": ["node12"] },
  { "Name": "node14", "ReportingPerson": ["node12"] },
```

```

    { "Name": "node15", "ReportingPerson": ["node12"] },
    { "Name": "node116", "ReportingPerson": ["node22", "node12"] },
    { "Name": "node16", "ReportingPerson": [] },
    { "Name": "node18", "ReportingPerson": [] },
    { "Name": "node21" },
    { "Name": "node22", "ReportingPerson": ["node114"] },
    { "Name": "node23", "ReportingPerson": ["node22"] },
    { "Name": "node24", "ReportingPerson": ["node22"] },
    { "Name": "node25", "ReportingPerson": ["node22"] },
    { "Name": "node26", "ReportingPerson": [] },
    { "Name": "node28", "ReportingPerson": [] },
    { "Name": "node31" },
    { "Name": "node114", "ReportingPerson": ["node11", "node21",
"node31"]}
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%',
    height: '590px',
    //Uses layout to auto-arrange nodes on the diagram page
    layout: {
        //Sets layout type
        type: 'ComplexHierarchicalTree',
        connectionPointOrigin:
ej.diagrams.ConnectionPointOrigin.DifferentPoint,
        horizontalSpacing: 40, verticalSpacing: 40, horizontalAlignment:
"Left", verticalAlignment: "Top",
        margin: { left: 0, right: 0, top: 0, bottom: 0 },
        orientation: 'TopToBottom'
    }, //Configures data source for diagram
    dataSourceSettings: {
        id: 'Name',
        parentId: 'ReportingPerson',
        dataManager: items
    }, //Sets the default properties for nodes
    getNodeDefaults: (obj) => {
        obj.width = 40; obj.height = 40;
        obj.shape = { type: 'Basic', shape: 'Rectangle', cornerRadius: 7 };
        obj.style = { fill: '#6BA5D7', strokeColor: 'none', strokeWidth: 2
};

        obj.borderWidth = 1;
        obj.backgroundColor = '#6BA5D7';
        return obj;
    }, //Sets the default properties for and connectors
    getConnectorDefaults: (connector, diagram) => {
        connector.type = 'Orthogonal';
        connector.cornerRadius = 7;
        connector.targetDecorator.height = 7;
        connector.targetDecorator.width = 7;
        connector.style = { strokeColor: '#6BA5D7', strokeWidth: 1 };
        connector.targetDecorator.style.fill = '#6BA5D7';
        connector.targetDecorator.style.strokeColor = '#6BA5D7';
        return connector;
    }
});
diagram.appendTo('#element');
diagram.fitToPage({ mode: 'Width' });

```

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>
```

Layout spacing

Layout provides support to add space horizontally and vertically between the nodes. The [horizontalSpacing](#) and [verticalSpacing](#) properties of the layout allows you to set the space between the nodes in horizontally and vertically.

Layout margin

Layout provides support to add some blank space between the layout bounds/viewport and the layout. The [margin](#) property of the layout allows you to set the blank space.

The following code illustrates how to set the layout margin.

INDEX.JS


```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
    {Name: "Steve-Ceo"},
    {Name: "Kevin-Manager", ReportingPerson: "Steve-Ceo"},
    {Name: "Peter-Manager", ReportingPerson: "Kevin-Manager"},
    {Name: "John- Manager", ReportingPerson: "Peter-Manager"},
    {Name: "Mary-CSE ", ReportingPerson: "Peter-Manager"},
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'HierarchicalTree', bounds: new
ej.diagrams.Rect(0,0,500,500),horizontalSpacing:25,verticalSpacing:30,horizo
ntalAlignment:'Left',verticalAlignment:'Top' },
    dataSourceSettings: { id: 'Name', parentId: 'ReportingPerson',
dataManager: items },
    getNodeDefaults: (obj) => {
        obj.shape = { type: 'Text', content: (obj.data ).Name };
        obj.style = { fill:'None', strokeColor: 'none',strokeWidth: 2, bold:
true, color: 'white'};
        obj.borderColor = 'black';
        obj.width=100;
        obj.height=40;
        obj.backgroundColor = 'darkcyan';
        obj.borderWidth = 1;
        (obj.shape).margin = { left: 25, right: 25, top: 25, bottom: 25 };
        return obj;
    }, getConnectorDefaults: (connector, diagram) => {
        connector.type = 'Orthogonal';
        return connector;
    }
});
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>

```

Layout orientation

The layout orientation can be used to arrange the layout based on the direction. There are different orientation types that are defined in the following table.

Orientation	Description
----- -----	
TopToBottom	Aligns the layout from top to bottom. All the roots are placed at top of diagram.
LeftToRight	Aligns the layout from left to right. All the roots are placed at left of diagram.
BottomToTop	Aligns the layout from bottom to top. All the roots are placed at bottom of the diagram.
RightToLeft	Aligns the layout from right to left. All the roots are placed at right of the diagram.

Diagram provides support to customize the [orientation](#) of layout. You can set the desired orientation using `layout.orientation`.

Note: In the diagram the default orientation is `TopToBottom`.

The following code illustrates how to arrange the nodes in a `BottomToTop` orientation.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data= [
    {Name: "Steve-Ceo"},
    {Name: "Kevin-Manager", ReportingPerson: "Steve-Ceo"},
    {Name: "Peter-Manager", ReportingPerson: "Steve-Ceo"},
    {Name: "John- Manager", ReportingPerson: "Peter-Manager"},
    {Name: "Mary-CSE ", ReportingPerson: "Peter-Manager"},
    {Name: "Jim-CSE ", ReportingPerson: "Kevin-Manager"},
    {Name: "Martin-CSE", ReportingPerson: "Kevin-Manager"}];

```

```

var items = new ej.data.DataManager(data , new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'HierarchicalTree', bounds: new
ej.diagrams.Rect(0,0,500,500), horizontalSpacing:25, verticalSpacing:30, horizontalAlignment:'Left', verticalAlignment:'Top', orientation:'BottomToTop'},
    dataSourceSettings: { id: 'Name', parentId: 'ReportingPerson',
dataManager: items },
    getNodeDefaults: (obj) => {
        obj.shape = { type: 'Text', content: (obj.data).Name };
        obj.style = { fill:'None', strokeColor: 'none', strokeWidth: 2, bold:
true, color: 'white'};
        obj.borderColor = 'black';
        obj.width=100;
        obj.height=40;
        obj.backgroundColor = 'darkcyan';
        obj.borderWidth = 1;
        (obj.shape).margin = { left: 25, right: 25, top: 25, bottom: 25 };
        return obj;
    }, getConnectorDefaults: (connector, diagram) => {
        connector.type = 'Orthogonal';
        return connector;
    }
});
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>

```

Fixed node

Layout provides support to arrange the nodes with reference to the position of a fixed node and set it to the [fixedNode](#) of the layout property. This is helpful when you try to expand/collapse a node. It might be expected that the position of the double-clicked node should not be changed.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data= [
  {Name: "Steve-Ceo"},
  {Name: "Kevin-Manager", ReportingPerson: "Steve-Ceo"},
  {Name: "Peter-Manager", ReportingPerson: "Steve-Ceo"},
  {Name: "John- Manager", ReportingPerson: "Peter-Manager"},
  {Name: "Mary-CSE ", ReportingPerson: "Peter-Manager"},
  {Name: "Jim-CSE ", ReportingPerson: "Kevin-Manager"},
  {Name: "Martin-CSE", ReportingPerson: "Kevin-Manager"}];
var items = new ej.data.DataManager(data , new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
  width: '100%', height: '550px',
  layout: { type: 'HierarchicalTree', bounds: new
ej.diagrams.Rect(0, 0, 500, 500), horizontalSpacing: 25, verticalSpacing: 30, horizontalAlignment: 'Left', verticalAlignment: 'Top', orientation: 'BottomToTop'},
  dataSourceSettings: { id: 'Name', parentId: 'ReportingPerson',
  dataManager: items },
  getNodeDefaults: (obj) => {
    obj.shape = { type: 'Text', content: (obj.data).Name };
    obj.style = { fill: 'None', strokeColor: 'none', strokeWidth: 2, bold:
true, color: 'white'};
    obj.borderColor = 'black';
    obj.width=100;
    obj.height=40;
    obj.backgroundColor = 'darkcyan';
    obj.borderWidth = 1;
    (obj.shape ).margin = { left: 25, right: 25, top: 25, bottom: 25 };
    return obj;
  }, getConnectorDefaults: (connector, diagram) => {
    connector.type = 'Orthogonal';
    return connector;
  }
});
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>

```

Expand and collapse

Diagram allows to expand/collapse the subtrees of a layout. The node's `isExpanded` property allows you to expand/collapse its children. The following code example shows how to expand/collapse the children of a node.

INDEX.JS

```

ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
  {
    'Id': 'parent1', 'Name': 'Maria ', 'Designation': 'Managing
Director',

```

```

        'ImageUrl': '../content/images/orgchart/Clayton.png',
        'IsExpand': true, 'RatingColor': '#C34444'
    },
    {
        'Id': 'parent', 'Name': ' sam', 'Designation': 'Managing
        Director', 'ReportingPerson': 'parent1',
        'ImageUrl': '../content/images/orgchart/Clayton.png',
        'IsExpand': true, 'RatingColor': '#C34444'
    },
    {
        'Id': 'parent3', 'Name': ' sam geo', 'Designation': 'Managing
        Director', 'ReportingPerson': 'parent1',
        'ImageUrl': '../content/images/orgchart/Clayton.png',
        'IsExpand': true, 'RatingColor': '#C34444'
    },
    {
        'Id': '80', 'Name': ' david', 'Designation': 'Managing
        Director', 'ReportingPerson': 'parent3',
        'ImageUrl': '../content/images/orgchart/Clayton.png',
        'IsExpand': true, 'RatingColor': '#C34444'
    },
    {
        'Id': '82', 'Name': ' pirlo', 'Designation': 'Managing
        Director', 'ReportingPerson': 'parent',
        'ImageUrl': '../content/images/orgchart/Clayton.png',
        'IsExpand': true, 'RatingColor': '#C34444'
    }
];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '590px', selectedItems: { constraints:
    ~ej.diagrams.SelectorConstraints.ResizeAll },
    snapSettings: { constraints: 0 },
    layout: {
        enableAnimation: true,
        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, diagram) => {
        obj.expandIcon = { height: 15, width: 15, shape: "Plus", fill:
        'lightgray', offset: { x: .5, y: .85 } }
        obj.collapseIcon.offset = { x: .5, y: .85 }
        obj.collapseIcon.height = 15;
        obj.collapseIcon.width = 15;
        obj.collapseIcon.shape = "Minus";
        obj.collapseIcon.fill = 'lightgray';
        obj.height = 50;
        obj.backgroundColor = 'lightgrey';
        obj.style = { fill: 'transparent', strokeWidth: 2 };
    }
});

```

```
        return obj;
    }, getConnectorDefaults: (connector, diagram) => {
        connector.style = { strokeColor: '#6BA5D7', strokeWidth: 2 };
        connector.targetDecorator.style.fill = '#6BA5D7';
        connector.targetDecorator.style.strokeColor = 'white';
        connector.targetDecorator.shape = 'None';
        connector.type = 'Orthogonal';
        return connector;
    },
    });
    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>
```

In the previous example, while expanding/collapsing a node, it is set as fixed node in order to prevent it from repositioning.

[Refresh layout](#)

Diagram allows to refresh the layout at runtime. To refresh the layout, refer to Refresh layout.

[setNodeTemplate](#)

The setNodeTemplate function is provided for the purpose of customizing nodes. It will be called for each node on node initialization. In this function, the node style and its properties can be customized and can bind the custom JSON with node.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding,
ej.diagrams.HierarchicalTree);
var data = [
    {Name: "Steve-Ceo"},
    {Name: "Kevin-Manager", ReportingPerson: "Steve-
Ceo",color:'darkcyan'},
    {Name: "Peter-Manager", ReportingPerson: "Steve-Ceo",color:'white'},
    {Name: "John- Manager", ReportingPerson: "Peter-
Manager",color:'darkcyan'},
    {Name: "Mary-CSE ", ReportingPerson: "Peter-Manager",color:'white'},
    {Name: "Jim-CSE ", ReportingPerson: "Kevin-
Manager",color:'darkcyan'},
    {Name: "Martin-CSE", ReportingPerson: "Kevin-
Manager",color:'white'}];
var items = new ej.data.DataManager(data, new ej.data.Query().take(7));
var diagram = new ej.diagrams.Diagram({
    width: '100%', height: '550px',
    layout: { type: 'HierarchicalTree',bounds:new
ej.diagrams.Rect(0,0,500,500),horizontalSpacing:25,verticalSpacing:30,horizo
ntalAlignment:'Left',verticalAlignment:'Top'},
    dataSourceSettings: { id: 'Name', parentId: 'ReportingPerson',
dataManager: items },
    getNodeDefaults: (obj) => {
        obj.shape = { type: 'Text', content: (obj.data ).Name };
        obj.style = { fill:'None', strokeColor: 'none',strokeWidth: 2, bold:
true, color: 'white'};
        obj.borderColor = 'black';
        obj.width=100;
        obj.height=40;
        obj.backgroundColor = 'darkcyan';
        obj.borderWidth = 1;
        (obj.shape ).margin = { left: 25, right: 25, top: 25, bottom: 25 };
        return obj;
    }, setNodeTemplate: function (obj, diagram) {
        obj.style.color=(obj.data).color;
    }
});
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>

```

Accessibility in EJ2 JavaScript 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](#)

Commands in EJ2 JavaScript Diagram control

<!-- markdownlint-disable MD010 -->

There are several commands available in the diagram as follows.

- Alignment commands
- Spacing commands
- Sizing commands
- Clipboard commands
- Grouping commands
- Z-order commands
- Zoom commands
- Nudge commands
- FitToPage commands
- Undo/Redo commands

Align

Alignment commands enable you to align the selected or defined objects such as nodes and connectors with respect to the selection boundary. Refer to [align](#) commands which shows how to use align methods in the diagram.

<!-- markdownlint-disable MD033 -->

Parameters	Description	
:-----	:-----:	
['Alignment Options'](..api/diagram/alignmentOptions#AlignmentOptions)		

Defines the specific direction, with respect to which the objects to be aligned.
The accepted values of the argument "alignment options" are as follows.

Left Aligns all the selected objects at the left of the selection boundary.

Right Aligns all the selected objects at the right of the selection boundary.

Center Aligns all the selected objects at the center of the selection boundary.

Top Aligns all the selected objects at the top of the selection boundary.

Bottom Aligns all the selected objects at the bottom of the selection boundary.

Middle Aligns all the selected objects at the middle of the selection boundary.

|

| Objects |

<p align="left">Defines the objects to be aligned. This is an optional parameter. By default, all the nodes and connectors in the selected region of the diagram gets aligned.</p> |

[`Alignment Mode`](../api/diagram/alignmentMode#AlignmentMode) |

Defines the specific mode, with respect to which the objects to be aligned. This is an optional parameter. The default alignment mode is `Object`.

The accepted values of the argument "alignment mode" are as follows.

Object Aligns the objects based on the first object in the selected list.

Selector Aligns the objects based on the selection boundary.

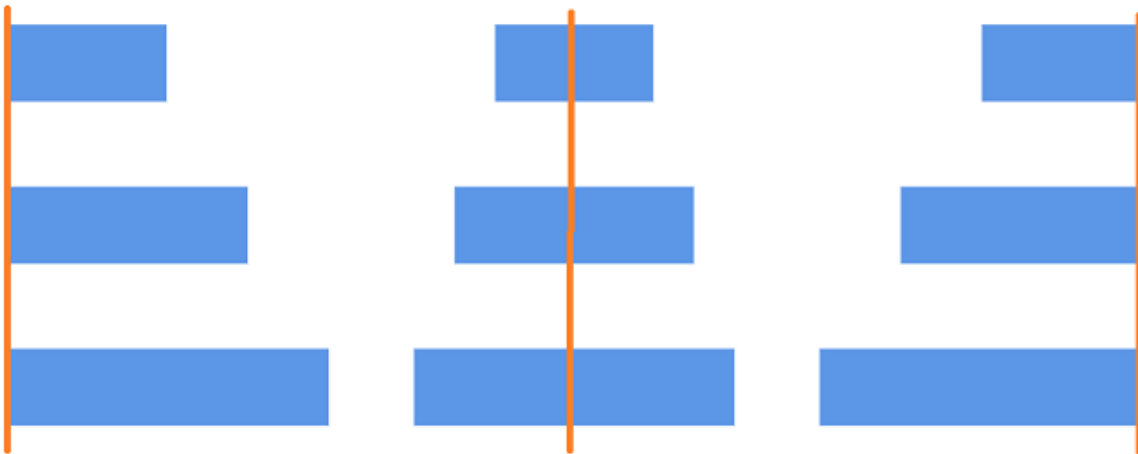
|

The following code example illustrates how to align all the selected objects at the left side of the selection boundary.

```
`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding, ej.diagrams.NodeModel);
//Initializes the node
var node = {
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
```

```
};  
var node2 = {  
  id: 'node2',  
  width: 100,  
  height: 60,  
  offsetX: 100,  
  offsetY: 170,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
};  
var node3 = {  
  id: 'node3',  
  width: 140,  
  height: 60,  
  offsetX: 100,  
  offsetY: 240,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
};  
//Initializes the Diagram Component  
var diagram = new ej.diagrams.Diagram({  
  width: '100%',  
  height: '350px',  
  nodes: [node, node2, node3]  
}, '#element');  
var selArray = [];  
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);
```

```
//Selects the nodes
diagram.select(selArray);
//Sets direction as left
diagram.align('Left', diagram.selectedItems.nodes, 'Selector');
```



Distribute

The [Distribute](#) commands enable to place the selected objects on the page at equal intervals from each other. The selected objects are equally spaced within the selection boundary.

The factor to distribute the shapes [DistributeOptions](#) are listed as follows:

- RightToLeft: Distributes the objects based on the distance between the right and left sides of the adjacent objects.
- Left: Distributes the objects based on the distance between the left sides of the adjacent objects.
- Right: Distributes the objects based on the distance between the right sides of the adjacent objects.
- Center: Distributes the objects based on the distance between the center of the adjacent objects.
- BottomToTop: Distributes the objects based on the distance between the bottom and top sides of the adjacent objects.
- Top: Distributes the objects based on the distance between the top sides of the adjacent objects.
- Bottom: Distributes the objects based on the distance between the bottom sides of the adjacent objects.
- Middle: Distributes the objects based on the distance between the vertical center of the adjacent objects.

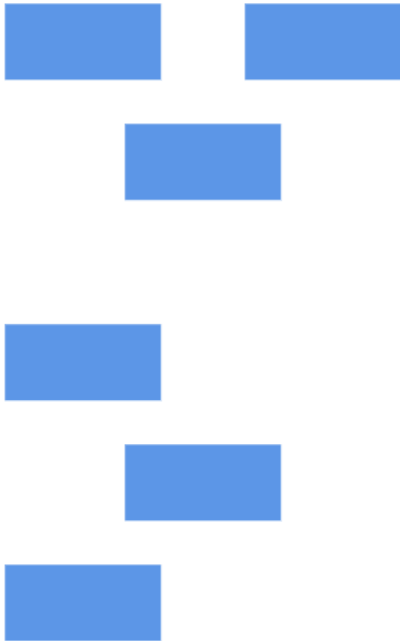
The following code example illustrates how to execute the space commands.

```
`javascript
```

```
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding, ej.diagrams.NodeModel);  
//Initializes the node  
var node = {  
  id: 'node1',  
  width: 90,  
  height: 60,  
  offsetX: 100,  
  offsetY: 100,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
};  
var node2 = {  
  id: 'node2',  
  width: 90,  
  height: 60,  
  offsetX: 240,  
  offsetY: 100,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
};  
var node3 = {  
  id: 'node3',  
  width: 90,  
  height: 60,  
  offsetX: 170,  
  offsetY: 150,  
  style: {
```



```
fill: '#6BA5D7',
strokeColor: 'white',
strokeWidth: 1
},
};
//Initializes the diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '350px',
nodes: [node, node2, node3]
},'#element');
let selArray: (NodeModel | ConnectorModel)[] = [];
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);
//Selects the nodes
diagram.select(selArray);
//Distributes space between the nodes
diagram.distribute('RightToLeft', diagram.selectedItems.nodes);
`
```



Sizing

Sizing [sameSize](#) commands enable to equally size the selected nodes with respect to the first selected object.

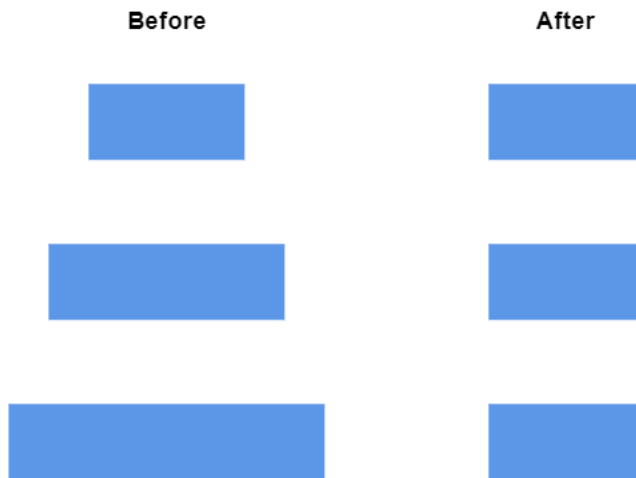
[SizingOptions](#) are as follows:

- Width: Scales the width of the selected objects.
- Height: Scales the height of the selected objects.
- Size: Scales the selected objects both vertically and horizontally.

The following code example illustrates how to execute the size commands.

```
`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.DataBinding, ej.diagrams.NodeModel);
//Initializes the node
var node = {
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
};
var node2 = {
  id: 'node2',
  width: 100,
  height: 60,
  offsetX: 100,
  offsetY: 170,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
```

```
};  
var node3 = {  
  id: 'node3',  
  width: 130,  
  height: 60,  
  offsetX: 100,  
  offsetY: 230,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
};  
//Initializes the diagram component  
var diagram = new ej.diagrams.Diagram({  
  width: '100%',  
  height: '350px',  
  nodes: [node, node2, node3]  
}, '#element');  
let selArray: (NodeModel)[] = [];  
selArray.push(diagram.nodes[0], diagram.nodes[1], diagram.nodes[2]);  
//Selects the nodes  
diagram.select(selArray);  
//Resizes the selected nodes with the same width  
diagram.sameSize('Width', diagram.selectedItems.nodes);  
`
```



Clipboard

Clipboard commands are used to cut, copy, or paste the selected elements. Refer to the following link which shows how to use clipboard methods in the diagram.

- Cuts the selected elements from the diagram to the diagram's clipboard, [cut](#).
- Copies the selected elements from the diagram to the diagram's clipboard, [copy](#).
- Pastes the diagram's clipboard data (nodes/connectors) into the diagram, [paste](#).

The following code illustrates how to execute the clipboard commands.

INDEX.JS

```
var nodes = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 240,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}
];
```

```
var connector = {
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
    targetDecorator: {
      style: {
        fill : '#6BA5D7',
        strokeColor : '#6BA5D7'
      }
    }
  }
};

var diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  nodes: [nodes],
  connectors: [connector]
}, '#element');
diagram.select([diagram.nodes[0], diagram.nodes[1], diagram.connectors[0]]);
diagram.copy();
diagram.paste(diagram.copy());
```

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>
```

Grouping

Grouping commands are used to group/ungroup the selected elements on the diagram. Refer to the following link which shows how to use grouping commands in the diagram.

[Group](#) the selected nodes and connectors in the diagram.

[Ungroup](#) the selected nodes and connectors in the diagram.

The following code illustrates how to execute the grouping commands.

INDEX.JS

```
var nodes = [{
  id: 'node1',
  width: 100,
  height: 70,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 100,
  height: 70,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node3',
  width: 100,
  height: 70,
  offsetX: 200,
  offsetY: 200,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
```

```

        strokeWidth: 1
    },
    },
    {
        id: 'group',
        children: ['node1', 'node2', 'connector1']
    },
    {
        id: 'group2',
        children: ['node3', 'group']
    }
];
var connector = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    style: {
        strokeColor : '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth : 2,
        targetDecorator: {
            style: {
                fill : '#6BA5D7',
                strokeColor : '#6BA5D7'
            }
        }
    }
};
var diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    nodes: nodes,
    connectors: [connector],
}, '#element');
diagram.selectAll();
//Groups the selected elements.
diagram.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>
```

Z-Order command

Z-Order commands enable you to visually arrange the selected objects such as nodes and connectors on the page.

bringToFront command

The [bringToFront](#) command visually brings the selected element to front over all the other overlapped elements. The following code illustrates how to execute the `bringToFront` command.

INDEX.JS

```
var nodes = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
},
{
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 240,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    }
}
```



```

    },
    },
    {
        id: 'node3',
        width: 90,
        height: 60,
        offsetX: 160,
        offsetY: 90,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
    },
}
];
var diagram = new ej.diagrams.Diagram({
    width: '650px',
    height: '350px',
    nodes: [nodes]
}, '#element');
let selArray = [];
selArray.push(diagram.nodes[2]);
diagram.select(selArray);
//Brings to front
diagram.bringToFront();

```

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>
```

sendToBack command

The [sendToBack](#) command visually moves the selected element behind all the other overlapped elements. The following code illustrates how to execute the `sendToBack` command.

INDEX.JS

```
var nodes = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 240,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node3',
  width: 90,
  height: 60,
  offsetX: 160,
  offsetY: 90,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
];
var diagram = new ej.diagrams.Diagram({
```

```

        width: '650px',
        height: '350px',
        nodes: [nodes]
    }, '#element');
    let selArray = [];
    selArray.push(diagram.nodes[2]);
    diagram.select(selArray);
    //Sends back the selected elements
    diagram.sendToBack();

```

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>

```

[moveForward](#) command

The [moveForward](#) command visually moves the selected element over the nearest overlapping element. The following code illustrates how to execute the `moveForward` command.

INDEX.JS

```
var nodes = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 180,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
];
var diagram = new ej.diagrams.Diagram({
  width: '650px',
  height: '350px',
  nodes: [nodes]
}, '#element');
let selArray = [];
selArray.push(diagram.nodes[1]);
diagram.select(selArray);
//Moves forward
diagram.moveForward();
```

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>

```

sendBackward command

The [sendBackward](#) command visually moves the selected element behind the underlying element. The following code illustrates how to execute the `sendBackward` command.

INDEX.JS

```

var nodes = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
},
{
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 180,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
},
];
var diagram = new ej.diagrams.Diagram({

```

```

        width: '650px',
        height: '350px',
        nodes: [nodes]
    }, '#element');
    let selArray = [];
    selArray.push(diagram.nodes[1]);
    diagram.select(selArray);
    //Sends backward
    diagram.sendBackward();

```

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>

```

Zoom

The [zoom](#) command is used to zoom-in and zoom-out the diagram view.

The following code illustrates how to zoom-in/zoom out the diagram.

```
`javascript
import {
  Diagram
} from '@syncfusion/ej2-diagrams';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
});
diagram.appendTo('#element');
// Sets the zoomFactor
//Defines the focusPoint to zoom the diagram with respect to any point
//When you do not set focus point, zooming is performed with reference to the center of current
diagram view.
diagram.zoom(1.2, {
  x: 100,
  y: 100
});
`
```

Nudge command

The [nudge](#) commands move the selected elements towards up, down, left, or right by 1 pixel.

[NudgeDirection](#) nudge command moves the selected elements towards the specified direction by 1 pixel, by default.

The accepted values of the argument "direction" are as follows:

- Up: Moves the selected elements towards up by the specified delta value.
- Down: Moves the selected elements towards down by the specified delta value.
- Left: Moves the selected elements towards left by the specified delta value.
- Right: Moves the selected elements towards right by the specified delta value.

The following code illustrates how to execute nudge command.

```
`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram);
//Initializes the Diagram component
var diagram = new ej.diagrams.Diagram({
```

```
width: '100%',  
height: '350px',  
, '#element');  
//Nudges to right  
diagram.nudge('Right');  
,
```

Nudge by using arrow keys

The corresponding arrow keys are used to move the selected elements towards up, down, left, or right direction by 1 pixel.



Nudge commands are particularly useful for accurate placement of elements.

BringIntoView

The [bringIntoView](#) command brings the specified rectangular region into the viewport of the diagram.

The following code illustrates how to execute the [bringIntoView](#) command.

```
`javascript  
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram);  
//Initializes the diagram component  
var diagram = new ej.diagrams.Diagram({  
width: '100%',  
height: '350px',  
, '#element');  
//Brings the specified rectangular region of the diagram content to the viewport of the page.  
let bound: Rect = new Rect(200, 400, 500, 400);  
diagram.bringIntoView(bound);  
,
```

BringToCenter

The [bringToCenter](#) command brings the specified rectangular region of the diagram content to the center of the viewport.

The following code illustrates how to execute the [bringToCenter](#) command.

```
`javascript  
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram, ej.diagrams.React);  
//Initializes the Diagram component  
var diagram = new ej.diagrams.Diagram({
```



```
width: '100%',  
height: '350px',  
, '#element');  
//Brings the specified rectangular region of the Diagram content to the center of the viewport.  
var bound = ej.diagram.Rect(200, 400, 500, 400);  
diagram.bringToCenter(bound);  
`
```

FitToPage command

The [fitToPage](#) command helps to fit the diagram content into the view with respect to either width, height, or at the whole.

The [mode](#) parameter defines whether the diagram has to be horizontally/vertically fit into the viewport with respect to width, height, or entire bounds of the diagram.

The [region](#) parameter defines the region that has to be drawn as an image.

The [margin](#) parameter defines the region/bounds of the diagram content that is to be fit into the view.

The [canZoomIn](#) parameter enables/disables zooming to fit the smaller content into a larger viewport.

The [customBounds](#) parameter the custom region that has to be fit into the viewport.

The following code illustrates how to execute **FitToPage** command.

```
`javascript  
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram);  
//Initializes the Diagram component  
var diagram = new ej.diagrams.Diagram({  
width: '100%',  
height: '350px',  
, '#element');  
//fit the diagram to the page with respect to mode and region  
diagram.fitToPage({  
mode: 'Page',  
region: 'Content',  
margin: {  
bottom: 50  
},  
canZoomIn: false  
});  
`
```

Command manager

Diagram provides support to map/bind command execution with desired combination of key gestures. Diagram provides some built-in commands.

[CommandManager](#) provides support to define custom commands. The custom commands are executed, when the specified key gesture is recognized.

Custom command

To define a custom command, specify the following properties:

- [execute](#): A method to be executed.
- [canExecute](#): A method to define whether the command can be executed at the moment.
- [gesture](#): A combination of [keys](#) and [KeyModifiers](#).
- [parameter](#): Defines any additional parameters that are required at runtime.
- [name](#): Defines the name of the command.

To explore the properties of custom commands, refer to [Commands](#).

The following code example illustrates how to define a custom command.

```
`javascript
import {
  Diagram,
  Keys,
  KeyModifiers
} from '@syncfusion/ej2-diagrams';
//Initializes the Diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '350px',
  commandManager: {
    commands: [{
      name: 'customCopy',
      parameter: 'node',
      //Method to define whether the command can be executed at the current moment
      canExecute: function() {
        //Defines that the clone command can be executed, if and only if the selection list is not empty.
        if (diagram.selectedItems.nodes.length > 0 || diagram.selectedItems.connectors.length > 0) {
          return true;
        } else {
          return false;
        }
      }
    }]
  }
});
```

```
}  
},  
//Command handler  
execute: function() {  
  //Logic to clone the selected element  
  diagram.copy();  
  diagram.paste();  
  diagram.dataBind();  
},  
//Defines that the clone command has to be executed on the recognition of key press.  
gesture: {  
  key: Keys.G,  
  keyModifiers: KeyModifiers.Shift | KeyModifiers.Alt  
}  
}]  
},  
},'#element');  
`
```

Modify the existing command

When any one of the default commands is not desired, they can be disabled. To change the functionality of a specific command, the command can be completely modified.

The following code example illustrates how to disable a command and how to modify the built-in commands.

```
`javascript  
import {  
  Diagram,  
  Keys,  
  KeyModifiers  
} from '@syncfusion/ej2-diagrams';  
//Initializes the diagram component  
var diagram = new ej.diagrams.Diagram({  
  width: '100%',  
  height: '350px',  
  //Disables the nudging commands
```

```
commandManager: {
  commands: {
    //Assigns null value to an existing command and disables its execution
    "nudgeUp": null,
    "nudgeDown": null,
    "nudgeRight": null,
    //Modifies the existing command - nudgeLeft
    "nudgeLeft": {
      canExecute: function(args){
        if (args.model.selectedItems.length) {
          return true;
        }
      },
      //Command handler
      execute: function(args) {
        diagram.nudge("left");
      },
      gesture: {
        key: Keys.Left
      }
    },
    '#element'
  },
  '

```

[See Also](#)

- [How to create the custom context menu items](#)

Undo redo in EJ2 JavaScript Diagram control

Diagram tracks the history of actions that are performed after initializing the diagram and provides support to reverse and restore those changes.

Undo and redo

Diagram provides built-in support to track the changes that are made through interaction and through public APIs. The changes can be reverted or restored either through shortcut keys or through commands.

Note: If you want to use Undo-Redo in diagram, you need to inject UndoRedo in the diagram.

Undo/redo through shortcut keys

Undo/redo commands can be executed through shortcut keys. Shortcut key for undo is Ctrl+z and shortcut key for redo is Ctrl+y.

Undo/redo through public APIs

The client-side methods [undo](#) and [redo](#) help you to revert/restore the changes. The following code example illustrates how to undo/redo the changes through script.

```
`javascript
// initialize diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
// Add node
nodes: [];
},'#element');
// Reverts the last action performed
diagram.undo();
// Restores the last undone action
diagram.redo();
`
```

When a change in the diagram is reverted or restored (undo/redo), the historyChange event gets triggered.

Group multiple changes

History list allows to revert or restore multiple changes through a single undo/redo command. For example, revert/restore the fill color change of multiple elements at a time.

The client-side method [startGroupAction](#) is used to notify the diagram to start grouping the changes. The client-side method [endGroupAction](#) is used to notify to stop grouping the changes. The following code illustrates how to undo/redo fillColor change of multiple elements at a time.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.UndoRedo);
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
});
diagram.appendTo('#element');
//Start to group the changes
diagram.startGroupAction();
//Makes the changes
var color = ['black', 'red', 'green', 'yellow']
for (var i = 0; i < color.length; i++) {
    // Updates the fillColor for all the child elements.
    diagram.nodes[0].style.fill=color[i];
    diagram.dataBind();
}
//Ends grouping the changes
diagram.endGroupAction();
```

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>
```

Track custom changes

Diagram provides options to track the changes that are made to custom properties. For example, in case of an employee relationship diagram, track the changes in the employee information. The `historyList` of the diagram enables you to track such changes. The following example illustrates how to track such custom property changes.

Before changing the employee information, save the existing information to `historyList` by using the client-side method `push` of `historyList`.

The `historyList` `canLog` method can be used which takes a history entry as argument and returns whether the specific entry can be added or not.

The following code example illustrates how to save the existing property values.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
}, '#element');
//Creates a custom entry
var entry = {
undoObject: diagram.nodes[0];
};
// adds that to history list
diagram.historyList.push(entry);
`
```

canLog

`canLog` in the history list, which takes a history entry as argument and returns whether the specific entry can be added or not.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.UndoRedo);
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');
//Creates a custom entry and adds that to history manager
var entry = { undoObject: diagram.nodes[0] };
    diagram.historyList.push(entry);
    diagram.dataBind();
// Reverts the last action performed
diagram.undo();
// Restores the last undone action
diagram.redo();

```

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>

```

Track undo/redo actions

The historyList undoStack property is used to get the collection of undo actions which should be performed in the diagram. The undoStack/redoStack is the read-only property.


```
`javascript
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
nodes: nodes,
},'#element');
//get the collection of undoStack objects
let undoStack = diagram.historyList.undoStack;
//get the collection of redoStack objects
let redoStack = diagram.historyList.redoStack;
`
```

History change event

The [historyChange](#) event triggers, whenever the interaction of the node and connector is take place.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
nodes: nodes,
},'#element');
// history change event
diagram.historyChange = (arg) => {
//causes of history change
let cause: string = arg.cause;
}
`
```

Stack Limit

The [stackLimit](#) property of history manager is used to limits the number of actions to be stored on the history manager.

INDEX.JS

```
ej.diagrams.Diagram.Inject(ej.diagrams.UndoRedo);
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,
    historyManager: { stackLimit: 3 },
    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>
</body>
</html>

```

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

Retain Selection

You can retain a selection at undo/redo operation by using the client-side API Method called **updateSelection**. Using this method, we can select a diagram objects.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
updateSelection: (object: NodeModel, diagram: Diagram) => {
let selArr = [];
selArr.push(object)
diagram.select(selArr);
},
nodes: nodes,
},'#element');
`
```

Virtualization in EJ2 JavaScript Diagram control

Virtualization in Diagram

Virtualization is the process of loading the diagramming objects available in the visible area of the Diagram control, that is, only the diagramming objects that lie within the ViewPort of the Scroll Viewer are loaded (remaining objects are loaded only when they come into view).

This feature gives an optimized performance while loading and dragging items to the Diagram that consists of many Nodes and Connectors.

The following code illustrates how to enable Virtualization mode in the diagram.

```
`ts
//Initialize diagram
var diagram = new ej.diagrams.Diagram({
width: '800px', height: '500px',
//Enable virtualization in diagram
constraints: DiagramConstraints.Default | DiagramConstraints.Virtualization,
},'#diagram');
`
```

Serialization in EJ2 JavaScript Diagram control

Serialization is the process of saving and loading for state persistence of the diagram.

Save

The diagram is serialized as string while saving. The client-side method, [saveDiagram](#) helps to serialize the diagram as a string. The following code illustrates how to save the diagram.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
var saveData;
//returns serialized string of the Diagram
saveData = diagram.saveDiagram();
```

This string can be converted to JSON data and stored for future use. The following snippet illustrates how to save the serialized string into local storage.

```
`javascript
//Saves the string in to local storage
localStorage.setItem('fileName', saveData);
saveData = localStorage.getItem('fileName');
```

Diagram can also be saved as raster or vector image files. For more information about saving the diagram as images, refer to [Print and Export](#).

Load

Diagram is loaded from the serialized string data by client-side method, [loadDiagram](#).

The following code illustrates how to load the diagram from serialized string data.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram: Object[] = diagramElement.ej2_instances[0];
//Loads the diagram from saved json data
diagram.loadDiagram(saveData);
```

Note: Before loading a new diagram, existing diagram is cleared.

Prevent Default Values

The [preventDefaults](#) property of `serializationSettings` is used to simplifying the saved JSON object without adding the default properties that are presented in the diagram.

The following code illustrates how to simplify the JSON object.

```
`javascript
var diagram = new ej.diagrams.Diagram({
  serializationSettings: { preventDefaults: true },
});
`
```

Export in EJ2 JavaScript Diagram control

Diagram provides support to export its content as image/svg files. The client-side method [exportDiagram](#) helps to export the diagram. The following code illustrates how to export the diagram as image.

Note: To use Print and Export, you need to inject `PrintAndExport` in the diagram.

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

```
`javascript
var diagram = new ej.diagrams.Diagram({
  width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
diagram.exportDiagram(options);
`
```

Exporting options

Diagram provides support to export the desired region of the diagram to desired formats.

File Name

[FileName](#) is the name of the file to be downloaded. By default, the file name is set to **Diagram**.

Format

[Format](#) is to specify the type/format of the exported file. By default, the diagram is exported as .jpg format. You can export diagram to the following formats:

- JPG
- PNG
- BMP
- SVG

```
`javascript
var diagram = new ej.diagrams.Diagram({
  width: 1500, height: 1500
}, '#element');
```

```
var options = {};  
options.mode = 'Download';  
options.format = 'SVG';  
diagram.exportDiagram(options);  
`
```

Margin

[Margin](#) specifies the amount of space that has to be left around the diagram.

<!-- markdownlint-disable MD033 -->

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
width: 1500, height: 1500  
, '#element');  
var options = {};  
options.mode = 'Download';  
options.margin = { left: 10, right: 10, top: 10, bottom: 10};  
options.fileName = 'format';  
options.format = 'SVG';  
diagram.exportDiagram(options);  
`
```

Mode

[Mode](#) specifies whether the diagram will be exported as files or get base64 data(ImageURL/SVG). The export options are as follows:

- Download: Exports and downloads the diagram as image/SVG.
- Data: return a base64 string.

The following code example illustrates how to export the diagram as raw data.

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
width: 1500, height: 1500  
, '#element');  
var options = {};  
options.mode = 'Data';  
options.margin = { left: 10, right: 10, top: 10, bottom: 10};  
options.fileName = 'format';
```

```
options.format = 'SVG';  
var base64data = diagram.exportDiagram(options);  
`
```

Region

You can export any particular [region](#) of the diagram and it is categorized into three types as follows.

- PageSettings
- Content
- CustomBounds

PageSettings

Diagram is exported based on the given PageSettings width and height. The Properties available in page settings are as follows.

- width
- height
- margin
- orientation
- boundaryConstraints
- background
- multiplePage
- showPageBreaks
- fitOptions

boundaryConstraints

Defines the editable region of the diagram.

- Infinity - Allow the interactions to take place at infinite height and width.
- Diagram - Allow the interactions to take place around the diagram's height and width.
- Page - Allow the interactions to take place around the page's height and width.

multiplePage

While setting multiple pages as false, the diagram is exported as a single image and while setting multiple pages as true, the diagram is exported as a separate image based on width and height.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`javascript  
var diagram = new ej.diagrams.Diagram({  
width: 1500, height: 1500  
}, '#element');  
var options = {};  
options.mode = 'Download';  
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
```

```
options.fileName = 'format';
options.format = 'SVG';
options.region = 'PageSettings';
diagram.exportDiagram(options);
`
```

Content

The diagram content alone will be exported as an image.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
options.region = 'Content';
diagram.exportDiagram(options);
`
```

Custom bounds

Diagram provides support to export any specific region of the diagram by using [bounds](#).

The following code example illustrates how to export the region occupied by the diagram elements.

```
`ts
var diagram = new ej.diagrams.Diagram({
width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'CustomBounds';
options.bounds.x = 10;
```



```
options.bounds.y = 10;
options.bounds.height = 100;
options.bounds.width = 100;
diagram.exportDiagram(options);
`
```

Export diagram with stretch option

Diagram provides support to export the diagram as image for [stretch](#) option. The exported images will be clearer but larger in file size.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'Content';
options.stretch = 'Stretch';
diagram.exportDiagram(options);
`
```

Print

The client-side method [print](#) helps to print the diagram as image.

Name	Type	Description
region	enum	Sets the region of the diagram to be printed.
bounds	object	Prints any custom region of diagram.
stretch	enum	Resizes the diagram content to fill its allocated space and printed.
multiplePage	boolean	Prints the diagram into multiple pages.
pageWidth	number	Sets the page width of the diagram while printing the diagram into multiple pages.
pageHeight	number	Sets the page height of the diagram while printing the diagram into multiple pages.
pageOrientation	enum	Sets the orientation of the page.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`ts
var diagram = new ej.diagrams.Diagram({
width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
options.region = 'PageSettings';
options.multiplePage = true;
options.pageHeight = 300;
options.pageWidth = 300;
diagram.print(options);
`
```

Limitations

We have a limitation in exporting the image with HTML and Native node. So, Syncfusion Essential PDF library is used, which supports HTML Content to Image conversion by using the advanced Qt WebKit rendering engine. You can refer to the following KB link for more details.

[<https://www.syncfusion.com/kb/13298/how-to-print-or-export-the-html-and-native-node-into-image-format>]

Tool tip in EJ2 JavaScript Diagram control

<!-- markdownlint-disable MD010 -->

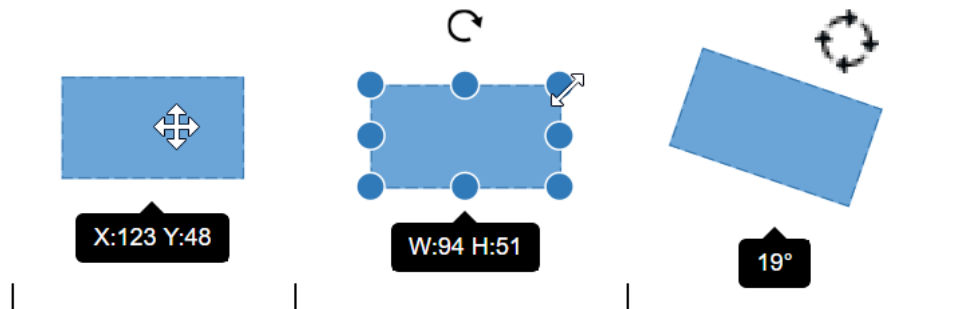
In Graphical User Interface (GUI), the tooltip is a message that is displayed when mouse hovers over an element. The diagram provides tooltip support while dragging, resizing, rotating a node, and when the mouse hovers any diagram element.

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.TS

```
import {
  Diagram,
  NodeModel,
  DiagramConstraints,
  NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
  NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
  constraints: DiagramConstraints.Default | 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: NodeConstraints.Default | NodeConstraints.Tooltip,
  }],
});
```

```
//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.TS

```
import {
    Diagram,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '350px',
    constraints: DiagramConstraints.Default | 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: NodeConstraints.Default | 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'
        },
    }],
}, '#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.

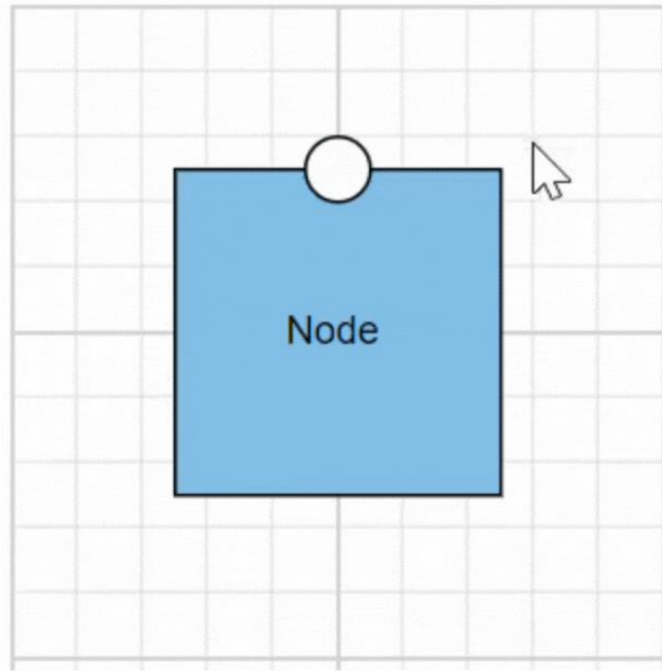
Tooltips for Ports can be enabled or disabled using the [PortConstraints](#) Tooltip property.

```
`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.TS

```
import {
  Diagram,
  NodeModel,
  PortVisibility,
  PortConstraints
} 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,
    tooltip: {
      content: 'Port Tooltip',
    },
  }],
}
```



```

        constraints: PortConstraints.Default | PortConstraints.Tooltip
    }}
};
// 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-
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.TS

```
import {
    Diagram,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '350px',
    constraints: DiagramConstraints.Default | 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: NodeConstraints.Default | 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'
        }
    }
}]);
```

```

}, '#element');
function getContent(): HTMLElement {
    let tooltipContent: HTMLElement = document.createElement('div');
    tooltipContent.innerHTML = '<div style="background-color: #f4f4f4;
color: black; border-width:1px;border-style: solid;border-color: #d3d3d3;
border-radius: 8px;white-space: nowrap;"> <span style="margin: 10px;">
Tooltip !!! </span> </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.TS

```
import {
    Diagram,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '350px',
    constraints: DiagramConstraints.Default | 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: (NodeConstraints.Default &
~NodeConstraints.InheritTooltip) | NodeConstraints.Tooltip,
        //Defines mouse over tooltip for a node
        tooltip: {
            content: 'Node1',
            //Sets the alignment properties
            position: 'BottomRight',
            //Sets to show tooltip around the element
            relativeMode: 'Object',
        },
    }],
});
```

```
    }],
  }, '#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.TS

```
import {
```

```

    Diagram,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '350px',
    constraints: DiagramConstraints.Default | 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: NodeConstraints.Default | NodeConstraints.Tooltip,
        //Defines mouse over tooltip for a node
        tooltip: {
            content: 'Node1',
            //Sets to show tooltip at mouse position
            relativeMode: 'Mouse',
        },
    }],
}, '#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.TS

```

import {
    Diagram,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
//Initializes the Diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '350px',
    constraints: DiagramConstraints.Default | 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: NodeConstraints.Default | 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: 'ZoomIn',
//Duration of the animation that is completed per
animation cycle.
      duration: 1000,
//Indicating the waiting time before animation begins.
      delay: 0
    },
//Animation settings to be applied on the tooltip, when it
is closed.
    close: {
      effect: 'ZoomOut',
      duration: 500,
      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 EJ2 JavaScript 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.TS

```
import { Diagram, NodeModel, BasicShapeModel, MoveTool, MouseEventArgs,
IElement, UserHandleModel, ToolBase, SelectorConstraints, Actions, randomId,
cloneObject } from '@syncfusion/ej2-diagrams';
let shape: BasicShapeModel = { type: 'Basic', shape: 'Rectangle' };
let node1: NodeModel = { id: 'node', offsetX: 100, offsetY: 100, shape:
shape };
let handles: UserHandleModel[] = [{
  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 }
}];
let diagram: Diagram = new Diagram({
  width: '100%', height: '600px', nodes: [node1],
  selectedItems: { constraints: SelectorConstraints.All, userHandles:
handles },
  getNodeDefaults: (node: NodeModel): NodeModel => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = '#6BA5D7';
    return node;
  },
  getCustomTool: getTool
});
diagram.appendTo('#element');
function getTool(action: string): ToolBase {
  let tool: ToolBase;
  if (action === 'clone') {
    tool = new CloneTool(diagram.commandHandler);
  }
}
```

```

    return tool;
}
//Defines the clone tool used to copy Node/Connector
class CloneTool extends MoveTool {
    public mouseDown(args: MouseEventArgs): void {
        let newObject: any;
        if (diagram.selectedItems.nodes.length > 0) {
            newObject = cloneObject(diagram.selectedItems.nodes[0]) as
NodeModel;
        } else {
            newObject = cloneObject(diagram.selectedItems.connectors[0]) as
ConnectorModel;
        }
        newObject.id += randomId();
        diagram.paste([newObject]);
        args.source = diagram.nodes[diagram.nodes.length - 1] as IElement;
        args.sourceWrapper = args.source.wrapper;
        super.mouseDown(args);
        this.inAction = true;
    }
}

```

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>

```

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.TS

```

import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
    offsetX: 250,
    offsetY: 250,
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
},
// A fixed user handle is created and stored in fixed user handle collection of Node.
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' }],
};
let diagram: Diagram = new 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

- 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.TS

```
import { Diagram, NodeModel, ConnectorModel } from '@syncfusion/ej2-
diagrams';
let node: NodeModel = {
  offsetX: 150,
  offsetY: 150,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
width: 20, height: 20,
padding: { left: 3, right: 3, top: 3, bottom: 3 }, iconStrokeColor: 'white', fill: 'black'
, 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' }]
};
let connector: ConnectorModel = {
  id: 'connector1',
  type: 'Orthogonal',
  sourcePoint: { x: 300, y: 100 },
  targetPoint: { x: 400, y: 200 },
  fixedUserHandles: [{ offset: 0.5, width: 20, alignment: 'Before',
padding: { left: 3, right: 3, top: 3, bottom: 3 }, iconStrokeColor: 'white', fill: 'black'
, 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' }]
};
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  nodes: [node],
```

```

        connectors: [connector]
    });
    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 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.


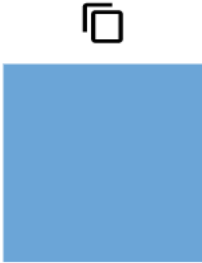


Margin for the node fixed user handle

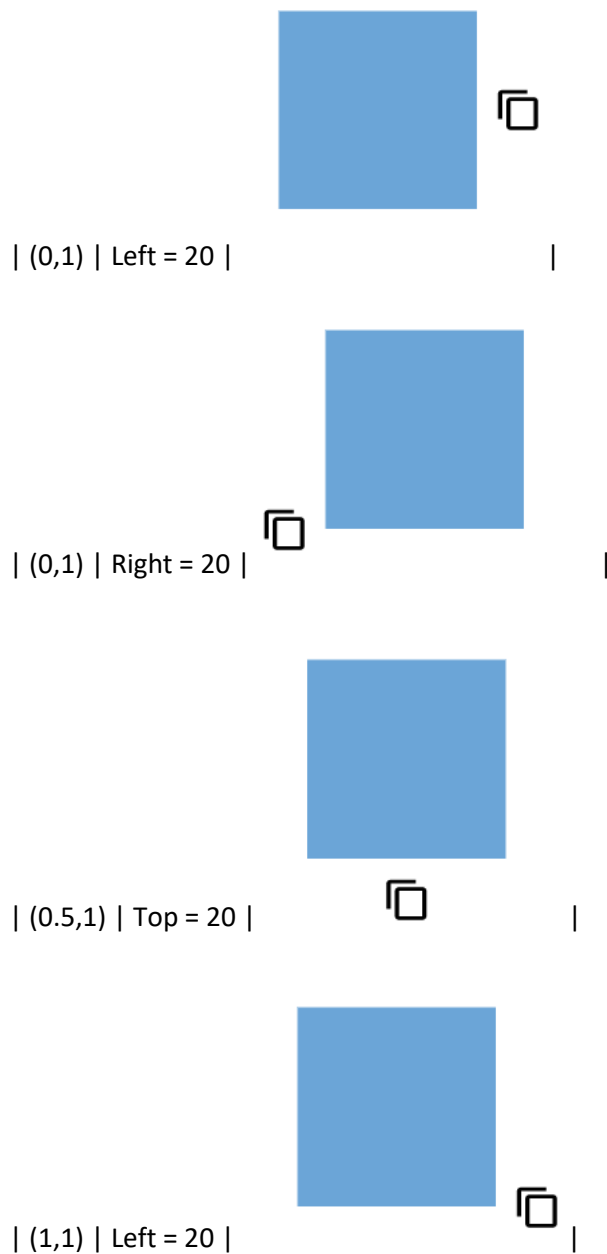
Margin is an absolute value used to add some blank space in any one of its four sides. The fixed user handle can be displaced with the margin property.

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	
(1,0)	Left = 20	
(0,0.5)	Right = 20	



The following code explains how to customize the node fixed user handle.

INDEX.TS

```
import { Diagram, NodeModel } from '@syncfusion/ej2-diagrams';
let node: NodeModel = {
  offsetX: 200,
  offsetY: 200,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  }
}
```

```

    },// A fixed user handle is created and stored in fixed user handle
    collection of Node.
    fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
width: 20,height: 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' }]
    };
let diagram: Diagram = new 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 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.


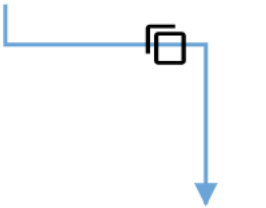
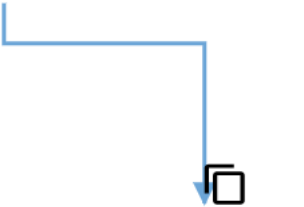
Offset for the connector fixed user handle

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.


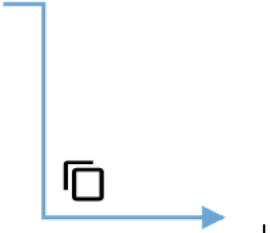
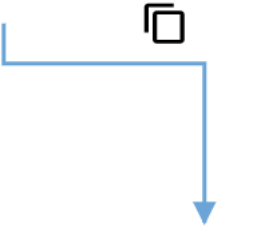
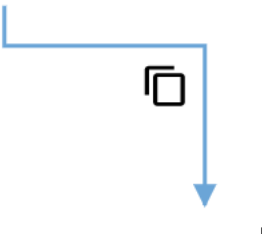
The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Alignment	Output
-----	-----	-----
0 Before		
0.5 Center		
1 After		

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
-----	-----	-----
x=10 Before		
x=10 After		
y=10 Before		
y=10 After		

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.TS

```
import { Diagram, ConnectorModel } from '@syncfusion/ej2-diagrams';
let connector: ConnectorModel = {
  id: 'connector1',
  type: 'Orthogonal',
  sourcePoint: { x: 300, y: 100 },
  targetPoint: { x: 400, y: 200 },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Connector.
  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' }]
};
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '600px',
  connectors: [connector]
});
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>

```

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.TS

```

import {
    Diagram,
    NodeModel,
    DiagramConstraints,
    UserHandleModel,
    SelectorConstraints,
    NodeConstraints
} from '@syncfusion/ej2-diagrams';
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
let node: NodeModel = {
    // 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: NodeConstraints.Default | NodeConstraints.Tooltip
};
let handle: UserHandleModel[] = [{
    name: 'handle1',
    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, backgroundColor: 'black', offset: 0, side: 'Right',
    pathColor: 'white',
    tooltip: { content: 'handle1', position: 'BottomRight', relativeMode:
'Object' },
}]
// initialize Diagram component
let diagram: Diagram = new Diagram({
    width: '100%',
    height: '600px',
    // Add node

```

```

    nodes: [node],
    constraints: DiagramConstraints.Default | DiagramConstraints.Tooltip,
    userHandleTemplate: '#userHandleTemplate',
    selectedItems: { constraints: SelectorConstraints.All, userHandles:
handle },
});
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>

```

Style in EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import {
  Diagram} from '@syncfusion/ej2-diagrams';
let diagram: Diagram = new 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.TS

```
import { Diagram } from '@syncfusion/ej2-diagrams';
let diagram: Diagram = new Diagram({
    width: '100%', height: '600px', rulerSettings: {
        showRulers: true, horizontalRuler: { interval: 8, segmentWidth: 100,
        thickness: 25, 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.

Layers in EJ2 JavaScript Diagram control

Layer is used to organize related shapes on a diagram control. A layer is a named category of shapes. By assigning shapes to different layers, you can selectively view, remove, and lock different categories of shapes.

In diagram, [Layers](#) provide a way to change the properties of all shapes that have been assigned to that layer. The following properties can be set.

- Visible
- Lock
- Objects
- AddInfo

Visible

The layer's [visible](#) property is used to control the visibility of the elements assigned to the layer.

```
`javascript
```

```
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram,  
ej.diagrams.ConnectorModel,ej.diagrams.NodeModel,ej.diagrams.LayerModel);
```

```
// A node is created and stored in nodes array.
```

```
var nodes = [{  
  id: 'node1',  
  width: 100,  
  height: 100,  
  offsetX: 100,  
  offsetY: 100,  
  annotations: [{  
    content: 'Default Shape'  
  }]  
},  
{  
  id: 'node2',  
  width: 100,  
  height: 100,
```

```
offsetX: 300,
offsetY: 100,
shape: {
  type: 'Path',
  data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
  content: 'Path Element'
}]
}
];
var connectors = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];
// initialize diagram component
var diagram = new ej.diagrams.Diagram({
  width: '100%',
  height: '600px',
  connectors: connectors,
  nodes: nodes,
  layers: [{
```

```
id: 'layer1',  
visible: true,  
objects: ['node1']  
}  
],  
, '#element');  
,
```

Lock

The layer's [lock](#) property is used to prevent or allow changes to the elements dimension and position.

```
`javascript  
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram,  
ej.diagrams.ConnectorModel, ej.diagrams.NodeModel, ej.diagrams.LayerModel);  
  
// A node is created and stored in nodes array.  
var nodes = [  
  {  
    id: 'node1',  
    width: 100,  
    height: 100,  
    offsetX: 100,  
    offsetY: 100,  
    annotations: [{  
      content: 'Default Shape'  
    }]  
  },  
  {  
    id: 'node2',  
    width: 100,  
    height: 100,  
    offsetX: 300,  
    offsetY: 100,  
    shape: {  
      type: 'Path',
```



```
data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
content: 'Path Element'
}]
}
];
var connectors = [{
id: 'connector1',
type: 'Straight',
sourcePoint: {
x: 100,
y: 300
},
targetPoint: {
x: 200,
y: 400
},
}];
// initialize diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
connectors: connectors,
nodes: nodes,
layers: [
{
id: 'layer1',
visible: true,
objects: ['node1'],
```

```
lock: true
},
{
id: 'layer2',
visible: true,
objects: ['node2'],
lock: false
}
]
}, '#element');
`
```

Objects

The layer's [objects](#) property defines the diagram elements to the layer.

```
`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram,
ej.diagrams.ConnectorModel, ej.diagrams.NodeModel, ej.diagrams.LayerModel);
// A node is created and stored in nodes array.
var nodes = [{
id: 'node1',
width: 100,
height: 100,
offsetX: 100,
offsetY: 100,
annotations: [{
content: 'Default Shape'
}]
},
{
id: 'node2',
width: 100,
height: 100,
offsetX: 300,
offsetY: 100,
shape: {
```

```
type: 'Path',
data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
content: 'Path Element'
}]
}
];
var connectors = [{
id: 'connector1',
type: 'Straight',
sourcePoint: {
x: 100,
y: 300
},
targetPoint: {
x: 200,
y: 400
},
}];
// initialize diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
connectors: connectors,
nodes: nodes,
layers: [
{
id: 'layer1',
visible: true,
```

```
objects: ['node1', 'node2']
},
{
id: 'layer2',
visible: true,
objects: ['node2']
}
]
}, '#element');
`
```

AddInfo

The [addInfo](#) property of layers allow you to maintain additional information to the layers.

The following code illustrates how to add additional information to the layers.

```
`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.Diagram,
ej.diagrams.ConnectorModel, ej.diagrams.NodeModel, ej.diagrams.LayerModel);
// A node is created and stored in nodes array.
var nodes = [{
id: 'node1',
width: 100,
height: 100,
offsetX: 100,
offsetY: 100,
annotations: [{
content: 'Default Shape'
}]
},
{
id: 'node2',
width: 100,
height: 100,
offsetX: 300,
offsetY: 100,
shape: {
```

```
type: 'Path',
data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
content: 'Path Element'
}]
}
];
var connectors = [{
id: 'connector1',
type: 'Straight',
sourcePoint: {
x: 100,
y: 300
},
targetPoint: {
x: 200,
y: 400
},
}];
var addInfo = { Description: 'Layer1' };
// initialize Diagram component
var diagram = new ej.diagrams.Diagram({
width: '100%',
height: '600px',
connectors: connectors,
nodes: nodes,
layers: [{
id: 'layer1',
visible: true,
```

```
objects: ['node1', 'node2'],
addInfo: addInfo
},
{
id: 'layer2',
visible: true,
objects: ['node2']
}
]
});
`
```

Add layer at runtime

Layers can be added at runtime by using the [addLayer](#) public method.

The layer's [ID](#) property defines the ID of the layer, and its further used to find the layer at runtime and do any customization.

The following code illustrates how to add a layer.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
// add the layers to the existing diagram layer collection
diagram.addLayer(
{
id: 'newlayer',
objects: [],
visible: true,
lock: false,
zIndex: -1
},
[
{
type: 'Straight',
sourcePoint: {
x: 100,
y: 300
},

```

```
targetPoint: {  
  x: 200,  
  y: 400  
}  
});  
`
```

Remove layer at runtime

Layers can be removed at runtime by using the [removeLayer](#) public method.

The following code illustrates how to remove a layer.

```
`javascript  
var diagramElement = document.getElementById('element');  
var diagram = diagramElement.ej2_instances[0];  
// remove the diagram layers  
diagram.removeLayer([diagram.model.layers[i]]);  
`
```

moveObjects

Objects of the layers can be moved by using the [moveObjects](#) public method.

The following code illustrates how to move objects from one layer to another layer from the diagram.

```
`javascript  
var diagramElement = document.getElementById('element');  
var diagram = diagramElement.ej2_instances[0];  
// move the objects of diagram layers  
diagram.moveObjects(['connector1'], 'layer2');  
`
```

bringLayerForward

Layers can be moved forward at runtime by using the [bringLayerForward](#) public method.

The following code illustrates how to bring forward to layer.

```
`javascript  
var diagramElement = document.getElementById('element');  
var diagram = diagramElement.ej2_instances[0];  
// move the layer forward  
diagram.bringLayerForward('layer1');  
`
```

sendLayerBackward

Layers can be moved backward at runtime by using the [sendLayerBackward](#) public method.

The following code illustrates how to send backward to layer.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
// move the layer backward
diagram.sendLayerBackward('layer1');
`
```

cloneLayer

Layers can be cloned with its object by using the [cloneLayer](#) public method.

The following code illustrates how to bring forward to layer.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram = diagramElement.ej2_instances[0];
// clone a layer with its object
diagram.cloneLayer('layer2');
`
```

getActiveLayer

To get the active layers back in diagram, use the [getActiveLayer](#) public method.

The following code illustrates how to bring forward to layer.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram: Object[] = diagramElement.ej2_instances[0];
// gets the active layer back
diagram.getActiveLayer();
`
```

setActiveLayer

Set the active layer by using the [setActiveLayer](#) public method.

The following code illustrates how to bring forward to layer.

```
`javascript
var diagramElement = document.getElementById('element');
var diagram: Object[] = diagramElement.ej2_instances[0];
// set the active layer
`
```


//@param layerName defines the name of the layer which is to be active layer

```
diagram.setActiveLayer('layer2');
```

,

Context menu in EJ2 JavaScript 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.TS

```
import {
  Diagram,
  DiagramContextMenu,
  ConnectorModel,
  NodeModel,
  DiagramBeforeMenuOpenEventArgs
} from '@syncfusion/ej2-diagrams';
import {
  MenuEventArgs
} from '@syncfusion/ej2-navigations';
Diagram.Inject(DiagramContextMenu);
//Initializes the connector
let connector: ConnectorModel = {
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  type: 'Straight',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
  }
  targetDecorator: {
    style: {
      fill : '#6BA5D7',
      strokeColor : '#6BA5D7'
    }
  }
};
//Initializes the nodes
let node: NodeModel = {
  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'
      }
    }]
  };
let node2: NodeModel = {
  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'
    }
  }]
};
//Initializes the Diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
  nodes: [node, node2],
  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.TS

```
import {
    Diagram,
    DiagramContextMenu,
    ConnectorModel,
    NodeModel,
    DiagramBeforeMenuOpenEventArgs
} from '@syncfusion/ej2-diagrams';
import {
    MenuEventArgs
} from '@syncfusion/ej2-navigations';
Diagram.Inject(DiagramContextMenu);
//Initializes the connector
let connector: ConnectorModel = {
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor : '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth : 2,
    },
    targetDecorator: {
        style: {
            fill : '#6BA5D7',
            strokeColor : '#6BA5D7'
        }
    }
};
//Initializes the nodes
let node: NodeModel = {
    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'
        }
    }]
};
let node2: NodeModel = {
    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'
  }
}]
};
//Initializes the Diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
  nodes: [node, node2],
  connectors: [connector],
  contextMenuSettings: {
    //Enables the context menu
    show: true,
    // Defines the custom context menu items
    items: [{
      // Text to be displayed
      text: 'Save',
      //Sets the id for the item
      id: 'save',
      //ContextMenu can be visible based on the target in which
you open the ContextMenu.
      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.TS

```
import {
  Diagram,
  DiagramContextMenu,
  ConnectorModel,
  NodeModel
} from '@syncfusion/ej2-diagrams';
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
import { createElement } from '@syncfusion/ej2-base';
Diagram.Inject(DiagramContextMenu);
//Initializes the connector
let connector: ConnectorModel = {
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2,
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
};
//Initializes the nodes
let node: NodeModel = {
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'labell1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
};
let node2: NodeModel = {
```

```

    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'
        }
    }]
  }];
};
//Initializes the Diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
  nodes: [node, node2],
  connectors: [connector],
  contextMenuSettings: {
    //Enables the context menu
    show: true,
    items: [{
      text: 'Cut', id: 'Cut', target: '.e-diagramcontent',
      iconCss: 'e-Cut'
    },
    {
      text: 'Copy', id: 'Copy', target: '.e-diagramcontent',
      iconCss: 'e-Copy'
    }
  ],
    // Hides the default context menu items
    showCustomMenuOnly: true,
  },
  contextMenuBeforeItemRender: (args: MenuEventArgs) => {
    // To render template in li.
    let shortCutSpan: HTMLElement = createElement('span');
    let text: string = args.item.text;
    let shortCutText: string = text === 'Cut' ? 'Ctrl + S' : (text ===
'Copy' ?
      'Ctrl + U' : 'Ctrl + Shift + I');
    shortCutSpan.textContent = shortCutText;
    args.element.appendChild(shortCutSpan);
    shortCutSpan.setAttribute('class', 'shortcut');
  }
}, '#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.TS

```

import {
  Diagram,
  DiagramContextMenu,
  DiagramBeforeMenuOpenEventArgs,
  MenuEventArgs,
  ConnectorModel,
  NodeModel
} from '@syncfusion/ej2-diagrams';
Diagram.Inject(DiagramContextMenu);

```

```
//Initializes the connector
let connector: ConnectorModel = {
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
  },
  targetDecorator: {
    style: {
      fill : '#6BA5D7',
      strokeColor : '#6BA5D7'
    }
  }
};

//Initializes the nodes
let node: NodeModel = {
  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'
    }
  }]
};

let node2: NodeModel = {
  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'
      }
    }
  ]
};
//Initializes the Diagram component
let diagram: Diagram = new Diagram({
  width: '100%',
  height: '350px',
  nodes: [node, node2],
  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: DiagramBeforeMenuOpenEventArgs) {
    //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: MenuEventArgs) {
    //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>

```

Symbol palette in EJ2 JavaScript Diagram control

The **SymbolPalette** displays a collection of palettes. The palette shows a set of nodes and connectors. It allows to drag and drop the nodes and connectors into the diagram.

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

```
<!-- markdownlint-disable MD010 -->
```

Create symbol palette

The [width](#) and [height](#) properties of the symbol palette allows to define the size of the symbol palette.

```

`javascript
ej.diagrams.Diagram.Inject(ej.diagrams.SymbolPalette);

//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
width: '100%',
height: '700px',
}, '#element');
`

```

Add palettes to SymbolPalette

A palette allows to display a group of related symbols and it textually annotates the group with its header. A [Palettes](#) can be added as a collection of symbol groups.

The collection of predefined symbols can be added in palettes using the [symbols](#) property.

To initialize a palette, define a JSON object with the property [ID](#) that is unique ID is set to the palettes.

The following code example illustrates how to define a palette and how its added to symbol palette.

INDEX.JS

```
/**
 * Default symbol palette sample
 */
//Initialize the flowshapes for the symbol palatte
var flowshapes = [{
  id: 'process',
  shape: {
    type: 'Flow',
    shape: 'Process'
  }
},
{
  id: 'document',
  shape: {
    type: 'Flow',
    shape: 'Document'
  }
},
{
  id: 'predefinedprocess',
  shape: {
    type: 'Flow',
    shape: 'PreDefinedProcess'
  }
},
{
  id: 'papertap',
  shape: {
    type: 'Flow',
    shape: 'PaperTap'
  }
},
{
  id: 'sequentialdata',
  shape: {
    type: 'Flow',
    shape: 'SequentialData'
  }
},
];
var basicShapes = [{
  id: 'Rectangle',
  shape: {
    type: 'Basic',
    shape: 'Rectangle'
  }
}
```

```
    },
    {
      id: 'Ellipse',
      shape: {
        type: 'Basic',
        shape: 'Ellipse'
      }
    },
    {
      id: 'Hexagon',
      shape: {
        type: 'Basic',
        shape: 'Hexagon'
      }
    },
    {
      id: 'Parallelogram',
      shape: {
        type: 'Basic',
        shape: 'Parallelogram'
      }
    },
    {
      id: 'Triangle',
      shape: {
        type: 'Basic',
        shape: 'Triangle'
      }
    }
  ],
];
//Initializes connector symbols for the symbol palette
var connectorSymbols = [{
  id: 'Link1',
  type: 'Orthogonal',
  sourcePoint: {
    x: 0,
    y: 0
  },
  targetPoint: {
    x: 40,
    y: 40
  },
  targetDecorator: {
    shape: 'Arrow'
  }
},
{
  id: 'link3',
  type: 'Orthogonal',
  sourcePoint: {
    x: 0,
    y: 0
  },
  targetPoint: {
    x: 40,
    y: 40
  },
}
```

```
    style: {
      strokeWidth: 2
    },
    targetDecorator: {
      shape: 'None'
    }
  },
  {
    id: 'Link21',
    type: 'Straight',
    sourcePoint: {
      x: 0,
      y: 0
    },
    targetPoint: {
      x: 40,
      y: 40
    },
    targetDecorator: {
      shape: 'Arrow'
    }
  },
  {
    id: 'link23',
    type: 'Straight',
    sourcePoint: {
      x: 0,
      y: 0
    },
    targetPoint: {
      x: 40,
      y: 40
    },
    style: {
      strokeWidth: 2
    },
    targetDecorator: {
      shape: 'None'
    }
  },
  {
    id: 'link33',
    type: 'Bezier',
    sourcePoint: {
      x: 0,
      y: 0
    },
    targetPoint: {
      x: 40,
      y: 40
    },
    style: {
      strokeWidth: 2
    },
    targetDecorator: {
      shape: 'None'
    }
  }
```

```

    }
  ];
  //Initializes the SVG shapes for the symbol palette
  var svgShapes = [
    {
      id: 'node2',
      style: {
        fill: 'none'
      },
      annotations: [{
        content: 'Start \n Text Editing'
      }],
      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-
6.858-1.849-9.298l-8.533-8.533 c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
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-
3.311,3.335-8.671,0.054-12.012l-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-
3.335,8.671-0.054,12.012l8.533,8.533 c3.341,3.281,8.701,3.256,12.012-
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-
3.311,3.335-8.671,0.054-12.012l-8.533-8.533 c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
  <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012 c-3.311-3.311-8.671-
3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/> <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/> <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/> <path
d="M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
  <path d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
  <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012 c-3.311-3.311-8.671-3.335-
12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/> <path
d="M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271s3.1,5.28,6.065,6.065 c2.965,0.785,6.122-0.082,8.271-
2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/> <path
d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
  <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.5l-8.533,8.533 c-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-
0.073,8.262-2.251l8.533-8.533 c1.601-1.6,2.5-3.77,2.5-
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-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
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-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
```

```
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
    <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.012l-8.533-8.533    c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>    <path
d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012    s-8.671-3.335-12.012-0.054l-8.533,8.533c-
2.44,2.441-3.169,6.11-
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>',
```

```

    },
    {
      id: 'syncfusion',
      style: {
        fill: 'none'
      },
      shape: {
        type: 'Native',
        content: '<g xmlns="http://www.w3.org/2000/svg">' +
          '<rect height="256" width="256" fill="#34353F"/>' +
          '<path id="path1" transform="rotate(0,128,128)'
translate(59,61.2230899333954) scale(4.3125,4.3125) " fill="#FFFFFF"
d="M18.88501,23.042998L26.804993,23.042998 26.804993,30.969001
18.88501,30.969001z M9.4360352,23.042998L17.358032,23.042998
17.358032,30.969001 9.4360352,30.969001z
M0.014038086,23.042998L7.9360352,23.042998 7.9360352,30.969001
0.014038086,30.969001z M18.871033,13.609001L26.791016,13.609001
26.791016,21.535994 18.871033,21.535994z
M9.4219971,13.609001L17.342041,13.609001 17.342041,21.535994
9.4219971,21.535994z M0,13.609001L7.9219971,13.609001 7.9219971,21.535994
0,21.535994z M9.4219971,4.1859968L17.342041,4.1859968 17.342041,12.113998
9.4219971,12.113998z M0,4.1859968L7.9219971,4.1859968 7.9219971,12.113998
0,12.113998z M25.846008,0L32,5.2310026 26.773987,11.382995
20.619019,6.155998z"/>' +
          '</g>'
      },
    },
    {
      id: 'network',
      style: {
        fill: 'none'
      },
      shape: {
        type: 'Native',
        content: '<g xmlns="http://www.w3.org/2000/svg">' +
          '<rect height="256" width="256" fill="#34353F"/>' +
          '<path id="path1" transform="rotate(0,128,128)'
translate(59.1078108549118,59) scale(4.3125,4.3125) " fill="#FFFFFF"
d="M12.12701,24.294998C12.75201,24.294998 13.258998,24.803009
13.258998,25.428009 13.258998,26.056 12.75201,26.563004 12.12701,26.563004
11.499019,26.563004 10.993007,26.056 10.993007,25.428009 10.993007,24.803009
11.499019,24.294998 12.12701,24.294998z
M7.9750035,24.294998C8.6010101,24.294998 9.1090057,24.803009
9.1090057,25.428009 9.1090057,26.056 8.6010101,26.563004 7.9750035,26.563004
7.3480199,26.563004 6.8399942,26.056 6.8399942,25.428009 6.8399942,24.803009
7.3480199,24.294998 7.9750035,24.294998z
M7.9750035,20.286011C8.6010101,20.286011 9.1090057,20.792999
9.1090057,21.419006 9.1090057,22.044006 8.6010101,22.552002
7.9750035,22.552002 7.3500035,22.552002 6.8420084,22.044006
6.8420084,21.419006 6.8420084,20.792999 7.3500035,20.286011
7.9750035,20.286011z
M18.499994,19.317001C18.313013,19.317001,18.156,19.472,18.156,19.656006L18.1
56,27.01001C18.156,27.195007,18.313013,27.350006,18.499994,27.350006L29.5219
93,27.350006C29.707998,27.350006,29.865988,27.195007,29.865988,27.01001L29.8
65988,19.656006C29.865988,19.472,29.707998,19.317001,29.521993,19.317001z
M17.243006,17.443008L30.778003,17.443008C31.425007,17.445007,31.947986,17.96

```

2006,31.950001,18.602997L31.950001,28.542007C31.947986,29.182999,31.425007,29.702011,30.778003,29.703003L25.654012,29.703003C25.511007,29.703003
25.399008,29.824997 25.413992,29.964996 25.430013,30.13501
25.452993,30.360001 25.477011,30.559998 25.506002,30.809998
25.727987,30.980011
25.976003,31.033997L27.756002,31.419006C27.907003,31.452011 28.015005,31.584
28.015005,31.738007 28.015005,31.883011 27.895986,32
27.74999,32L27.571005,32 20.450004,32 20.318016,32C20.171013,32
20.053001,31.883011 20.053001,31.738007 20.053001,31.585007
20.161003,31.452011
20.312004,31.419998L22.115989,31.033005C22.35601,30.98201
22.572014,30.815002 22.596,30.574005 22.616997,30.363007 22.636009,30.130997
22.648002,29.960007 22.658012,29.819 22.542015,29.70401
22.399986,29.70401L17.243006,29.703003C16.596002,29.702011,16.072992,29.1829
99,16.071008,28.542007L16.071008,18.602997C16.072992,17.962006,16.596002,17.
445007,17.243006,17.443008z M7.9750035,16.133011C8.6020172,16.133011
9.1100128,16.641006 9.1100128,17.268005 9.1100128,17.893997
8.6020172,18.402008 7.9750035,18.402008 7.3489964,18.402008
6.8410013,17.893997 6.8410013,17.268005 6.8410013,16.641006
7.3489964,16.133011 7.9750035,16.133011z
M24.027,13.762009C24.654014,13.762009 25.16201,14.270004 25.16201,14.895996
25.16201,15.522003 24.654014,16.029999 24.027,16.029999 23.400993,16.029999
22.892998,15.522003 22.892998,14.895996 22.892998,14.270004
23.400993,13.762009 24.027,13.762009z M24.027,9.6110077C24.653007,9.6110077
25.161003,10.119003 25.161003,10.74501 25.161003,11.37001
24.653007,11.878006 24.027,11.878006 23.402,11.878006 22.894005,11.37001
22.894005,10.74501 22.894005,10.119003 23.402,9.6110077 24.027,9.6110077z
M24.027,5.6000061C24.654014,5.6000061 25.16201,6.1080017 25.16201,6.7350006
25.16201,7.3610077 24.654014,7.8690033 24.027,7.8690033 23.400993,7.8690033
22.892998,7.3610077 22.892998,6.7350006 22.892998,6.1080017
23.400993,5.6000061 24.027,5.6000061z
M19.876001,5.6000061C20.503013,5.6000061 21.011009,6.1080017
21.011009,6.7350006 21.011009,7.3610077 20.503013,7.8690033
19.876001,7.8690033 19.249994,7.8690033 18.743006,7.3610077
18.743006,6.7350006 18.743006,6.1080017 19.249994,5.6000061
19.876001,5.6000061z
M2.4290157,1.8740082C2.2420037,1.8740082,2.0850215,2.029007,2.0850215,2.2140
045L2.0850215,9.5680084C2.0850215,9.753006,2.2420037,9.9069977,2.4290157,9.9
069977L13.451014,9.9069977C13.637995,9.9069977,13.795008,9.753006,13.795008,
9.5680084L13.795008,2.2140045C13.795008,2.029007,13.637995,1.8740082,13.4510
14,1.8740082z
M1.1730042,0L14.706996,0C15.353999,0.0019989014,15.877009,0.51899719,15.8789
93,1.1600037L15.878993,11.100006C15.877009,11.740005,15.353999,12.26001,14.7
06996,12.26001L9.5830047,12.26001C9.4399994,12.26001 9.3290069,12.382004
9.3420074,12.52301 9.3600128,12.692001 9.3829925,12.917999
9.4060028,13.117004 9.4349945,13.367004 9.6570099,13.53801
9.9049957,13.591003L11.684994,13.975998C11.835994,14.009003
11.945003,14.141998 11.945003,14.294998 11.945003,14.440002
11.826015,14.557007 11.679012,14.557007L11.499996,14.557007
4.3789966,14.557007 4.2470081,14.557007C4.1000049,14.557007
3.9819935,14.440002 3.9819937,14.294998 3.9819935,14.141998
4.0899952,14.009003
4.2409961,13.977005L6.0450113,13.589996C6.2860086,13.539001
6.501005,13.373001 6.5249918,13.130997 6.5460184,12.921005
6.5650003,12.688004 6.5769937,12.516998 6.5870035,12.376999
6.4710062,12.262009
6.3290079,12.262009L1.1730042,12.26001C0.52499391,12.26001,0.0020143806,11.7

```

40005,0,11.100006L0,1.1600037C0.0020143806,0.51899719,0.52499391,0.001998901
4,1.1730042,0z"/>' +
        '</g>'
    }
}
];
var htmlShapes = [{
    id: 'HTML',
    borderColor: 'black',
    borderWidth: 1,
    shape: {
        type: 'HTML',
        content: '<div style="height:100%;width:100%;"><button
type="button" style="width:100%;overflow:hidden">HTML</button></div>'
    }
},
{
    id: 'Checkbox',
    borderColor: 'black',
    borderWidth: 1,
    shape: {
        type: 'HTML',
        content: '<div><div style="height:50%;width:100%;"><input
type="checkbox" value="Yes" style="width:25%">Yes</input></div>' +
            '<div style="height:50%;width:100%;"><input type="checkbox"
value="No" style="width:25%">No</input></div></div>'
    }
},
{
    id: 'Dropdown',
    borderColor: 'black',
    borderWidth: 1,
    shape: {
        type: 'HTML',
        content: '<div style="height:100%;width:100%;"><select
style="width:100%"><option>SVG</option><option>Canvas</option></select></div>'
    }
},
];
//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
    palettes: [{
        id: 'flow',
        expanded: true,
        symbols: flowshapes,
        title: 'Flow Shapes'
    },
    {
        id: 'basic',
        expanded: true,
        symbols: basicShapes,
        title: 'Basic Shapes'
    },
    {
        id: 'SVG',
        expanded: true,

```

```

        symbols: svgShapes,
        title: 'SVG Shapes',
    },
    {
        id: 'html',
        expanded: true,
        symbols: htmlShapes,
        title: 'HTML Shapes'
    },
    {
        id: 'connectors',
        expanded: true,
        symbols: connectorSymbols,
        title: 'Connectors'
    }
],
width: '100%',
height: '700px',
symbolHeight: 60,
symbolWidth: 60,
});
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>
```

```
<style>
  @font-face {
    font-family: 'e-ddb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+1DAAACIAAAAIJnbHlmlw/g
RIAAAvgAACw0aGVhZBGJTLcAAADQAAAAANmhoZWEIXQqPAAAArAAAACRobXR4oAAAAAAYAAAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYW1ldAwInAAALyWAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAAFwEAAAAAAEAAAABAAAAAIAAAAKAABAAAAQAAJo24vV8
PPPUACwQAAAAAANC1g90AAAAA1zWD3QAAAAAAEAQAAAAACAACAAAAAIAAAAAEAAAAoAOWABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAA
AAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAAABAAAAQAAAAEAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAAQACAAMABAAFAAYABwAIAAKACgALAAw
ADQAOAA8AEAAARABIAEwAUABUAFgAXABgAGQAaABsAHAAAdAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMIIJaAnaClYLMauqC7gMpg2ODmQ
Owg8ad9IQoBF6EL1YTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYffQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFbY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVfwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxQEBEQERIRDw8PDw4
PDg4NDhoZGBP6XfoYegkICQcIBgYGBQQAeAwMCAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPER
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBAOLdg8REhQWFgWLDQwMDQ0NDA0MDAwLfxUUExEpDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLcgoJCAgHBgYUFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAwwKcW4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEawMCAgH9rgEBAGQDBQYGBwcJCQkLCwsMDQ0NDg4PDxQEBEQERIRDw8PDw4PDg4
NDhoZGBP6XvOYewkJCAgHBwYFBQUDAwMCAgICAwwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAREBERERI
REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDAwNCwwWFhQSEQ8OCwoEawMCAgE
BAgIDAwwKcW4PERIUfHYMCw0MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxE
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAwMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkaADchNSETFR8dOwE/HTU
RIxEpDy8PayOJAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBARESEhERERAQDg4NCwUJCAyEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAgGBwUFBQAQDAgEBAgMEBAUFBwYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTEExERDw4OCwsJBwYFAgEBAgUGBwLcW0PBxAREhMUAcAAAAABAAAAAAD9AO
1AAMABwAvADMAAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIWHC/scBOXwBof7
HfAI+fP7HATl8ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfzMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9fT4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQKQKwoJBwUCAGUHCQoLCgQAAAAAagAAAAADtQP0AdcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBwZ9Pv2QPn0GBwYLCgghBQEB+X58frwCvP2OBgYGCwoJBgUCAQEBCQYJCgsGBgY
CcgyEBGgsKCQYFAGF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAAADAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMFDR0BDw4JjNQhPw8vDz8MLw8hAi8KCQkJCACIBgYGBAQEAgEBAQEBAQEAgYGCACJCAk
JCpx9CQoJCAgIBwCGBQEAwMBAQMDBAUFBgCHAgICQoJfbwBhxQVExMRERAODQwKCQcFawEBAQM
EBAYGCAgJCQsLCwwNEwAPBgYFBQQDAwIBAQEBCAcICgWNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICAcHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwGGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCAG
JCQoLDAwMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+IX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAGMFBQYHCAkKCgsMDAwNABUNDawMCwoKCQgHBgU
```

FAWJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCACFAQH9kLv6Au76+okGBwYLCggHBQEBAQEBAQUHCAoLBgcGAj07fx0/AAA
AAQAAAAADdwN3ANEAAABMhJz8L0wEfHr0BDx0jLw8jHx47AT8dPQEvHSMpDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwCGBQUFBAMCAgEBAgIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUfBMSEA8NDAsIB14EBAQFBgcHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTExISEhIREBAQDw8ODg0MDAsLCQoIBwCGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTExISExESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQcJCwwODxESExU
VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwCICgkLCwwMDQ4ODw8QEBA
REhISEhMTExMTExISEhIREBAQDw8ODg0MDAsLCQoIBwCGBQQEAgIBAQIEBAUHBggJCQoLCwwNCQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXuzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAwQFCg0OEBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgcJCQoMDA4ODxARERISFBMVFBVUFbQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDAMDAWFRMSEA4MCwUEAwMCAgEBAgIDAwQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFbQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUHNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTExAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAwMDA0MDQwNDQwMDAwMCxYUEXIQDgWLBaQEAgICAQECAGMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCgsNDQ4PEBAREhM
TExQVFRoaGhkZFxYWAVEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgcICQoMDA0PDw8RERI
SExQUFBVUFbQCbzfLBgsODxEsFBYWDawMDAwNDQwNDA0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAwMDA0MDA0NDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVUFbMTExIREQ8QDg0NDAoKCACGBQQAQEEBQgKDA4Q/qseAgICAgQEBQUFBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAO8AAMAIwArAC8
ASgAAARUhNscPAh0BHwU7AT8FPQEvBSsBDwEIESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQEAgICAgQEBQUGBQUGFasan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFbQYFBQQEAgICAgQEBQUGBQUGFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAet+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYE/BhEvBiEPBgJvpzc4Ab391DcMjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVarX9Lm8CBAYICQoKAxYKCGkIBgQCAQMFBwgKCgAAAAA
DAAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRakwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbGcJCgoMDA4OEBAREhITGRgWfxcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCfBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEawIBWyIDBQYICQsNDQ8RERMUFURUXGBgzDRobG0cTExiQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUakAAAATmFDROBDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBgcICGoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCAgHBwYGBQQEAWMCAQECAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAgQGCAoLDhAREhQVFXga9wHIAQIDBAUFBwCICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAD4BTgEBAgMDBAUGBwCJCQkLCwwPDQwMCwoJCQcHBQQEAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwCICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAAD9AN3AAMAHwBUAAABAYETJzmFDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ0OEakKAXL+IAkJCAcHBwUFlhkFCgkGBQIBAxXMAwICAQIFBgkKCwYghAEBBQc
ICgsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwCQAwcGBAIBfQEBAwQFBgcI/tMCCzo
CBwkKCwYg/UoBmgcHBwCGBgYLCgkGBQIBgwcGCwoIBwUBAQBQAQIFBwCQAwcGBAIBQAIAAAAAAGa
AAAAADaQ08AAMABwALAB8AIwBeAAALMxEjAzMRiWmZESMLEQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwCICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBwV/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwC
ICAgJan83PgSKAgGBAIBAAABAAAAA08A7wAXgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQgKDg4cGhoZFxcVFBMQEDfegQ0OEBITFBWGBgZGhsbGxwaGhoZGRcXfHUUFBIREA4ODA0JCAY
FAGEBAGUGCAkKDA4OEBESFBQVFhcXGQwaGRsdEBAQE8PDw8PDg4ODQ0MDAwLCwsKChIIBwCHBgU
ENgUGBwCICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM

BAQMFbgkLDA0PERIUFBYXGRkaGxwdHR4eHh4dA60FBAsMDhARExQWGBgad984GRcXFRQSEQ8ODAO
JBgUDAQECBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwgICQkKCgMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKCQkIBwcHBQUEAwM
CAQEDBACJCwwNDxEXEXUWFxkZGhscHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODAOJBwQDAQMFAAAAAGAA
AAADFQ08AAMAAaAAANYE1IREfHjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwcGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhCWfHqVEXERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwcICakJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww
NDxEREXQVFhYB/wAAAAEAAAAAArEDvAADAAAlMwEjAU86ASg6RAN4AAADAAAAAAOQA5AACwBMANM
AAAEjFTVMVzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBawUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEEXER/vUPDw8NDgwMDAsLCgkJCAcHBwUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBQEEREBGrkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQcEBAgMEBAYGBwc
ICQoJCwsMDAwODQ8PDxAQEBARERASERARERAEAJkZGRkZGQOCakRERMTFRUWFRUVExMREREDQs
JBwUDAQEDBQcJCw0OEREREXMVFRUWFRUTEEXEREQ4NCwkHBQMBAQMFbWkLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEEREBERESEBEREBAQEAPDw0ODA0LCwsKCQkIBwcHBQUDAwICAQMEBwgLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqgFhYXGBgZGRsRERAREBAQEAPDw0ODA0LCwsKCQkIBwcHBQUDAwICAQI
DAwUFAAMAAAAA5ADKAADAEQAYwAAASE1ISUFCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPw
vHisBDwUBOAEs/tQBIwCHDQsJBwUDAQEDBQcJCw0OEREREXQVFYVFRUTEEXIREA8MCwkHBQMBAQM
FBWkLDA8QERITEEXUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwcHBQUDAwICAQIDAUFBwCHCAk
JCgsLCw0MDg0PDhAQEBQEERARERsZGRgYfXyWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4MCwgHBAM
BAQECaWQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEEREBIREBEREBAQAgBkcggJERETEXUVFhUVFRM
TERERDg0LCQcFAwEBawUHCQsNDhERERMTRUFVfHUVExMREREDQsJBwUDAQEDBQcJCw2RBwcICQk
KCwsLDQwODQ8PDxAQEBAREBEREHARERAQEBAPDw8NDgWNCwsLCgkJCAcHBwUFAwMCAgEDBACICww
OqQQDAgICAQICAwJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBAPDw8NDgWNCwsLCgkJCAcHBwU
FAwMCAgICAwMFBQAAAAGAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARC7AR8KDxArAS8WPwg
nNw8BJyMfCRUFgJ7WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAGICAwP8+gMDAg8
HOGUFBgkJAwQDAgULAQEDBAIFBwcLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAWQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQAFBgYHCAGKCgsMDQ4PEBASEhMTFRULIhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBiIKCwsMAGMKOCN1LM4CAwNJAwmCAgIBAQICAgMDSQM
DAgICAQECaPMBAgIFCAMJCw89fVYjHhgLDw8OEwWNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJKxIAGFAGIEAQIDJgcEAQYuAwMEBAQFBBE14jgfgH0ODg0MDAsKCgkICQcIBgcFBQQAQAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFQBUBAQcCagIQGwEFBQAEAAAAAAOQA5AAAwAjACC
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAQm
EBAUFBBQFAwQCAgICBAMFBAUFBBQBM/7UZDIYcQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAWMDAQEBAMDBAQEBQFBAQDAgIBA+WWlpYBBQFbgYHCAj+opaWAV4HCAsGBwUEAvO
AAAAEAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBAcICQoL+QIEBwgJCgtjCgoJCAcEAvkLCgkIBwQCAgQHCAkKC/kCBACICQoKXGsKCgOABQk
KCvoCBACICQoLYwoKCQgHBAL5CwoJCAcEAgIEBwgJCgv5AgQHCAkKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADWgPCAMABwAJAFUAmwAAARUHNQEVIzUHNsmVHw8hPw81FxEjNS8PIQ8PFsm
RNQ8PER8PIT8PETUVdzECyP5wASyWlmQBAQIEBAUGBgICakJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgICakJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKArwKCgoJCQgIBwYGBQQAQEBAGIDBAQGBp8HBwcICAgJCgFqyMgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAWMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKyAK8ZAEBAgQEBQYGBwgICQkKCGr9RAoKCgkJCAgHBgYFBQAQAEBAQIEBAU
GBgcICakJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAAOQA5AAbQCxAAABHwQPCc8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAGMEuAUFBgCGAwgFAwMCAgE
jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBQDDBAMDCMRDQsIAwMBAQECAwIHBQUGBwKCGw
NDw8REhQWGBocHB8BAgIDAUFBwcGBQX+JgoJCAyFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQCAwUGCAkKCwwNDA4ODw/+DA8PDg4NDAwDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEB1MBAgQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgWNDAwbHRsOHw8
PDQ0NDA0MDAsJCQgHBgYEawIBUwUFBQQDBAMCAgEBAGMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAGMFBggJCgsMDQ0NDg8PAfQPDw4NDQ0MCwoJCAyFAwIBAQIDBQYICQAAAaAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAgEBAGQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUFCAZBBQiHhEQ2Q+IAioZEwKGAQECBQCBQMDAwUaRQHxyRcXfHuwFRUUEX
QBw4MCwkDBAICAgEBawQGBwkLDQ0PEBAREhMTDSCTFQkIBgYFBQQAQEBAGMEBggJCwsNDQ8PERA

RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBawUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBAwEUawQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgCICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCACGBQYUCw8IBwcICQoLDaw
MDhMSEhAQDg0MCgoJCACGBQQDAgEBAAAAAAMAAAAA/QDcAAqAFYAuQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVBwUFBAMCAgEBawSaCAg
KDAsMCwv9wAYFAwMDAQIDBJoICAoMCwWLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhgIEBQcJCQlNCAgFBQQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwWm/uMCAGQFBGcICQkKCgsLCwyoCwWLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQEBAgMDAwUEBQYGVggHBwUFBaIBAU8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgCkCwWNPQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQECBacJCgW
MxQgIBwgICAgICQkJCQYKQgHBAQBVawMCwoKCgkJCACGBQQDAQEQDAwLCgoKCQkIBwYFBAMBAQE
BwQFBGcAAAAABQAAAAADXgOQACEAQwBlAGkAxQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUBAQ
DAgEBAQECAwQEBQUFBQQEAWIBfAEBAgMEBAUFBQUEBAMCAQEBAQIDBAQFBQUBAQDAgF8AQECawQ
EBQUFBQQEAWIBAQEBAgMEBAUFBQUEBAMCAbAU1xRCIn0FBQQEAWIBAQEBAgMEBAUFGQIBAwMEBAU
FBgYHBwCHCAHCCAChBwCGBgUFBaQDAwECGQUFBaQDAgEBAQECAwQEBQWWIgQFBwCICakKvwkKCAg
HBwUCcP68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQ
EBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEZzI
yJFYBAQIDBAQFBRkFBQQEAWIBaf3zCACHBwCGBgUFBaQDAwECAGEDAwQEBQUGBgCHBwCICAg0BAQI
DBAQFBRkFBQQEAWIBAVYICACFBQMCAQECAwUFBWgAAAAAAQAAAAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHisBDwUBbBIRERAPEA4OSAQFBAUEBQoEBAMCAgE
BAgMEBQYGBuoFBQQEBA MDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBWgHBgYKBQM
DAwEBAQEBAQMDAwUKBgYHCACJCQkJCgsLCwWMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgsMAgQDBAQ
DAkGDAQMPDxARERMTFBQUFRUWFhYWFBUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAQM
DBAYGCAgJCgoLDawNDg4PEBAREhITEhMUExQUFBMTExITEhIDcwcJCQoKCw0MRgMCAgEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQEBSRDgWKAYEAQEBAQIDBAQFDAYHBWgJCAkKCgsKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBWgHBgYGBAUDAwMBAQEBAQIDBAUFBggHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgsMDA0ODg0QEBESEhITExQTFBQUFBQExQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAAATMfBBUHCwEPBjc
fAj8CLwE3Ez8GBysBLwEbKAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBatjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRI tCgEGBAIbGBAPLwGZiIEKBB0YFggBBwAABAAAAAAEAQAAMABwA
LACMAAAEVITUHfSE1ARUHnQMzFSERIxEhESM1IRUjESERIxEhNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAAUAAAAAAQAAAAEAAQ
AAHYAAAEHIREhLwcPDx8PPw8hETmfDz8PLw8PBgMSAf7v/u8LCwWNDw8REQ0NDawLCwkKCAChBQO
DAgEBAgMEBQCHCAoJCwsMDA0NDQ0MDAsLCQoIBwCFBAMCAQFAwAECawQFBwCICgkLCwWMDQ0NDQW
MCwsJCggHBwUEAwIBAQIDBAUHBWgKCQsLDawNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBWg
KCQsLDawNDQ0NDawLCwkKCAChBQQDAgEBAgMEBQCHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBWgKCQsLDawNDQ0NDawLCwkKCAChBQQDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAAwAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBWgEBAQIDBQY
HCAoKDAwNDw8PejP92QEcAkABBAUICQsNDxAREhQUFhYXfYVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVFhCXfHYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwCFBQUEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQQBAQFCAkLDQ8QERIUFBYWFxwFhQUEhE
QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBGcICQoLDawNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwCHCAkJCQoLCwsMDawNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAAEAAAAABAAAAAABABsAAQABAAAAAAACAACHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAECCQAAAAIAVQADAAEECCQABADYAvwADAAEECCQAGADYBhQADAAEECCQAKAFgBuwADAAEECCQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJ1aWwXkZXJSZWdlbGZyTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lvbiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ1
1aWwXkZXJGbz250IGdlbmVyYXRlZCB1c2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHCAIABNAGEAdABLAHIAaQBhAGwAXwBEAGkAYQB
nAHIAAYQBTAEIADQBpAGwAZABLAHIAATgBlAHCAIABNAGEAdABLAHIAaQBhAGwAXwBEAGkAYQBnAHIA
AYQBTAEIADQBpAGwAZABLAHIAVgBlAHIAcWBPAG8AbgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQBy
YAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAABnAGUAbgBlAHIA

```

AYQB0AGUAZAAGAHUAacwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgBAgEDAQQBBQEGAQcBCAEJAQoBCwEMAQ0BDgEPARA
BEQESARMBFAEVARYBfWfEYARKBGgEbARwBHQEeAR8BIAEhASIBIwEkASUBJgEnASgBKQAHWm9vbU1
uTQhab29tT3V0TQpVbmR1cmxpbmVNB1ByaW50TQROZXdnbnVhdmVNB0V4cG9ydE0FQm9sZE0LT3B
lbkZvbGRlck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pvb21JbkYlWm9vbU91dEYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVGCFJlZnJlc2hGClV
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFpvb21PdXRCClVuZGVybgGluZUIGUHJpbnRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXB1cw1Db25uZWN0b3ILQmFzaWNTaGFwZXMAAAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-ddb-icons {
    font-family: 'e-ddb-icons';
    speak: none;
    font-size: 16px;
    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-basic::before {
    content: "\e726";
}
.e-flow::before {
    content: "\e724";
}
.e-connector::before {
    content: "\e725";
}
</style>
<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 palette header

Palettes can be annotated with its header texts.

The [title](#) displayed as the header text of palette.

The [expanded](#) property of palette allows to expand/collapse its palette items.

The [height](#) property of palette sets the height of the symbol group.

The [iconCss](#) property sets the content of the symbol group.

The [description](#) defines the text to be displayed and how that is to be handled in `getSymbolInfo`.

Also, any HTML element into a palette header can be embedded by defining the `getSymbolInfo` property.

The following code example illustrates how to customize palette headers.

INDEX.JS

```
/**
 * Default symbol palette sample
 */
//Initialize the basicshapes for the symbol palatte
var basicShapes = [{
  id: 'Rectangle',
  shape: {
    type: 'Basic',
    shape: 'Rectangle'
  }
},
{
  id: 'Ellipse',
  shape: {
    type: 'Basic',
    shape: 'Ellipse'
  }
},
{
  id: 'Hexagon',
  shape: {
    type: 'Basic',
    shape: 'Hexagon'
  }
}
];
//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
  expandMode: 'Multiple',
  palettes: [{
    id: 'basic',
    expanded: true,
    symbols: basicShapes,
    title: 'Basic Shapes',
    iconCss: 'e-ddb-icons e-basic'
  }],
  symbolHeight: 80,
  symbolWidth: 80,
  getSymbolInfo: (symbol) => {
    if (symbol['text'] !== undefined) {
      return {
        width: 75,
        height: 40,
        description: {
          text: symbol['text'],
          overflow: 'Wrap'
        }
      };
    }
  }
});
return {
```

```

        width: 75,
        height: 40,
        description: {
            text: symbol.shape['shape']
        }
    };
}
});
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>
<style>
  @font-face {
    font-family: 'e-ddb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjltShgAAAEoAAAVmNtYXDon+lDAAACIAAAAIJnbHlmw/g
RIAAAavgAACw0aGVhZBGJTLcAAADQAAAAANmhoZWEIXQpAAAArAAAACRobXR4oAAAAAAYAAAAC
gbG9jYdYyYe4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYW1ldAwInAAALywAAAMVcG9zdNAiwIs
AADJEAAABuQABAAAEAAAAAFwEAAAAAAEAAABAAAAAIAAAABAAAAAQAABAAAAQAABAAAAQAAB
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAIAAAABAAAAAQAABAAA
AAgAAAAoACgAAAP8AAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAAB
AAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAIAAAABAAAAAQAABAAAAQAABAAAAQAAB
EAAAABAAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAA
AAAAEAAAABAAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAAAAQAABAA
```

AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAATAAADAAAAFAADAAEAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAQACAAMABAAFAAYABwAIAAKACgALAaw
ADQAOAA8AEAARABIAEwAUABUAFgAXABgAGQAaABsAHAAdAB4AHwAgACEAIgAjACQAjQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEBZwGkgd8B+YH/ghMCMIJJaAnaClYLMauQc7gMpg2ODmQ
Owg8aD9IQoBF6E1YTRhRGFIQUwBVMFhoAAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBy
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDaKEBAMCAQEBAQEBAgMEBAkMDQ8RExQVfwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBp6XfoYegkICQcIBgYGBQQEAWMCAQEBAgMEBQUGBwgICQoKCwMDA0ODg4PDxAPERA
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAGHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBAoLDg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUEXEPDQwKAwQDAgEBAQEBAQI
LBAMKDA0PERMUFRcLDAwMDQwNEhERERAREAE8PDw4ODg0MDAwLCgoJCAGHBgYUFBAMCAgECaWMDbQU
FBw0QEhMy+176EwsLDaWNDQ4ODg8ODw8PEA8REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwCGBgUDBAI
BAQEBAgQDBQYGBwCJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXWdjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4Phu0BOP7IAZYBAgIDAwQKCw4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCGwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAGQDBQYGBwCJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBp6XvOYEWKJCAGHBWYFBQUdAwMCAQICAwQFBQYHCAgJCgoLDaWMDQ4ODg8PDxAREBERERI
REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwCGBgUDBAIBAlc/Hw0NDAwNCwwWFhQSEQ8OCwoEAWMCAgE
BAgIDAwQKCw4PERIUfHYMCw0MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCGwNDxE
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAwMCwoKCQgIBWYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEpDy8Pay0JAu79Ej8BAgMDBQqGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBARESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAGGBWUFBQDAgEBAGMEBAUFbWYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTExERDw4OCwsJBWYFAGEBAGUGBwkLCw0PBxAREhMUAcAAAAABAAAAAAD9AO
1AAMABwAvADMAAEVITU1FSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBWYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBWYGBAM
QCECAwQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIWHC/scBOxWbOf7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUZNSMBFSM1IxUhNSMRFzMRIRE7AT8HNRE
1LwcjISMPBwGdfX0BT4+/kp9fT4BeHwFBAoLDcGkHBQICBQcJCgsKBAX9kAUECGsKCQCfAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQKQKwoJBWUCAGUHCQoLCgQAAAAAAGAAAAADTQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBWYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBWZ9Pv2QPn0GBWYLCgGHBQEB+X58frwCvP2OBgYGCwoJBgUCAQEcbQYJCgsGBgY
CcgYGBgsKCQYFAGf9/gwB9H0BAGUGCQoLBgZ2/ooBdrwAAAAADAAAAAMoA3cAIGBFAIUAAAEfDw8
OKwE1EzmfDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCACIBgYGBAQEAQEBQEBAQEBAgYGCACJCAk
JCpx9CQoJCAGIBwCGBQEAwMBAQMDBAUFBgcHCAgICQoJfbwBhxQVExMRERAODQwKCQCFAwEBAQM
EBAYGCAgJCQsLCwwNEXAPBgYFBQQDAwIBAQEcbACICGwNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICACHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAWUHCQoLDg4QEBITExQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQN3/RJ9AQIDBAUGCAG
JCQoLDaWMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+ix36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBWYFBQMCAGMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBWYB9AYHBgsKCAcFAQH9kLv6Au76+okGBWYLCgGHBQEBQEBAQEUHCAoLBgcGAj07fX0/AAA
AAQAAAAADdwN3ANEABMhJz8LowEfHR0BDx0jLw8jHx47AT8dPQEvHSPMDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwCGBQUBFAMCAgEBAgIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUFBMSEA8NDAsIB14EBAQFBgcHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTExISEhIREBAQDw8ODg0MDAsLCQoIBwCGBQQAQEAQICAgQEBQYHBwgKCQsLDaWNDg4PDxAQEBE
SEhISExMTExISExESEREREAE8QDg8NDXECpooJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDaWMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBWYFBQQAQEAQICAgQEDBQcJCwwODxESExU
VFhCQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBWYFBQQAQEAQICAgIEBAUGBwCICgkLCwwMDQ4ODw8QEB
REhISEhMTExMTExISEhIREBAQDw8ODg0MDAsLCQoIBwCGBQQAQEAQICAgIEBAUGBwCICgkLCwwMDQ4ODw8QEB
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAwQFCg00EBITFRYLDaW

MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgcJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDAwMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFBQVFRQ
VExQSEhEREa8ODgWMCgkJBwYFAwIBAgMGAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUEXiQDgwLBAQEAgICAQECAGMEBAoLDg8
REhQVFfwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCGsNDQ4PEBAREhM
TExQVFRoaGhkZFxYWAWEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgcICQoMDA0PDw8RERI
SExQUFBVFBQCbzfLBgsODxESFBYWDawMDAwNDQwNDAw0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVFBMTExIREQ8QDg0NDAoKCACGBQQAQEEBQgKDA4Q/qseAgICAgQEBQUFBQY
EBQFVFRYYGBkZGhsVFBQUExMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvA08AAMAIwArAC8
ASgAAARUhNScPah0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQAeAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFbQYFBQQAeAgICAgQEBQUGBQUFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERiXehETMRIxehESM
nESMRfYyE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAC
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVArX9Lm8CBAYICQoKAxYKCGkIBgQCAQMFBwgKCGAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUExMFBgcJCgoMDA40EBAREhITGRgWfXcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBWUfHgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCAgIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEAWIBWyIDBQYICQsNDQ8RERMUFURUXGBgZDRobG0cTExiQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUakAAAATmFDROBDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBgcICgoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCAGHBwYGBQQAeAwMCAQECAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAQUGCAoLDhAREhQVFxga9wHIAQIDBAUFbwcICQoLCw0NDQwLDA0JCQgHBgUEBAI
BAd4BTgEBAgMDBAgBwJCQkLCwwPDQwMCwoJCQkHBQQAeAgLe/WUCBAYICQkKCCQoLCgS
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwcICQoLDawNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABAYETJzmFDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ00EakKAXL+IAkJCAChBwUfLhkFCgkGBQIBAxXMAwICAQIFBgkKCwYGHAEbBQc
ICGsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwcQAwcGBAIBfQEBAwQFBgcI/tMCCzo
CBwkKCwYG/UoBmgCHBwcGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwcQAwcGBAIBAQIAAAAABgA
AAAADAQ08AAMABwALAB8AIwBeAAALMxEjAzMRIwMzESM1EQ8HIS8GNRELFSM1Jw8FFSMVMxEfdjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwcICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBvW/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBgJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwc
ICAgJAn83PgSKAgGBAIBAAABAAAAA08A7wAXgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQKGd4cGhoZFxcVFBMQEDfegQ00EBITFBWUGBgZGhsbGxwaGhoZGRcXFhUUFBIREA40DAoJCAY
FAGEBAGUGCAkKDA40EBESFBQVFhcXGQwaGRsdEBAQEa8PDw8PDg4ODQ0MDAwLCwsKChIIBwcHBgU
ENgUGBwcICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhARExQWGBgad984GRcXFRQSEQ80DAo
JBgUDAQECEBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTEExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwgICQkKCgoMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKCQkIBwcHBQEAwM
CAQEDBACJCwwNDxESExUWFxkZGhsCHR0eHh4eHR0cGxoZGRcWFBQSEQ80DAoJBwQDAQMFAAAAAAG
AAAADFQ08AAMAAaAAANyE1IREfHjsBPx4RIxEpDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwcGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhcWfHqVExERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwcICakJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww
NDxERExQVFhYB/wAAAAEAAAAAArEDvAADAAALMwEjAU86ASg6RAN4AAADAAAAAAQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBAwUHCQsNDhERERMUFBUFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEER/vUPDw8NDgwMDAsLCgkJCAChBwUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWfQoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwc

ICQoJCwsMDAwODQ8PDxAQEBAERERASERARERAQEAJkZGRkZGQOCakRERMTFRUWFRUVExMREREODQs
JBwUDAQEDBQcJCw00ERERExMVFRUWFRUTExEREQ4NCwkHBQMBQMFbWkLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQEA8PDw0ODA0LCwsKCQkIBwCHBQUDAwICAQMEBwgLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqqFhYXGBgZGRsRERAREBAQEA8PDw0ODA0LCwsKCQkIBwCHBQUDAwICAQI
DAwUFAAMAAAAA5ADkAADAEQAYwAAASE1ISUfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwc
vHisBDwUBOAes/tQBIwCHDQsJBwUDAQEDBQcJCw00ERERExQUFRYVFRUTExIREA8MCwkHBQMBQMF
BwLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUDAwICAQIDAUFbWCHCAk
JCgsLCw0MDg0PDhAQEBAQERARERsZGRgYFxyWqgQFBgUGBg0MBQUKCQcDAQMDAQMHqQ4MCwgHBAM
BAQECawQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBAwUHCQsNDhERERMTFRUVFhUVExMREREODQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxAQEBAEREBERehARERAQEBAPDw8NDgwnCwsLCgkJCAcHBwUFAwMCAgEDBACICww
QoQQDAgICAQICAwCJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBAPDw8NDgwnCwsLCgkJCAcHBwU
FAwMCAgICAQMFBAQAAAgAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARc7AR8KDxArAs8WPwg
nNw8BJyMfCRUfGj8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAGICAwP8+gMDAg8
HOGUFBGkJAwQDAgULAQEDBAIFBwCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBAQFBgYHCAgKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBiIKCwsMAGMKOCN1LM4CAwNJAwmCAgIBAQICAgMDSQM
DAgICAQECapMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECawMEBQYECwYGBg8KDAwNDQ4
PEJKxIAGfAgIEAQIDJgcEAQYuAwMEBAQFBBe14jgfgHhODg0MDAsKCgkICQcIBgcFBQOEAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFBUBAQcCagIQGwEFBQAEAAAAAQA5AAAwAjACc
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAQm
EBAUFbQQFAwQCAgICBAMFBAUFbQQBm/7UZDIyCQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFbQQEBAMDAQEBAQMDBAQEBQUFBAQDAgIBA+WWlpYBBQQFBgYHCAj+opaWAV4HCAsGBwUEAvo
AAAAEAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBAcICQoL+QIEBwgJCgtjCgoJCACeAvkLCgkIBwQCAgQHCAkKC/kCBACICQoKXGsKCgOABQk
KCvoCBACICQoLYwoKCQgHBAL5CwoJCACeAgIEBwgJCgv5AgQHCAkKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwgPCAAMABwAJAFUAmwAAARUHNQEVIzUHNsmVHw8hPw81FxEjNS8PIQ8PFsM
RNQ8PER8PIT8PETUvDzECyP5wASyWlmQBAQIEBAUGBgCICakJCgoKASwKCgoJCQgIBwYGBQODAwE
BljIBAQMDBAUGBgCICakJCgoK/nAKCgoJCQgIBwYGBQODAwEBMgoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgCICakJCgoKArwKCgoJCQgIBwYGBQOEAgEBAgIDBAQGBp8HBwCICAgJCgFqyMgB9MjIyMj
ICgoKCgkICACGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgCICakJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAGHBgYFBAQCAQEBAQIEBAU
GBgcICakJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAQA5AAAbQCxAAABHwQPCC8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAgMEuAUFbGcGAwGFawMCAgE
jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDCMRDQsIAwMBAQECAwIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAGIDAUFbWcGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECAwUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEB1MBAgQFBAMEBQUGBgCICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgwNDawbHRSOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAgEBAgMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAgMFBggJCgsMDQ0NDg8PafQPdW4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8Baz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAKHCAUEBAMCAgEBAgQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBQMF
EBgcJCgwODg4QEBIUfCAZBBQiHhEQ2Q+iAioZEwKGAQECBQCBQMDAwUaRQHxyRcXFhUWFRUUExE
QBw4MCwkDBAICAgEBAwQGBwkLDQ0PEBAREhMTDScTFQkIBgYFBQOEAwEBAQMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBAwUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBAwEUawQBazUGKwQEBAMEAgILVv4rIR4ICAC
BCA0xCwICAgMEBgCICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCACGBQYUCw8IBwCICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQQDAgEBAwAAAAAAMAAAAA/QDcAAqAFYAUAQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbWUFBAwMCAgEBAwSaCag
KDAwMCww9wAYFAwMDAQIDBjJoICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhgIEBQcJCQlNCAgFBQQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAGIEBQYHCAkJCgoKCwwM/uMCAgQFBGcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQEBAgMDAwUEBQYGVggHBwUFAIBAUA8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgCkCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQEBCACJCgw
MxQgIBwgICAgICQkJCQYKCQgHBAQBVawMCwoKCgkJCACGBQQDAQEQDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBGcAAAAABQAAAAADXgOQACEAQwBLAGkAxQAAAREPBY8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIY8IIw8GApYBAQIDBAQFBQUFBAQ


```
DAgEBAQECAwQEBQUFBQQEAwIBfAEBAgMEBAUFQBUEBAMCAQEBAQIDBAQFBQUFBAQDAgF8AQECAwQ
EBQUFBQQEAwIBAQEBAgMEBAUFQBUEBAMCAbAU1xRCIn0FBQQEAwIBAQEBAgMEBAUFQIBAwMEBAU
FBgYHBwcHCAHCCaCHBwcGBgUFBAQDAwECGQUFBAQDAgEBAQECAwQEBQWWIqQFBwcICakKvwkKCAg
HBwUCCp68BgQEBAQMDAQEBAMDBAQEBgFEBgQEBAQMDAQEBAMDBAQEBv68BgQEBAQMDAQEBAMDBAQ
EBgFEBgQEBAQMDAQEBAMDBAQEBv68BgQEBAQMDAQEBAMDBAQEBgFEBgQEBAQMDAQEBAMDBAQEBzI
yJFYBAQIDBAQFBRkFBQQEAwIBaf3zCAChBwcGBgUFBAQDAwECAGEDAwQEBQUGBGcHBwcIAg0BAQI
DBAQFBRkFBQQEAwIBAVYICaCFBQMCAQECAwUFBwgAAAAAAQAAAAADjwOPA0gAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHisBDwUBbIRERAPEA4OSAQFBAUEBQoEBAMCAgE
BAgMEBQYGBuofBQQEBAMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBwgHBgYKBQM
DAwEBAQEBAQMDAwUKBgYHCACJCQkJCgSLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDAMCgSMagQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWFhYWFBUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAQM
DBAYGCAGJCgoLDAwNDg4PEBAREhITEhMUEXQUBMTExITEhIDcwcJCQoKCw0MRgMCAgEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQUEBUsRDgwKCAyEAQEBAQIDBAQFDAYHBwgJCAkKCgSKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBggHCQkKCws0AgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgSMDA0ODg8QEBESEhITEhXQTFBQUFBQUEXQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAAATmFBBUHCwEPBjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBATjBgSNGR8gJAN
aAwQDAwMNF/7x/soPDAoHBRITcgEGBAIBGBAPLwGZiiEKBBOYFggBBwAABAAAAAAEAAQAAAMABwA
LACMAAAEVITUhFSE1ARUhNQmzFSERiXehESM1IRUjESERiXehNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAUAAAAAAQAAAAEAAQ
AAHYAAAEHIREhLwcPDx8PPw8hETMfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDAwLCwkKCAChBQ
DAgEBAgMEBQcHCAoJCwsMDA0NDQ0MDAsLCQoIBwcFBAMCAQFAwAECawQFBwcICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDAwLCwkKCAChBQQDAgEBAgMEBQcHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDAwLCwkKCAChBQQDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAAwAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
HCAoKDAwNDw8PEjP92QEcAkABBAUICQsNDxAREhQUFhYXfXyVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVFhcXfYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDAMCwsKCwkJCQgHBwcFBQUEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQQAQFCakLDQ8QERIUFBYWFxcWFhQUEhE
QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBgcICQoLDAwNDg4PFqf/AAIA/cd+gIMCAQE
CAwMEBQUFBwcHCAkJCQoLCwsMDAwNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAAEAAAAABAAAAAABABsAAQABAAAAAAACAAcAHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAVQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJ1aWxkZXJSZWd1bGZyTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lvbiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ
1aWxkZXJGb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQB
nAHIAIYQBtAEIAdQBpAGwAZABlAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQBnAH
IAYQBtAEIAdQBpAGwAZABlAHIAVgBlAHIAcWBPAG8AbgAgADEALGAWAE4AZQB3ACAATQBhAHQAZQB
yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAABNAGUAbgBlAH
IAYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAoAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAAB
BEQESARMBFAEYARkBFwEYARkBGgEbARwBHQEAEAR8BIAEhASIBIwEkASUBJgEnASgBKQAHWm9vbU
lTQhab29tT3V0TQpVbmRlcmxpbmVNB1ByaW50TQROZXdNBVNhdnVNB0V4cG9yde0FQm9sZU0LT3B
lbkZvbGR1ck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsZWNNB1pvb21JbkYlWm9vbU9ldEYGUHJpbmR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVcb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVGCFJlZnJlc2hGClV
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFpvb21PdXRCClVuZGVybgGluZUIGUHVpbmRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVcb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXBlcwldDb25uZWNOB3ILQmFzaWNTaGFwZXMAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
```

```

.e-ddb-icons {
    font-family: 'e-ddb-icons';
    speak: none;
    font-size: 16px;
    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-basic::before {
    content: "\e726";
}
.e-flow::before {
    content: "\e724";
}
.e-connector::before {
    content: "\e725";
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Restrict expansion of the palette panel

The symbol palette panel can be restricted from getting expanded. The `[cancel]` argument of the `[paletteExpanding]` property defines whether the palette's panel should be expanded or collapsed. By default, the panel is expanded. This restriction can be done for each of the palettes in the symbol palette as desired.

In the following code example the basic shapes palette is restricted from getting collapsed whereas the swimlane shapes palette can be expanded or collapsed.

INDEX.JS

```

/**
 * Default symbol palette sample
 */
//Initialize basic shapes palette
function getBasicShapes() {
    var basicShapes = [
        { id: 'Rectangle', shape: { type: 'Basic', shape: 'Rectangle' },
        style: { strokeWidth: 2 } },
        { id: 'Ellipse', shape: { type: 'Basic', shape: 'Ellipse' }, style:
        { strokeWidth: 2 } },
    ];
    return basicShapes;
}

```

```
//Initializes the symbol palette
var symbolPalette = new ej.diagrams.SymbolPalette({
  palettes: [
    {
      id: 'swimlaneShapes', expanded: true,
      title: 'Swimlane Shapes',
      symbols: [
        {
          id: 'stackCanvas1',
          shape: {
            type: 'SwimLane', lanes: [
              {
                id: 'lane1',
                style: { fill: '#f5f5f5' }, height: 60,
                width: 150,
                header: { width: 50, height: 50, style: {
                  fill: '#C7D4DF' } },
                orientation: 'Horizontal', isLane: true
              }
            ],
            height: 60,
            width: 140,
            style: { fill: '#f5f5f5' },
            offsetX: 70,
            offsetY: 30,
          }
        }
      ],
      { id: 'basic', expanded: true, symbols: getBasicShapes(), title:
        'Basic Shapes' }
    ],
    symbolHeight: 50, symbolWidth: 50,
    symbolPreview: { width: 100, height: 100 },
    expandMode: 'Multiple',
    height: '400px',
    width: '100%',
    paletteExpanding: function (args) {
      if (args.palette.id === 'basic') {
        // Basic shapes panel does not collapse
        args.cancel = true;
      }
      else {
        // Swimlane shapes panel collapse and expand
        args.cancel === false;
      }
    }
  }
});
symbolPalette.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>
<style>
    @font-face {
        font-family: 'e-ddb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+lDAAACIAAAAIJnbHlmw/g
RIAAAvGAAACw0aGVhZBGJTLcAAADQAAANmhoZWEIXQQAARAAACRobXR4oAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYwllldAwInAAALyWAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAFwEAAAAAAAAEAAABAAAAAAAAAAAAAAAAAKAABAAAAQAAJo24vV8
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAAEAQAAAAACAACAAAAAAAAAAEAAAAoAwABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAAQACAAMABAFAAYABwAIAAKACgALAAw
ADQAOAA8AEAAARABIAEwAUABUAFgAXABgAGQAaABsAHAAAdAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMiJaAnaClYLMauqC7gMpg2ODmQ
Owg8ad9IQoBF6ELyTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYffQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFbY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBP6XfoYegkICQcIBgYGBQQEAWMCAQEBAgMEBQUGBwgICQoKCwMDA0ODg4PDxAPER
RERESERSEBEQEBApDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQMDA0LDBYWFBIRDw4
LCgQDAwICAQECAgMDBAoLDg8RehQWFGwLDQwMDQ0NDA0MDAwLFxUUExEpDQwKawQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHBgYUFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAwQKCw4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
```

NDhoZGBP6XvoyEwkJCAgHBwYFBQUDAwMCAQICAwQFBQYHCAGJCgoLDAwMDQ4ODg8PDxAREBERERI
REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDAwNCwwWfHqSEQ8OCwoEawMCAgE
BAgIDAQKQCw4PERIUfHYMCw0MDA0NDQwNDawMCxcVFBMRDw0MCgMEawIBAQEBAQECAwQDCgWNDxE
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDawMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEpDy8Pay0JAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBARERESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAGGBwUFBAQDAgEBAGMEBAUFbWYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbB+ShQTExERDw4OCwsJBwYFAgEBAGUGBwkLCw0PBxAREhMUAcAAAAAABAAAAAAD9AO
1AAMABwAvADMAAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBoF7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9fT4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUeCgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQQKCwoJBwUCAgUHCQoLCgQAAAAAAGAAAAADtQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBwZ9Pv2QPn0GBwYLCggHBQEB+X58fwrCvP2OBgYGCwoJBgUCAQEcbQYJCgsGBgY
CcgYGBgsKCQYFAgF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAADAAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzmFDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCACIBgYGBAQEAgEBAQEcbAQEBgYGCACJCAk
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgcHCAGICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAgJCQsLCwwNExAPBgYFBQQDAwIBAQEcbACICgWNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICACHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITExQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCAg
JCQoLDAwMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+ix36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAGMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAAANxUfCSE/CTURITcjFSE1IzUjYAEBBQc
ICgsGBwYB9AYHBgsKCAcFAQH9kLv6Au76+okGBwYLCggHBQEBAQEBAQUHCAoLBgcGAj07fX0/AAA
AAQAAAAADdwN3ANEAAABMhJz8LowEfHR0BDx0jLw8jHx47AT8dPQEvHSMPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUBAACAGEBAGIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXFxYUFBMSEA8NDAsIB14EBAQFBgcHCAGJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTEhISExESERERE8QDg8NDXECpooJEQ8NBQUFAwQCAgEBAGIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQcJCwwODxESExU
VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEB
REhISEhMTEhMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwcBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAGIDAQwQFCg0OEBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgcJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEHMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAGIDAQwQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERERESExMUFBQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQADuICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEhIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEExAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUEXiQDgwLBAQEAgICAQEAgMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCgsNDQ4PEBAREhM
TExQVFRoaGhkZFXYWAVEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgcICQoMDA0PDw8RERI
SExQUFBVFBQCBzflBgSODxEsFBYWDawMDAwNDQwNDA0MCwwLFhUTERAODQoFBAMDAgEBAGICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVFBMTExIREQ8QDg0NDAoKCAcGBQQAQEEBQgKDA4Q/qseAgICAgQEBQUBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREpDw8NDQsLCQkHBgUDAwIEAAABQAAAAADvAO8AAMAIwArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIEDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ

GCAkKC6z+RAFZ3t6fBAUFBQYFBQQEAgICAgQEBQUGBQUFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wAcwAAASEVIREzESE1IREjAeT+YAGgOAG
g/mA4Ahw4/mABoDgBoAAEAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERIXEhETMRIxEhESM
nESMRfYE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCac
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVArX9Lm8CBAYICQoKAXYKCgkIBgQCAQMFBwgKCGAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUExMFBgcJCgoMDA40EBAREhITGRgWfXcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBWUfhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEawIBWyIDBQYICQsNDQ8RERMUFUXGBgZDRobG0cTExIQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAvoDhAAiAEUakAAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBgcICgoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA00DhAQVG/tDhsaGRgWFRQTCAGHBwYGBQQAEMCAQECAUGCAoKDA00Dw8REhIPDg4NDAs
KCQkHBgUEAwEBAGQGAoLDhAREhQVFxga9wHIAQIDBAUFBwCICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAd4BTgEBAGMDBAUGBwCJCQkLCwwPDQwMCwoJCQcHBQQAeAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwCICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABAYETJzMfDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAGHBgYLCgoVBQ0OEakKAXL+IAkJCAChBWUfLhkFCgkGBQIBAXXMAwICAQIFBgkKCwYGHAEBBQc
ICgsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwCQAwcGBAIBfQEBAwQFBgcI/tMCCzo
CBwkKCwYG/UoBmgcHBwCGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwCQAwcGBAIBAQIAAAAABgA
AAAADAQ08AAMABwALAB8AIwBeAAALMxEjAzMRIwMzESM1EQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwCICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBvW/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICACHBgYGBAQEAwIBAQIDBAQFBQYGBwC
ICAgJAn83PgSKAgGBAIBAAABAAAAA08A7wAXgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQKGDg4cGhoZFxcVFBMQEDfegQ00EBITFBWUGBgzGhsbGxwaGhoZGRcXFhUUFBIREA40DAoJCAY
FAGEBAGUGCAkKDA40EBESFBQVfHcXGQwaGRsdEBAQE8PDw8PDg4ODQ0MDAwLCwsKChIIBwCHBgU
ENgUGBwCICQkKCwsLDA0NDQ4PDg8QEBAEREREREhISEhITHh4dHRwbGhkZFxyUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhARExQWGBgad984GRcXFRQSEQ8ODAO
JBgUDAECEBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTEExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwCICQkKCgoMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKCQkIBwCHBgUEAwM
CQEDBACJCwwNDxEXESeXUWFxkZGhscHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODAOJBwQDAQMFAAAAGAA
AAAADFQ08AAMAAaAAANYE1IREfHjsBPx4RIxEpDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwCGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFHcWfHqVEExERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICACHBgYFBAQDAgI
BAQICAwQEBQYGBwCICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww
NDxERExQVFhYB/wAAAAEAAAAAARedvAADAALMwEjAU86ASg6RAN4AAADAAAAAAQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBawUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEExER/vUPDw8NDgwMDAsLCgkJCAChBWUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwC
ICQoJCwsMDAwODQ8PDxAQEBAERERASERARERAQEAJkZGRkZGQOCAkRERMTFRUWFRUVExMREREOQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTEExEREQ4NCwkHBQMBAQMFBwKLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQE8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAQMEBwgLDA6pBAMCAgI
BAGIDBwkKBQUMDQwFBQqqFhYXGBgZGRsRERAREBAQE8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAgI
DAwUFAAAAAA5ADkaADAEQAYwAAASE1ISufCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcv
vHisBDwUBOAEs/tQBIwCHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTEExIREA8MCwkHBQMBAQM
FBwkLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUdAwICAgIDAUFbWCHCAK
JCgsLCw0MDg0PDhAQEBAQERARERSZGRgYFxyWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4MCwgHBAM
JAQECAGQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEEREBEREBEBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBawUHCQsNDhERERMFRUFVhUVExMREREOQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxAQEBAEREBERehARERAQEBApDw8NDgwNCwsLCgkJCAChBWUFAwMCAgEDBACICww
OqQQDAgICAQICAwJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBApDw8NDgwNCwsLCgkJCAChBWU
FAwMCAgICAwMFBQAAAGAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARC7AR8KDxArAS8WPwg
nNw8BJyMfCRUFgJ8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAgEBAgICAwP8+gMDAg8
HOGUFBgkJAwQDAgULAQEDBAIFBwCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQAFBgYHCAGKCgsMDQ4PEBASEhMTFRULIhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIAwEBAGQBBiIKCwsMAGMKOCN1LM4CAwNJAwMCAgIBAQICAgMDSQM

DAgICAQECaPMBaGIFCAMJCw89fVYjHhgLDw8OEwwNDaGGBQYFAwECaWMEBQYECwYGBg8KDAwNDQ4
PEJKxIAGFaGIEAQIDJgcEAQYuAwMEBAQFBBEL4jgfgGhoODg0MDAsKCgkICQcIBgcFBQQEAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFBUBAQcCAGIQGwEFBQAEAAAAAQA5AAAwAjACc
ARQAAARUhNScfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgM
EBAUFBBQFAwQCAgICBAMFBAUFBBQBM/7UZDIyCQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAMDAQEBAQMDBAQEBQUFBAQDAgIBA+WWlpYBBQQFBgYHCAj+opaWAV4HCAsGBwUEAvo
AAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBACICQoL+QIEBwgJCgtjCgoJCACeAvkLCgkIBwQCAGQHCAkKC/kCBACICQoKXGsKCgOABQk
KCvoCBACICQoLYwoKCQgHBAL5CwoJCACeAgIEBwgJCgv5AgQHCakKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwgPCAAMABwAJAFUAmwAAARUhNQEVIZUHNSMVHw8hPw81FxEjNS8PIQ8PFMS
RNQ8PER8PIT8PETUvDzECyP5wASyWlmQBAQIEBAUGBgCICakJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgCICakJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgCICakJCgoKArwKCgoJCQgIBwYGBQQEAgEBAgIDBAQGBp8HBwcICAgJCgFqyMgB9MjIyMj
ICgoKCQkICACGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgCICakJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAGHBgYFBQAQAEBAQIEBAU
GBgcICakJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAGAAAAACAAAAAQA5AABQCxAAABHwQPC8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAgMEuAUFBgGAWgFAwMCAGe
jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDChMRDQsIAwMBAQECAwIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAgIDAwUFBwCGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECAwUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEBlMBAgQFBAMEBQUGBgcICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgwNDawbHRSOHw8
PDQ0NDA0MDAsJCQgHBgYEawIBUwUFBQQDBAMCAgEBAgMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAgMFBggJCgsMDQ0NDg8PaFQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAAAADAdbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAKHCAUEBAMCAgEBAgQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUFCABBBQiHhEQ2Q+iaioZEwGAQECBQCBQMDAwUaRQHxyRcXFhUWFRUUEX
QBw4MCwkDBAICAgEBAwQGBwKLDQ0PEBAREhMTDScTFQkIBgYFBQQEawEBAQMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwKJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBAwUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBAwEUAWQBazUGKwQEBAMEAgILVv4rIR4ICAC
BCA0xCwICAgMEBgCICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCACGBQYUCw8IBwcICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQDDAgEBAwAAAAAAMAAAAA/QDcAAqAFYAUAQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEVDiMPBgOVbwUFBAMCAgEBAwSaCAG
KDAsMCww9wAYFAwMDAQIDBJoICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhGIEBQcJCQlNCAgFBQQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwwM/uMCAGQFBgcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQBAGMDAwUEBQYGVggHBwUFBaIBAU8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQECBACJCgw
MxQgIBwgICAgICQkJCQYKQgHBAQBVAwMCwoKCgkJCACGBQDDAQEQDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBgcAAAAABQAAAAADXgOQACEAQwBlAgkAxQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUFBAQ
DAgEBAQECAwQEBQUFBQQEawIBfAEBAgMEBAUFBUQEBAMCAQEBAQIDBAQFBQUFBAQDAgF8AQECAwQ
EBQUFBQQEawIBAQBAGMEBAUFBUQEBAMCAbAU1xRCIn0FBQQEawIBAQBAGMEBAUFQGIBAwMEBAU
FBgYHBwcHCAHCCAChBwcGBgUFBQAQDAwECGQUFBAQDAgEBAQECAwQEBQWWIGQFBwCICakKvwkKCAg
HBwUCcP68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQ
EBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEZzI
yJFYBAQIDBAQFBRkFBQQEawIBaf3zCACHBwcGBgUFBQAQDAwECAGEDAwQEBQUGBgcHBwCIAg0BAQI
DBAQFBRkFBQQEawIBAVYICACFBQMCAQECAwUFBwgAAAAAQA5AAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHsBDwUBbIRERAPEA4OSAQBUEBQOEBAMCAgE
BAgMEBQYGBuoFBQYEBAMDBAEBAQECA0sTFBUXBgZGQ0ODQ0NDA0MGAsLCwoJCQkIBwgHBgYKBMQ
DAwEBAQEBAQMDAwUKBgYHCAcJCQkJCgsLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgsMAGQDBAQ
DAkgDAQMPDxARERMTFBQUFRUwFhYwFBQUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAQM
DBAYGCAgJCgoLDAwNDg4PEBAREhITEhMUEXQUBMTEhITEhIDcwcJCQoKCw0MRgMCAGEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQEBUSRDgwKCAyEAQEBAQIDBAQFDAYHBwgJCAkKCgsKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBgHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgsMDA0ODg0QEBESEhITEhQTFBQUFBQUExQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAMKA48AKAAATMfBBUHCwEPBjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBATjBgSNGR8gJAN

```

aAwQDAwMNF/7x/soPDAoHBRItCgEGBAIbGBAPLwGZiiEKBB0YFggBBwAABAAAAAAEAAQAAAMABwA
LACMAAAEVITUhFSE1ARUhNQmZFSERiXehESM1IRUjESERiXehNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAUAAAAAAQAAAAEAAQ
AAHYAAAHEHIREhLwcPDx8PPw8hETMfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAChBQQ
DAgEBAgMEBQChCAoJCwsMDA0NDQ0MDAsLCQoIBwcFBAMCAQFAwAECawQFBwcICgkLCwwMDQ0NDQw
MCwsJCgghBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBAgMEBQChCAoJCwsMDA0NAwANDQwMCwsJCgghBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBawYHCQoMAAAAAQAAAAAA/8EAAWAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
HCAoKDAwNDw8PEjP92QEcAkABBAUICQsNDxAREhQUFhYXfXyVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVFhcXFhYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwcFBQUEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQBAQQFCAkLDQ8QERIUFBYWFxcWFhQUEhE
QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBgcICQoLDAwNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwcHCAkJCQoLCwsMDawNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAEAAABAAAAAABABsAAQABAAAAAAACAAcAHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJ1aWxkZXJSZWdlbGFiTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lubiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ
1aWxkZXJGbz250IGdlbmVvYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGWAXwBEAGkAYQB
nAHIAIAYQBtAEIAdQBpAGWAZABlAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGWAXwBEAGkAYQBnAH
IAYQBtAEIAdQBpAGWAZABlAHIAVgBlAHIAcWBPAG8AbgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQB
yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAABnAGUAbgBlAH
IAYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAoAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAAB
BEQESARMBFAEVARYBfWfYARkBGGgEbARwBHQEeAR8BIAEhASIBIwEkASUBJgEnASgBKQAHWm9vbU
uTQhab29tT3V0TQpVbmRlcXpmbmVNB1Byaw50TQROZXNdNBVNhdmVNB0V4cG9ydE0FQm9sZE0LT3B
lbkZvbGRlck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pvb21JbkYlWm9vbU9ldEYGUHJpbmR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVGCFJlZnJlc2hGClV
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFpvb21PdXRCClVuZGVybgGluZUIGUHJpbmRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXBlcwldDb25uZWN0b3ILQmFzaWNTaGFwZXMAAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-ddb-icons {
font-family: 'e-ddb-icons';
speak: none;
font-size: 16px;
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-basic::before {
content: "\e726";
}
.e-flow::before {
content: "\e724";
}

```



```

    }
    .e-connector::before {
        content: "\e725";
    }
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Stretch the symbols into the palette

The [fit](#) property defines whether the symbol has to be fit inside the size, that is defined by the symbol palette. For example, when you resize the rectangle in the symbol, ratio of the rectangle size has to be maintained rather changing into square shape. The following code example illustrates how to customize the symbol size.

INDEX.JS

```

/**
 * Default symbol palette sample
 */
//Initialize the basicshapes for the symbol palatte
var basicShapes = [{
    id: 'Rectangle',
    shape: {
        type: 'Basic',
        shape: 'Rectangle'
    }
},
{
    id: 'Ellipse',
    shape: {
        type: 'Basic',
        shape: 'Ellipse'
    }
},
{
    id: 'Hexagon',
    shape: {
        type: 'Basic',
        shape: 'Hexagon'
    }
}
];
//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [{
        id: 'basic',
        expanded: true,
        symbols: basicShapes,
        title: 'Basic Shapes',

```

```

        iconCss: 'e-ddb-icons e-basic'
    }],
    symbolHeight: 80,
    symbolWidth: 80,
    getSymbolInfo: function(symbol) {
        // Enables to fit the content into the specified palette item size
        return {
            fit: true
        };
        // When it is set as false, the element is rendered with actual node
        size
    },
    });
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>
<style>
  @font-face {
    font-family: 'e-ddb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+lDAAACIAAAAIJnbHlmw/g
RIAAAvGAAACw0aGVhZBGJTLcAAADQAAANmhoZWEIXQpAAAArAAAACRobXR4oAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYW1ldAwInAAALyAAAMVcG9zdNAiwIs
AADJEAAABuQABAAAEAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAAAAKAABAAAAAQAAJo24vV8

```

PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAEEEEAAAAoAOwABgAAAA
AAgAAAAoACgAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAAB
AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAADAAEAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAAQAEAAAAQACAAMABAAFAAYABwAIAAkACgALAAw
ADQAOAA8AEAAARABIAEwAUABUAFgAXABgAGQAaABsAHAAADAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMIIJaAnaClYLMauQC7gMpg2ODmQ
Owg8aD9IQoBF6ElYTRhRGFIQUwBVMFhoAAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBP6XfoYegkICQcIBgYGBQQEAwMCAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPER
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBA0LDg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUExEPDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHBgYUFBAMCAgECawMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4Phu0BOP7IAZYBAgIDAQKcW4PERIUfHYMCw0
MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDawNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBP6XvOYewkJCAgHBWYFBQUdAwMCAQICAwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAREBERERI
REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBA1c/Hw0NDawNCwwWFhQSEQ8OCwoEAWMCAgE
BAgIDAQKcW4PERIUfHYMCw0MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxET
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDawMCwoKCQgIBWYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEPDy8PayOJAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDawLcwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBAERERESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDAsKCgkJCAgGBWYFBAQDAgEBAgMEBAUFBWYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTExERDw4OCwsJBWYFagEBAgUGBwKLCw0PBxAREhMUACAAAAAABAAAAAD9AO
1AAMABwAvADMAAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBWYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBWYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBoF7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9fT4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQKQKwoJBWUCAgUHCQoLCgQAAAAAagAAAAADtQP0AdcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBWYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBWz9Pv2QPn0GBWYLcggHBQEB+X58frwCvP2OBgYGCwoJBgUCAQEBCQYJCgsGBgY
CcgYGBgsKCQYFAgF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAAADAAAAAMoA3cAIGBFAIUAAAEfDw8
OKwE1EzMFDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCAcIBgYGBAQEAgEBAQECEBAQEByGCACJCAk
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgcHCAgICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAGJCQsLCwwNExAPBgYFBQQDAwIBAQEBCAcICgWNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICAcHBgYUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBWgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAACAAAAAAdP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCAG
JCQoLDawMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBgICQkKCwwMDA3+iX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBWYFBQMCAgMFBQYHCAkKCgsMDAwNABUNDAMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAagAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBWYB9AYHBgsKCAcFAQH9kLv6Au76+okGBWYLcggHBQEBQEBAQUHCAoLBgcGAj07fX0/AAA
AAQAAAAADdwN3ANEAAABMhJz8LOWEfHR0BDx0jLw8jHx47AT8dPQEvHSMPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUFBAMCAgEBAgIDBAUFBQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUFBMSEA8NDAsIB14EBAQFBgcHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEXiSEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTEXiSExESERERE8QDg8NDXECPOOJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBWYFBQUEAwICAQEDBQcJCwwODxESExU

VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTEhMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwCGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8daQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAuQFCg00EBITFRYLDaw
MDAwNDQ0MDQwMDAwLfhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAOQDQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBGcJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAuQFCwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLfhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFBQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEhIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQO8AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjSbPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUEXIQDgwLBAQEAgICAQECAGMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCGsNDQ4PEBAREhM
TEhXQVFRoaGhkZFxYWAWEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBGcICQoMDA0PDw8RERI
SEhXQVFBVFBQCbzflBgSODxESFBYWDawMDAwNDQwNDaw0MCwwLfhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMfHUUehEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUEhXQVFBVFBMTExIREQ8QDg0NDAoKCAcGBQQCAQEEBQgKDA4Q/qseAgICAgQEBQUBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAO8AAMAIwArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFBQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFBQYFBQQEAgICAgQEBQUGBQUFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYyE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVarX9Lm8CBAYICQoKAxYKCGkIBgQCAQMFBwgKCGAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUExMFBGcJCgoMDA4OEBAREhITGRgWfxcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCQFBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEAWIBWYIDBQYICQsNDQ8RERMUFUXGBgZDRobG0cTEhIQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUAKAAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBGcICGoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA00DhAQVG/tDhsaGRgWFRQTCAGHBWYGBQQEAWMCAQECAUGCAoKDA00Dw8REhIPDg4NDAs
KCQkHBgUEAwEBAGQGCALDhAREhQVFxga9wHIAQIDBAUFBwcICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAD4BTgEBAGMBAUGBwcJCQkLCwwPDQwMCwoJCQcHBQQEAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwcICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABayETJzMFdCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAghBgYLCgoVBQ00EakKAXL+IAkJCAChBWUf1hkFCgkGBQIBAxXMAwICAQIFBgkKCwYghAEBBQc
ICGsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwcQAwcGBAIBfQEBAwQFBGcI/tMCCzo
CBwkKCwYG/UoBmgCHBwcGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwcQAwcGBAIBAQIAAAAAAGA
AAAADAQO8AAMABwALAB8AIwBeAAAlMxEjAzMRIwMzESM1EQ8HIS8GNRE1FSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwcICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBwV/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBgJCgo+N/2BCQgICACHBgYGBAQEAWIBAQIDBAQFBQYGBwc
ICAgJAn83PgSKAgGBAIBAAABAAAAA08A7wAXgAAAQ8MNSMVMzUjPw8ffw8XLx4HHx4zPxcvFyM
PAQgKDg4cGhoZFxcVFBMQEDfegQ00EBITFBWUGBgZGhsbGxwaGhoZGRcXFhUUFBIREA4ODA0JCAY
FAGEBAGUGCAKDA4OEBESFBQVfHcXGQwaGRsDEBAQEAsPDw8PDg4ODQ0MDAwLCwsKChIIBwCHBAM
ENgUGBwcICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhAREXQWBGad984GRcXFRQSEQ8ODA0
JBgUDAQECBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTEhEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwgICQkKCgoMCwwMGg4ODg8PDw8OEHIREBEQDw8PDg4NDQwLCwsKCQkIBwCHBQUEAwM
CAQEDBACJCwwNDxESExUWFxkZGhsCHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODA0JBwQDAQMFAAAAAGA
AAAADFQO8AAMAAaAANyE1IREfHjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwcGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhCWFhQVEERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICACHBgYFBAQDAgI
BAQICAwQEBQYGBwcICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww

NDxERExQVFhYB/wAAAAEAAAAAArEDvAADAAALMwEjAU86ASg6RAN4AAADAAAAAAOQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBaWUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEExER/vUPDw8NDgwMDAsLCgkJCAcHBwUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwc
ICQoJCwsMDAwODQ8PDxAQEBAERERASERARERAQEAJkZGRkZGQOCaKkRERMTFRUWFRUVExMREREOQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTEExEREQ4NCwkHBQMBAQMFBwKLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQE8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAQMEBwGLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqgFhYXGBgZGRsRERAREBAQE8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAQI
DAwUFAAAAAAA5ADkAADAEQAYwAAASE1ISufCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPw
vHisBDwUBOAEs/tQBIwcHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTEExIREA8MCwkHBQMBAQM
FBwKLD8QERITEExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUdAwICAQIDAUFwCHCAK
JCgsLCw0MDg0PDhAQEBAQERARERsZGRgYFXYWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4KDCwgHBAM
BAQECawQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBaWUHCQsNDhERERMtFRUVFhUVExMREREOQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxAQEBAEREBERehARERAQEBApDw8NDgwNCwsLCgkJCAcHBwUFAwMCAgEDBACICww
OqQQDAgICAQICAwCJCgUFDA0MBQUKqHYWfXgYGRkbEREQERAQEBApDw8NDgwNCwsLCgkJCAcHBwU
FAwMCAgICAwMFBQAAAgAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARc7AR8KDxArAS8WPwg
nNw8BJyMfCRUFgJ8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAGICAwP8+gMDAg8
HOGUFBGkJAwQDAgULAQEDBAIFBwCLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQAFBgYHCAgKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBIKCwsMAGMKOCN1LM4CAwNJAwmCAgIBAQICAgMDSQM
DAgICAQECaPMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJKxIAgFAgIEAQIDJgcEAQYuAwMEBAQFBBEL4jgGfGhoODg0MDAsKCgkICQcIBGcFBQQAeAgIBAQE
EAgMEBAkKBGcHBw8QEBENDxoYESUqMLYYFRAFBQUBAQcAgIQGwEFBQAEAAAAAAOQA5AAAwAjACc
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgM
EBAUFBBQFAwQCAgICBAMFBAUFBBQBM/7UZDIyCQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAQMDAQEBQUBAQDAgIBA+WWlpYBBQFBBgYHCAj+opaWAV4HCAsGBwUEAvO
AAAAEAAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBAICQoL+QIEBwgJCgtjCgoJCAcEAvkLCgkIBwQCAgQHCakKC/kCBACICQoKXgsKCgOABQK
KCvoCBACICQoLYwoKCQgHBAL5CwoJCAcEAgIEBwgJCgv5AgQHCakKC2MKCGkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwGPCAAMABwAJAFUAmwAAARUHNQEVIzUHNsmVHw8hPw81FxEjNS8PIQ8PFsm
RNQ8PER8PIT8PETUvDzECyP5wASyWlMQBAQIEBAUGBgICakJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgICakJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKArwKCgoJCQgIBwYGBQQEAgEBAGIDBAQGBp8HBwCICAgJCgFqyMgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAGHBgYFBQAQAEBAQIEBAU
GBgCICakJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAAOQA5AAbQCxAAABHwQPCc8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAGMEuAUFBgGAWgFAwMCAgE
jHxsYCwoJCQgIBGcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDCMRDQsIAwMBAQECAwIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAgIDAUFwBwGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECAwUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEB1MBAgQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAg8cHBsaGgwNDawbHRSOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAgEBAGMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAGMFBggJCgsMDQ0NDg8PafQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAgEBAGQFBwgKDA0OEBITFRYIERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBGcJCgwODg4QEBIUFCAZBBQiHhEQ2Q+iAioZEwKQAECEBQCBQMDAwUaRQHxyRcXfHwUFRUUEX
QBw4MCwkDBAICAgEBAGwGBwKLDQ0PEBAREHMTDSCTFQkIBgYFBQQAeAwEBAQMEBggJCwsNDQ8PERA
RERIREkECBwMFawMEBQYGBwKJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAGEBawUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBAwEUawQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgICGoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCAcGBQYUCw8IBwCICQoLDAw
MDhMSEhAQDg0MCgoJCAcGBQQDAgEBAAAAAAMAAAAA/QDcAAqAFYAUQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVBwUFBAMCAgEBawSaCag
KDAsMCwv9wAYFAwMDAQIDBJoICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCAcGAWL+bhISEhM
SEA4NhgIEBQcJCQ1NCAgFBQQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwwM/uMCAgQFBGcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM

DAwUEBQYFvvggHBwYFBAIBAQEBAgMDAwUEBQYGVvggHBwUFBAIBAU8CBAUICakJLAoJCAcGawICBAU
ICakJWQEEBgkKCwNpQHECQkJBwUEAiAJCQoKCgSMDP4KDAwLCgoKCQkIBwYFBAMBAQEBCaJCgw
MxQgIBwgICAgICQkJCQYKCQgHBAQBVAwMCwoKCgkJCACGBQQDAQEEDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBgcAAAAABQAAAAADXgOQACEAQwBlAGkAxQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNychIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUFBAQ
DAgEBAQECAwQEBQUFBQQAeAwIBfAEBAgMEBAUFQBUEBAMCAQEBAQIDBAQFBQUFBAQDAgF8AQECAwQ
EBQUFBQQAeAwIBAQEBAgMEBAUFQBUEBAMCAbAU1xRCIn0FBQQAeAwIBAQEBAgMEBAUFQGIBAwMEBAU
FBgYHBwcHCAHCCAChBwcGBgUFBAQDAwECGQUFBQDAgEBAQECAwQEBQWWIGQFBwCICakKvwkKCAg
HBwUCcP68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQ
EBgFEBgQEBAMDAQEBAQMDBAQEBv68BgQEBAMDAQEBAQMDBAQEBgFEBgQEBAMDAQEBAQMDBAQEBzI
yJFYBAQIDBAQFBkFBQQAeAwIBaf3zCACHBwcGBgUFBAQDAwECAGEDAwQEBQUGBgCHBwCIAg0BAQI
DBAQFBkFBQQAeAwIBAVYICACFBQMCAQECAwUFBwgAAAAAQAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHisBDwUBbBIRERAPEA4OSAQFBAUEBAQoEBAMCAgE
BAgMEBQYGBuoFBQQAEBAMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBwgHBgYKBQM
DAwEBAQEBAQMDAwUKBgYHCAcJCQkJCgSLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgSMagQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWfHYWFBQUEXQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAQM
DBAYGCAGJCgoLDAwNDg4PEBAREhITEhMUEXQUBMTExITEhIDcwcJCQoKCw0MRgMCAgEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQAeBUsRDgwkCAYEAQEBAQIDBAQFDAYHBwgJCAkKCgSKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBgHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgSMDA0ODg8QEBESEhITEXQTFBQUFBQUEXQ
TEXISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAATMfBBUHCwEPBjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBAtjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRITCgEGBAIBGBAPLwGZiiEKBB0YFggBBwAABAAAAAAEAAQAAAMABwA
LACMAAAEVITUhFSE1ARUhNQmzfSERIXEhESM1IRUjESERIXEhNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAAUAAAAAAAAQAAAAEAAQ
AAHYAAAEHIREhLwcPDx8PPw8hETmfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAChBQ
DAgEBAgMEBQCHCAoJCwsMDA0NDQ0MDAsLCQoIBwCFBAMCAQFAwAECawQFBwCICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQSLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQSLDAwNDQ0NDawLCwkKCAChBQDAgEBAgMEBQCHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQSLDAwNDQ0NDawLCwkKCAChBQDAgEBAwYHCQoMAAAAAAQAADjwOPAOGAAAEpBy8DKwEPBx0
ABQCrAAABHDwEVHxAFAQUDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwEBAQIDBQY
HCAoKDAwNDw8PEjP92QECaKABBAUICsNDxAREhQUFhYXfXyVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVFhCXfHYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwCFBQQAeAwMBAgECAGMDBAKMDQ8REXQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLfhYUFBIREA8NCwkIBQQAQQFCakLDQ8QERIUFBYWfXcWfHQUEhE
QDw0LCQgFBAEBBAUICsNDxAREhQUFhYCCf7+AwQFBgCICQoLDAwNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwCHCAkJCQoLCwsMDAwNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAAEAAABAAAAAABABsAAQABAAAAAAACAAcAHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJlYWxkZXJSZSWd1bGFyTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lubiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJl
YXNkZXJGbz250IGdlbmVyYXRlZCBlc2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHcAIABNAGEAdABLAHIAaQBhAGwAXwBEAGkAYQB
nAHIAAYQBTAEIAdQBpAGwAZABLAHIAATgBlAHcAIABNAGEAdABLAHIAaQBhAGwAXwBEAGkAYQBnAHIA
YQBTAEIAdQBpAGwAZABLAHIAVgBlAHIAcWBPAG8ABgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQBy
YAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAABnAGUAbgB8AIA
YQBT0AGUAZAAGAHUAcwBPAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIAABnAGUAbgB8AIAIA
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAACgBAgEDAQBBQEGAQCBCAEJAQoBCwEMAQ0BDgEPARA
BEQESARMBFAEVARYBFwEYARKBGgEbARwBHQEeAR8BIAEHASIBIwEkASUBJgEnASgBKQAHWm9vbU1
uTQhab29tT3V0TQpVbmRlcmxpbmVNB1Byaw50TQROZXdnbnVhdmVNB0V4cG9yde0FQm9sZE0LT3B
1bkZvbGR1ck0HRGVsZXR1TQhSZWZyZXNoTQdJdGfSaWNNB1pvb21JbkYIwM9vbU91deYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkRgtPcGVuRm9sZGVyRgdEZWxldeGVGCFJlZnJlc2hGClV
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFPvb21PdXRCClVuZGVybgGluZUIGUHJpbnRCBE5ld0I

```

FU2F2ZUIHRXhwb3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxlIdGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXBlcwldDb25uZWN0b3ILQmFzaWNTaGFwZXMAAAAAA==) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-ddb-icons {
    font-family: 'e-ddb-icons';
    speak: none;
    font-size: 16px;
    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-basic::before {
    content: "\e726";
}
.e-flow::before {
    content: "\e724";
}
.e-connector::before {
    content: "\e725";
}
}
</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/Remove symbols to palette at runtime

- Symbols can be added to palette at runtime by using public method, [addPaletteltem](#).
- Symbols can be removed from palette at runtime by using public method, [removePaletteltem](#).

Customize the size of symbols

The size of the individual symbol can be customized. The [symbolWidth](#) and [symbolHeight](#) properties of node enables you to define the size of the symbols. The following code example illustrates how to change the size of a symbol.

INDEX.JS

```

/**
 * Default symbol palette sample
 */
//Initialize the basicshapes for the symbol palatte
var basicShapes = [{
    id: 'Rectangle',
    shape: {

```



```

        type: 'Basic',
        shape: 'Rectangle'
    },
    {
        id: 'Ellipse',
        shape: {
            type: 'Basic',
            shape: 'Ellipse'
        }
    },
    {
        id: 'Hexagon',
        shape: {
            type: 'Basic',
            shape: 'Hexagon'
        }
    }
];
//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [{
        id: 'basic',
        expanded: true,
        symbols: basicShapes,
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic'
    }],
    symbolHeight: 80,
    symbolWidth: 80,
    symbolMargin: {
        left: 15,
        right: 15,
        top: 15,
        bottom: 15
    }
});
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>
<style>
    @font-face {
        font-family: 'e-ddb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+1DAAACIAAAAIJnbHlmw/g
RIAAAavgAACw0aGVhZBGJTLcAAADQAAAAANmhoZWEIXQpAAAArAAAACRobXR4oAAAAAAYAAAAAC
gbG9jYdYyee4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYW1ldAwInAAALyWAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAAAAKAABAAAAQAAJo24vV8
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAAAAAAEAAAAoAOWABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAAQACAAMABAAFAAYABwAIAAAkACgALAaw
ADQAOAA8AEAARABIAEwAUABUAFgAXABgAGQAaABSAHAAdAB4AHwAgACEAIgAjACQAjQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMIIJaAnaClYLMauC7gMpg2ODmQ
Owg8ad9IQoBF6ELyTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYffQUVHx07AT8LFxUXNycjJz8ONSfDx4Ban19P319PwEZAQICAwMECQwNEBESFbY
WDASMDQwNDQwNDQwMDASXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDASmfhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBp6XfoYegkICQcIBgYGBQQEAwMCAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPER
RERESERESEBEQEBApDw4ODQ0NDASLCgoJCAgHbGyYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBAoLDg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUExEPDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHbGyYFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAwwKcW4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEAWMCAGH9rgEBAGQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBp6XvOYEWkJCAgHBwYFBQUDAwMCAQICAwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAREBERERI
REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDAwNCwwWFhQSEQ8OCwoEAWMCAG
EBAgIDAwwKcW4PERIUfHYMCw0MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDx
ETFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAwMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDASLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEPDy8PayOJAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkJCAGBgQFAwMCAXwCAwUHCAoLDQ4OEBARERESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDASKCgkJCAgGBwUFBAQDAgEBAgMEBAUFBwYICakJCgoLDASNDA0ODg4
ODw8QDxAQAbb+ShQTEExERDw4OCwsJBwYFAgEBAgUGBwkLCw0PBxAREhMUAcAAAAAABAAAAAAD9AO
```

1AAMABwAvADMAAAEVITU1FSM1IREzFSE1MxEvDyEPDjchNSECvP6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNdf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECaWQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBof7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRFzMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9fT4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUeCgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQQKCwoJBwUCAgUHCQoLCgQAAAAAAGAAAAADTQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBwZ9Pv2QPN0GBwYLCggHBQEB+X58fwrCvP20BgYGCwoJBgUCAQECBQYJCgsGBgY
CcgYGBgsKCQYFAgF9/gwB9H0BAGUGCQoLBgZ2/ooBdrwAAAAADAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMfDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCACIBgYGBAQEAQEBQECBAQEBgYGCACJCAK
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgHCAGICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAgJCQsLCwwNExAPBgYFBQQDAwIBAQECBACICgwnDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICAKKCQoJCQkICACHBgUFBAMCArwoAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEExQVDw8ODg4NDQwLCwsJCQgIBg8PeggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQN3/RJ9AQIDBAUGCag
JCQoLDAwMDQLuDQwMDAsKCQkICAYFBAMCAQECaWQFBggICQkKCwwMDA3+iX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAGMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAGAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCAcFAQH9kLv6Au76+okGBwYLCggHBQEBQEBQUHCAoLBgcGAj07fX0/AAA
AAQAAAAADdwN3ANEAAABMhJz8LowEfHR0BDx0jLw8jHx47AT8dPQEvHSMPPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUBFAMCAgEBAgIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgFXfYUFBMSEA8NDAsIB14EBAQFBgcHCAGJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDawNDg4PDxAQEBE
SEhISExMTEExISExESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQcJCwwODxESExU
VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTEExMTEExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAo
8AAsAbADWAAABIXuzFTM1MzUjNSM3Hw8daQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwcBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAwQFCg0OEBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgcJCQoMDA4ODxARERISFBMVFBVUBVFBQCPzhvbzhvWwU
GDA4QEHMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAwQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFbQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAwMDA0MDQwNDQwMDAwMCxYUEXiQDgWLBQEAQICAgQECAGMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCgsNDQ4PEBAREhM
TEExQVFRoaGhkZFxYWAVEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgcICQoMDA0PDw8RERI
SExQUFBVUBVFBQCbzfLBgsODxESFBYWDawMDAwNDQwNDawMDA0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAwMDA0MDA0NDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAGMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVUBVFBMTEExIREQ8QDg0NDAoKCAcGBQQAQEEBQgKDA4Q/qSEAgICAgQEBQUBBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAo8AAMAIwArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQEAgICAgQEBQUGBQFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCakKC6z+RAFZ3t6fBAUFbQYFBQQEAgICAgQEBQUGBQFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERIXEhETMRIxEhESM
nESMRfYE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCAkKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVArX9Lm8CBAYICQoKAxYKCgkIBgQCAQMFBwgKCgAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRaKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbgJCgoMDA4OEBAREhITGRgWfxcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCfBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCAGIEBAYGBggICQoLCwwUFBMTEExE
REQ8PDg0MCwkJCgcEAWIBWyIDBQYICQsNDQ8RERMUFUXGBgZDRobG0cTExiQEA4NDAoJCAyFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUAKAAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4

Copyright © 2001 -2024 Syncfusion Inc.

RNQ8PER8PIT8PETUvDzECyP5wASyWlMQBAQIEBAUGBgICAKJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgICAKJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICAKJCgoKArwKCgoJCQgIBwYGBQQEAgEBAGIDBAQGBp8HBwcICAGJCgFqyMgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICAKJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAGHBgYFBAMDAQEBAQIEBAU
GBgcICAKJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAQA5AAbQCxAAABHwQPCC8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAGMEuAUFBgGAWgFAwMCAgE
jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDChMRDQsIAwMBAQECawIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAGIDAwUFBwCGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECawUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEB1MBAgQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgwNDawbHRSOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQDDBAMCAgEBAGMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAGMFBggJCgsMDQ0NDg8PafQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAgEBAGQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUfCAZBBQiHhEQ2Q+iaiozEwGAQECCBQCBQMDAwUaRQHxyRcXfHwFRUUExE
QBw4MCwkDBAICAgEBAwQGBwkLDQ0PEBAREhMTDScTFQkIBgYFBQQAawEBAQMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBawUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBawEUawQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgICGoMDQcPERMUCwsMDAwZExMREBAPdg4MCwsJCACGBQYUCw8IBwcICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQDQAgEBAAAAAAMAAAAA/QDcAAqAFYAUQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbWUFBAMCAgEBawSaCAG
KDAsMCww9wAYFAwMDAQIDBJoICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhgIEBQcJCQlNCAgFBQQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwwM/uMCAgQFBgcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQEBAgMDAwUEBQYGVggHBwUFBaIBAU8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQECBacJCgw
MxQgIBwgICAgICQkJCQYKQgHBAQBVawMCwoKCgkJCACGBQDQAEQDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBgcAAAAABQAAAAADXgOQACEAQwBlAGkAxQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUBAQ
DAgEBAQECAwQEBQUBFQQAeAwIBfAEBAgMEBAUFbQUEBAMCAQEBAQIDBAQFBQUBAQDAgF8AQECAwQ
EBQUBFQQAeAwIBAQEBAgMEBAUFbQUEBAMCAbAU1xRCIn0FBQQAeAwIBAQEBAgMEBAUFbQUBAQ
FBgYHBwCHCAHCCAChBwCGBgUFBQAQDAwECGQUFBQAQDAgEBAQECAwQEBQWWIgQFBwCICakKvwkKCAg
HBwUCcP68BgQEBAQDAQEBAQMDBAQEBgFEBgQEBAQDAQEBAQMDBAQEBv68BgQEBAQDAQEBAQMDBAQ
EBgFEBgQEBAQDAQEBAQMDBAQEBv68BgQEBAQDAQEBAQMDBAQEBgFEBgQEBAQDAQEBAQMDBAQEBzZ
yJFYBAQIDBAQFBRkFBQQAeAwIBaf3zCACHwCGBgUFBQAQDAwECAGEDAwQEBQUGBgCHBwCIAg0BAQI
DBAQFBRkFBQQAeAwIBAVYICACFBQMCAQECawUFBwgAAAAAAQAAAAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHISBDwUBbBIRERAPEA4OSAQFBAUEBQoEBAMCAgE
BAGMEBQYGBuoFBQQAEBAMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBwgHBgYKBQM
DAwEBAQEBAQMDAwUKBgYHCAcJCQkJCgsLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgsMAgQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWfHYWFBQUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAQM
DBAYGCAgJCgoLDAwNDg4PEBAREhITEhMUExQUFBMTExITEhIDcwcJCQoKCw0MRgMCAgEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQAEBUSRDgwKCAyEAQEBAQIDBAQFDAYHBwgJCAkKCgsKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBggHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgsMDA0ODg8QEBESEhITEhXQTFBQUBQUExQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAAATMfBBUHCwEPbjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBatjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRIItCgEGBAIBGBAPLwGZiIEKBB0YFggBBwAABAAAAAAEAQAAMABwA
LACMAAAEVIUhfSE1ARUhNQmZFSERiXehESM1IRUjESERiXehNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFaaQCAAUAAAAAAQAAAAAAEAQ
AAHYAAAAEHIREhLwcPDx8PPw8hETmfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAChBQ
DAgEBAGMEBQCHCAoJCwsMDA0NDQ0MDAsLCQoIBwCFBAMCAQFAwAECawQFBwCICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDawLCwkKCAChBQDQAgEBAGMEBQCHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAChBQDQAgEBawYHCQoMAAAAAAQAAAAA/8EAAWAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
HCAoKDAwNDw8PEjP92QEcAkABBAUICQsNDxAREhQUFhYXfYVFRQSERAPDQsJCAUEAQEEBQgJCw0

```
PEEESFBUVFcXfHfYUFBIREA8NCwkIBQT/FxESESEPEEA4ODQ0LCwsJC1uMtEDS0gMARxUSDW4PDg4
NDg0NDawMCwsKCwkJCQgHBwcFBQUEAwMBAgECAgMDBAkMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQQAQQAQFCAkLDQ8QERIUFBYWFxcWFhQUEhE
QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBgcICQoLDaWNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwcHCAkJCQoLCwsMDaWNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCgDawIBQgAAAAA
AABIA3gABAAAAAAAAAAAAEAAAAABAAAAAABABsAAQABAAAAAAACAAcAHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAAFaAsAWQABAAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBawADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYXwxfRG1hZ3JhbUJlYWxkZXJSZWdlbGZyTmV3IE1hdGVyaWFSX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFSX0RpYWdyYW1CdWlsZGVyVmVyc2lubiAxLjBOZXcgTWF0ZXJpYXwxfRG1hZ3JhbUJ
1aWxkZXJGb250IGdlbmVvYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
aQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHcAIABNAGEAdABlAHIAaQbHAGwAXwBEAGkAYQB
nAHIAyQBtAEIAdQBpAGwAZABlAHIAATgBlAHcAIABNAGEAdABlAHIAaQbHAGwAXwBEAGkAYQBnAH
IAyQBtAEIAdQBpAGwAZABlAHIAVgBlAHIAcWBPAG8AbgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQB
yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAbnAGUAbgBlAH
IAyQB0AGUAZAAGAHUAcwBPAG4AZwAgAFMAeQBuaGMAZgBlAHMAaQbVAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgBlAHMAaQbVAG4ALgBjAG8AbQAAAAACAAAAA
AAaOAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgBAGEDAQQBBQEGAQCBCAEJAQoBCwEMAQ0BDgEPARA
BEQESARMBFAEVARYBFwEYARkBGGEBARwBHQEeAR8BIAEhASIBIWEkASUBJgEnASgBKQAHWm9vbU
1uTQhab29tT3V0TQpVbmRlcmxpbmVNB1ByaW50TQROZXdnbnVndmVNB0V4cG9ydE0FQm9sZE0LT3B
lbkZvbGRlck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pzb21JbkYlWm9vbU91dEYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVGCFJlZnJlc2hGClV
uZGVybgluZUYHSXRhbG1jRgdab29tSW5CCFpzb21PdXRCClVuZGVybgluZUIGUHJpbnRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXB1c2luZnVzaW9uZnVzaW9uZnVzaW9uZnVzaW9uZnVzaW9uZnVzaW9uZnVzaW9uZnV
font-weight: normal;
font-style: normal;
}
.e-ddb-icons {
font-family: 'e-ddb-icons';
speak: none;
font-size: 16px;
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-basic::before {
content: "\e726";
}
.e-flow::before {
content: "\e724";
}
.e-connector::before {
content: "\e725";
}
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
}
```

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

The [symbolMargin](#) property is used to create the space around elements, outside of any defined borders.

Symbol preview

The symbol preview size of the palette items can be customized using [symbolPreview](#).

The [width](#) and [height](#) properties of SymbolPalette enables you to define the preview size to all the symbol palette items.

The [offset](#) of the dragging helper relative to the mouse cursor.

The following code example illustrates how to change the preview size of a palette item.

INDEX.JS

```
/**
 * Default symbol palette sample
 */
//Initialize the basicshapes for the symbol palatte
var basicShapes = [{
  id: 'Rectangle',
  shape: {
    type: 'Basic',
    shape: 'Rectangle'
  },
  tooltip:{
    content:"Rectangle Tooltip",
  }
},
{
  id: 'Ellipse',
  shape: {
    type: 'Basic',
    shape: 'Ellipse'
  }
},
{
  id: 'Hexagon',
  shape: {
    type: 'Basic',
    shape: 'Hexagon'
  },
  tooltip: {
    content: 'Hexagon Tooltip',
  },
  //customized content of the Tooltip is enabled by Node Tooltip
  Cosnstraints
  constraints: ej.diagrams.NodeConstraints.Default |
  ej.diagrams.NodeConstraints.Tooltip
}
];
//Initializes the symbol palette
```

```
var palette = new ej.diagrams.SymbolPalette({
  expandMode: 'Multiple',
  palettes: [{
    id: 'basic',
    expanded: true,
    symbols: basicShapes,
    title: 'Basic Shapes',
    iconCss: 'e-ddb-icons e-basic'
  }],
  symbolHeight: 80,
  symbolWidth: 80,
  symbolPreview: {
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5
    }
  },
  symbolMargin: {
    left: 12,
    right: 12,
    top: 12,
    bottom: 12
  },
  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>
<style>
    @font-face {
        font-family: 'e-ddb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+ldAAACIAAAAIJnbHlmw/g
RIAAAvGAAACw0aGVhZBGJTLcAAADQAAANmhoZWEIXQQpAAAArAAAACRobXR4oAAAAAAYAAAAAC
gbG9jYdYyye4AAAKkAAAAUm1heHABOAD4AAABCAAAACBuYW1ldAwInAAALyWAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAKAABAAAAQAAJo24vV8
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAAAAAAAEAAAAoAOwABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
EAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQA
AAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAA
AAAQAAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAQAQAAAEAAAABAAAAIAAADAAAAFAADAAEAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAQAAAAQAQACAAMABAAFAAYABwAIAAaACgALAAw
ADQAOAA8AEAAARABIAEwAUABUAFgAXABgAGQAaABsAHAAADAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMIJJaAnaClYLMauqC7gMpg2ODmQ
Owg8ad9IQoBF6EL1YTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDaKEBAMCAQEBAQEBAgMEBAKMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEHQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQEERIRDw8PDw4
PDg4NDhoZGBp6XfoYegkICQcIBgYGBQQAQAwMCAQEBAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPEA
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBA0LDg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUExEPDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHBgYUFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQEERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAQKQCw4PERIUFhYMCw0
MDA0NDQwNDAMwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAMwNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQEERIRDw8PDw4PDg4
NDhoZGBp6XvOYEWkJCAgHBwYFBQUDAwMCAQICAwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAQEBERERI
REhEQERAQE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDAMwNCwwWFhQSEQ8OCwoEAWMCAgE
BAgIDAQKQCw4PERIUFhYMCw0MDA0NDQwNDAMwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxE
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAMwMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAQEBEREBEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEPDy8PAYOJAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDALCwoKCQkICACGBgQFAwMCAwCawUHCALDQ4OEBARESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAgGBwUFBQDAgEBAgMEBAUFbWYICakJCgoLDAsNDA0ODg4
ODw8QDxAPDw4ShQTExERDw4OCwsJBwYFAGEBAGUGBwLcW0PBxAREhMUAcAAAAABAAAAAD9AO
1AAMABwAvADMAAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAQAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBoF7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfzMRIRE7AT8HNRE
1LwcjISMPBwGDFx0BtT4+/kp9fT4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQKQKwJBwUCAGUHCQoLCgQAAAAAAGAAAAADtQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCAcFAQE
```


BAQEBBQcICgsGBwZ9Pv2QPn0GBwYLCggHBQEB+X58frwCvP2OBgYGCwoJBgUCAQECBQYJCgsGBgY
CcgYGBgsKCQYFAGf9/gwB9H0BAGUGCQoLBgZ2/ooBdrwAAAADAAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMfDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCAcIBgYGBAQEAgEBAQECEBAQEBgYGCACJCAk
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgCHCAgICQoJfBwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAGJCQsLCwwNExAPBgYFBQQDAwIBAQEBCAcICgwNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICaKkCQoJCQkICACHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITExQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCAG
JCQoLDAwMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+iX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAgMFBQYHCAkKCgsMDAwNabUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCAcFAQH9kLv6Au76+okGBwYLCggHBQEBABQEHCAoLBgcGAj07fx0/AAA
AAQAAAAADdwN3ANEAABMhJz8L0wEfHR0BDx0jLw8jHx47AT8dPQEvHSMPTDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUFBAMCAgEBAgIDBAUFQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUFBMSEA8NDAsIB14EBAQFBgCHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTEhISExESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQcJCwwODxESExU
VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTEhMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwcBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAwQFCg00EBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgCJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAwQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFBQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
VFBQTEhIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLXU9AT8UHwYnDxMVHxY/BwEfAjSbPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEExAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUExIQDgwLBAQEAgICAQECAgMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCgsNDQ4PEBAREhM
TEhQVFRoaGhkZfXyWAVEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgCICQoMDA0PDw8RERI
SEhQVFBVFBQCBzflBgsODxESFBYWDawMDAwNDQwNDA0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMfHUUehEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUEhQVFBVFBVFBMTExIREQ8QDg0NDAoKCAcGBQQAQEEBQgKDA4Q/qSEAgICAgQEBQUBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAABQAAAAADvAO8AAMAIwArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFQYFBQQEAgICAgQEBQUGBQUFBQAQAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAAA08A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYyE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVarX9Lm8CBAYICQoKAxYKCGkIBgQCAQMFBwgKCGAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbgCJCgoMDA4OEBAREhITGRgWFxcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCAGIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEAwIBWYIDBQYICsNDQ8RERMUFUXGBgZDRobG0cTEhIQEA4NDAoJCAyFBAI
BrAE7ATsAAAAAaaaaAvDhAAiAEUakAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPu1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBgCICGoLDA0ODxBjXhAPDg4MCwkJCAcGBAQDAQEBAgMEBQc
HCQsKDA00DhAQVG/tDhsaGRgWFRQTCAGHBwYGBQQEAWMCAQECEBAUGCAoKDA00Dw8REhIPDg4NDAs
KCQkHBgUEAwEBAgQGCaoLDhAREhQVFxga9wHIAQIDBAUFBwcICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAD4BTgEBAgMBAUGBwcJCQkLCwwPDQwMCwoJCQcHBQQEAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwcICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAAD9AN3AAMAHwBUAAABayETJzMfDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAghBgYLCgoVBQ00EakKAXL+IAkJCAcHBwUfLhkFCgkGBQIBAxXMAwICAQIFBgkKCwYghAEBBQc

ICgsGB/6LBwYGCwoKFQUNdHAJCr0GBgI+/okBd/oBAQIFBwCQAwcGBAIBfQEBaWQFBgcI/tMCCzo
CBwkKCwYG/UoBmgcHBwCGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwCQAwcGBAIBAQIAAAAABgA
AAAADaQ08AAMABwALAB8AIwBeAAALMxEjAzMRIwMzESM1EQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwCICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBvW/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwC
ICAgJAn83PgSKAgGBAIBAAABAAAAA08A7wAxgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQGKdG4cGhoZFxcVFBMQEDfegQ00EBITFBWGBgZGhsbGxwaGhoZGRcXfHUUFBIREA40DAoJCAY
FAGEBAGUGCAkKDA4OEBESFBQVfHcXGQwaGRsdEBAQEAE8PDw8PDg4ODQ0MDAwLCwsKChIIBwCHBgU
ENgUGBwCICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxyYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhARExQWGBgad984GRcXFRQSEQ80DAo
JBgUDAQECBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwGICQkKCgoMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKCQkIBwCHBQEAwM
CAQEDBacJCwwNDxESExUWFxkZGhscHR0eHh4eHR0cGxoZGRcWFBQSEQ80DAoJBwQDAQMFAAAAgA
AAAADFQ08AAMAAaAAANYe1IREfHjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwCGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFHcWfHqVExERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwCICakJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBAcICww
NDxERExQVfHfYB/wAAAAEAAAAAArEDvAADAAALMwEjAU86ASg6RAN4AAADAAAAAAOQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBaWUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEExER/vUPDw8NDgwMDAsLCgkJCAcHBwUFAwMCAgICAwMFBQcHBwGJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwC
ICQoJCwsMDAwODQ8PDxAQEBAERERASERARERAQEAJkZGRkZGQOCAkRERMTFRUWFRUVExMREREOQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTEExEREQ4NCwkHBQMBAQMFBwKLDZEHBwGJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQEAE8PDw0DA0LCwsKCQkIBwCHBQUDAwICAQMEBwGLDA6pBAMCAgI
BAGIDBwkKBQUMDQWFBQqqFhYXGBgZGRsRERAREBAQEAE8PDw0DA0LCwsKCQkIBwCHBQUDAwICAgI
DAwUFAMAAAAAA5ADkaADAEQAYwAAASE1ISufCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPw
vHisBDwUBOAes/tQBIwcHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTEExIREA8MCwkHBQMBAQM
FBwKLD8QERITEExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUDAwICAgIDAUFwBwHCak
JCgsLCw0MDg0PDhAQEBAQERARERSZGRgYFxyWqgQFBgUGB0MBQUKQCcDAQMDAQMHq4MCwgHBAM
BAQECawQEBgYHBwGJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVfHUVFRM
TERERDg0LCQcFAwEBaWUHCQsNDhERERMUFBUWFRUVExMREREOQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxAQEBAEREBEREhARERAQEBApDw8NDgwNCwsLCgkJCAcHBwUFAwMCAgEDBACICww
OqQQDAgICAQICAwCJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBApDw8NDgwNCwsLCgkJCAcHBwU
FAwMCAgICAwMFBQAAAgAAAAADkAQABsAtgAANw8CFR8FIT8FNS8FIQ8BARC7AR8KDXArAS8WPwg
nNw8BJyMfCRUFgJ9WlwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAgEBAgICAwP8+gMDAg8
HOGUFBgkJAwQDAgULAQEDBAIFBwCLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQAFBgYHCAGKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBIKCwsMAGMKOCN1LM4CAwNJAwmCAgIBAQICAgMDSQM
DAgICAQECAPMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBgQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJKxIAgFAgIEAQIDJgcEAQYuAwMEBAQFBBe14jgfgGhoODg0MDAsKCgkICQcIBGcFBQQEAgIBAQE
EAgMEBAkKBGcHBw8QEBENDxoYESUqMLYYFRAFBUBAQcCAGIQGwEFBQAEAAAAAAOQA5AAAwAjACc
ARQAAARUhNScfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgM
EBAUFBBQFAwQCAgICBAMFBAUFBBQBM/7UZDIYcQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAQMDAQEBQUBAQDAgIBA+WWlpYBBQFBBgYHCAj+opaWAV4HCASGBwUEAvO
AAAEAAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBAcICQoL+QIEBwGJCgtjCgoJCAcEAvkLCgkIBwCAGQHCakKC/kCBAcICQoKXgsKCgOABQk
KCvoCBACICQoLQIYwoKCQgHBAL5CwoJCAcEAgIEBwGJCgv5AgQHCakKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwGPCAAMABwAJAFUAmwAAARUhNQEVIZUHNSMVHw8hPw81FxEjNS8PIQ8PFMS
RNQ8PER8PIT8PETUvDzECyP5wASyWlMQBAQIEBAUGBgICakJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgICakJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKArwKCgoJCQgIBwYGBQQEAgEBAgIDBAQGBp8HBwCICAgJCgFqyMgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAwMEBQYGBwGICQkKCgq+oP3uyAoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKyAK8ZAEBAGQEBQYGBwGICQkKCgr9RAoKCgkJCAgHBgYFBQAQAEBAQIEBAU
GBGcICakJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAAOQA5AAbQCxAAABHwQPCCIpQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAgMEuAUFBgCgAwgFAwMCAgE

jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDCMRDQsIAwMBAQECawIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAgIDAUFbwcGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECawUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEBlMBAgQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgwNDawbHRsOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAgEBAgMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAgMFBggJCgsMDQ0NDg8PAfQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAgEBAgQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUFCZBBQiHhEQ2Q+iaioZEwkGAQECBQCBQMDAwUaRQHxyRcXFhUWFRUUExE
QBw4MCwkDBAICAgEBAwQGBwkLDQ0PEBAREhMTDScTFQkIBgYFBQOEAwEBAQMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBAwUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwcFBAMCAQEBawEUAWQBazUGKwQEBAMEAgILVv4rIR4ICAC
BCA0xCwICAgMEBgICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCACGBQYUCW8IBwcICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQQDAgEBAAAAAAMAAAAA/QDcAAqAFYAuQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbWUFBAMCAgEBAwSaCAG
KDAsMCwv9wAYFAwMDAQIDBJoICAoMCwLcJIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhgIEBQcJCQlNCAgFBQQDAQEBQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwwM/uMCAGQFBgcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQBAGMDAwUEBQYGVggHBwUFBABAU8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQECBACJCgw
MxQgIBwgICAgICQkJCQYKCQgHBAQBVAwMCwoKCgkJCACGBQQDAQEQAQDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBgcAAAAABQAAAAADXgOQACEAQwB1AGkAxQAAAREPBy8HET8HHwYHEQ8HLwcRPwcfBgCRDwc
vBxE/Bx8GNxcjNychIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUFBAQ
DAgEBAQECawQEBQUFBQOEAwIBfAEBAgMEBAUFBUQEBAMCAQEBQIDBAQFBQUFBAQDAgF8AQECawQ
EBQUFBQOEAwIBAQBAGMEBAUFBUQEBAMCAbAU1xRCIn0FBQOEAwIBAQBAGMEBAUFQGIBAwMEBAU
FBgYHBwcHCAHCCAChBwcGBgUFBQDAwECGQUFBQDAgEBAQECawQEBQWWIgQFBwcICakKvwkKCAg
HBwUCcP68BgQEBAMDAQEBQMDBAQEBgFEBgQEBAMDAQEBQMDBAQEBv68BgQEBAMDAQEBQMDBAQ
EBgFEBgQEBAMDAQEBQMDBAQEBv68BgQEBAMDAQEBQMDBAQEBgFEBgQEBAMDAQEBQMDBAQEBzI
yJFYBAQIDBAQFBRkFBQOEAwIBaf3zCACHBwcGBgUFBQDAwECAGEDAwQEBQUGBgCHBwCIAg0BAQI
DBAQFBRkFBQOEAwIBAVYICACFBQMCAQECawUFBwgAAAAAAQAAAAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAcP4vHIsBDwUbbBIRERAPEA4OSAQFBAUEBQoEBAMAgE
BAGMEBQYGBuoFBQOEBAQMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkIBwgHBgYKBQM
DAwEBAQEBAMDAwUKBgYHCAcJCQkJCgsLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgsMAgQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWFhYWFBUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBQM
DBAYGCAgJCgoLDAwNDg4PEBAREhITEhMUEXQUFBMTExITEhIDcwcJCQoKCw0MRgMCAgEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQOEBSRDgwKCAyEAQEBQIDBAQFDAYHBwgJCAkKCgsKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBQIDBAUFBgHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAGQEBgYICakKCgsMDA0ODg8QEBESEhITEhXQTFBQUFBQUExQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAAATMfBBUHCwEPBjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBATjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRItCgEGBAIbGBAPLwGZiiEKBB0YFggBBwAABAAAAAAEAQAAMABwA
LACMAAAEVITUhFSE1ARUhNQmzfSERIxehESM1IRUjESERIxehNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAAUAAAAAAQAAAAEAQ
AAHYAAAEHIREhLwcPDx8PPw8hETmfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAChBQ
DAgEBAgMEBQCHCAoJCwsMDA0NDQ0MDAsLCQoIBwcFBAMCAQFAwAECawQFBwcICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwGL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBAgMEBQCHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAAwAFc
AbQCrAAABDwEVHxAFaQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwEBAQIDBQY
HCAoKDAwNDw8PEjP92QECaKABBAUICsNDxAREhQUFhYXfYVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVfHcXFhYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwcFBQOEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQQBAQQFCAkLDQ8QERIUFBYWFxcWFhQUEhE
QDw0LCQgFBAEBBAUICsNDxAREhQUFhYCCf7+AwQFBgcICQoLDAwNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwcHCAkJCQoLCwsMDAwNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAAEAAAAABAAAAAABABsAAQABAAAAAAACAACHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA

```

DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJ1aWxkZXJSZWd1bGFyTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lvbiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ
1aWxkZXJGb250IGd1bmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsaF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHcAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQB
nAHIAIYQBTAEIADQBpAGwAZABlAHIAATgBlAHcAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQBnAH
IAYQBTAEIADQBpAGwAZABlAHIAVgBlAHIAcWBPAG8AbgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQB
yAGkAYQBsaF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAbnAGUAbgBlAH
IAYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBuAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAoAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAACgBAGEDAQBBQEGAQcBCAEJAQoBCwEMAQ0BDgEPARA
BEQESARMBFAEVARYBFwEYARkBGGgEbARwBHQEeAR8BIAEhASIBIwEkASUBJgEnASgBKQAHWm9vbU1
uTQhab29tT3V0TQpVbmRlcmxpbmVNB1ByaW50TQROZXdnbnVhdmVNB0V4cG9ydE0FQm9sZE0LT3B
1bkZvbGR1ck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pvb21JbkYlWm9vbU91dEYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVcb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVGCFJlZnJlc2hGC1V
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFpvb21PdXRCClVuZGVybgGluZUIGUHJpbnRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVcb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFJlZnJlc2hCB010YWxpY0I
KRmxvd1NoYXBlcwldDb25uZWNOb3ILQmFzaWNTaGFwZXMAAAAAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-ddb-icons {
font-family: 'e-ddb-icons';
speak: none;
font-size: 16px;
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-basic::before {
content: "\e726";
}
.e-flow::before {
content: "\e724";
}
.e-connector::before {
content: "\e725";
}
}
</style>
<script>
var ele = document.getElementById('container');
if(ele) {
ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Default settings

While adding more number of symbols such as nodes and connectors to the palette, define the default settings for those objects through the [getNodeDefaults](#) and the [getConnectorDefaults](#) properties of diagram allows to define the default settings for nodes and connectors.

INDEX.JS

```
/**
 * Default symbol palette sample
 */
//Initialize the flowshapes for the symbol palatte
var flowshapes = [{
    id: 'process',
    shape: {
        type: 'Flow',
        shape: 'Process'
    }
},
{
    id: 'document',
    shape: {
        type: 'Flow',
        shape: 'Document'
    }
},
{
    id: 'predefinedprocess',
    shape: {
        type: 'Flow',
        shape: 'PreDefinedProcess'
    }
}
];
function setPaletteNodeDefaults(node) {
    node.width = 100;
    node.height = 100;
    node.style.strokeColor = '#3A3A3A';
}
var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [{
        id: 'flow',
        expanded: true,
        symbols: flowshapes,
        title: 'Flow 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>
<style>
  @font-face {
    font-family: 'e-ddb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+lDAAACIAAAAIJnbHlmw/g
RIAAAAlgAACw0aGVhZBGJTLcAAADQAAANmhoZWEIXQQpAAAArAAAACRobXR4oAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUm1heHABOAd4AAABCAAAACBuYW1ldAwInAAALywAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAKAABAAAAQAAJo24vV8
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAEAAQAAAAACAACAAAAAAAAAAEAAAAoAOwABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA

```


AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAADAAEAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAAQAEAAAAAQACAAMABAAFAAYABwAIAAkACgALAAw
ADQAOAA8AEAAARABIAEwAUABUAFgAXABgAGQAaABsAHAAaDAB4AHwAgACEAIGaJACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEBZwGkgd8B+YH/ghMCMIIJaAnaC1YLMauqC7gMpg2ODmQ
Owg8aD9IQoBF6EL1YTRhRGFIQUwBVMFhOAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDAKEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhOZGBp6XfoYegkICQcIBgYGBQQAeAwMCAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPEA
RERESEREBEQEBAPDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBAoLDg8REhQWFgwLDQwMDQ0NDA0MDAwLFxUUExEPDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRcLDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHBgYUFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAQKcW4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAGQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhOZGBp6XvOYEWKJCAgHBwYFBQUDAwMCAQICAwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAAREBERERI
REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDAwNCwwWFhQSEQ8OCwoEAWMCAgE
BAgIDAQKcW4PERIUfHYMCw0MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxET
TFBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAwMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPEIRIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEPDy8PayOJAu79Ej8BAgMDBQqGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkJCacGBgQFAwMCAXwCAwUHCAoLDQ4OEBARERESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDALDAsKCgkJCAgGBwUFBQAQDAgEBAgMEBAUFbWYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTEExERDw4OCwsJBwYFAGEBAGUGBwkLCw0PBxAREhMUAcAAAAAABAAAAAAD9AO
1AAMABwAvADMAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkJBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIWHC/scBOXwBoF7
HfAI+fP7HATl8ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9ft4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQKQKwoJBwUCAGUHCQoLCgQAAAAAAGAAAAADTQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCAcFAQE
BAQEBBQcICgsGBwZ9Pv2QPn0GBwYLCggHBQEB+X58frwCvP2OBgYGCwoJBgUCAQEBCQYJCgsGBgY
CcgyGBgsKCQYFAGF9/gwB9H0BAGUGCQoLBgZ2/ooBdrwAAAADAAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMfDR0BDw4jNQMHpw8vDz8MLw8hAi8KCQkJCAcIBgYGBAQEAQEBQEBAQEBAgYGCACJCAk
JCpx9CQoJCAgIBwcGBQQAeAwMBAQMDBAUFBgCHCAgICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAgJCQsLCwwNExAPBgYFBQQDAwIBAQEBCAcICgwNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkJCacHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITExQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCag
JCQoLDAwMDQLuQwMDAsKCQkJCAYFBAMCAQECAwQFBggICQkKCwwMDA3+ix36DQwMDAsKCQkJCAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAGMFBQYHCAkKCgsMDAwNabUNDAAwMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCAcFAQH9kLv6Au76+okGBwYLCggHBQEBQEBAQEBAQUHCAoLBgcGAj07fX0/AAA
AAQAAAAAADdwN3ANEAAABMhJz8LOWefHr0BDx0jLw8jHx47AT8dPQEvHSMPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUBAMCAgEBAgIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUFBMSEA8NDAsIB14EBAQFBgCHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQAeAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTEExISExESERERE8QDg8NDXECPOOJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQQAeAwICAQEDBQcJCwwODxESExU
VFhcQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQAeAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTEExMTEExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQAeAgIBAQIEBAUHBggJCQoLCwwNCQA

AAQAAAAADdwn3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAuQFCg00EBITFRYLDaw
MDAwNDQ0MDQwMDAwLfhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBGcJCQoMDA4ODxARERISFBMVFBVUFVBQCpzhhvzbhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDAMwMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAuQFCwW0EBITFRYLDaw
MDA0MDQ0MDQwMDAwLfhUTEhAODAsFBAMDAgIBAQICAwMEPAYICaKLCw0NDhAPERESExMUFBQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFS
BLxU9AT8UHwYnDxMVHxY/BwEfAjSbPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTExAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUEXIQDgWLBaQEAgICAQECAgMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBWgKCGsNDQ4PEBARhM
TExQVFRoaGhkZFxYWAVEEBQUFBGUEBQMDAgICBP6vEA4NCggGAwIBAgMFBGcICQoMDA0PDw8RERI
SExQUFBVUFVBQCbzfLBgsODxESFBYWDAMwMDAwNDQwNDAMCwwLfhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMfHUUehEPDQwJBAMDAgIBAQEBAgMEBD0GBWgJCwsMDg4
PEBAREhIUExQVFBVUFVBMTExIREQ8QDg0NDAoKCAcGBQQAQEEBQgKDA4Q/qseAgICAgQEBQUBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAO8AAMAIwArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQAeAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFbQYFBQQAeAgICAgQEBQUGBQUFBQAQAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA0A8A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAAA0A8A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYf/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCAkKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVArX9Lm8CBAYICQoKAxYKCGkIBgQCAQMFBWgKCGAAAAA
DAAAAA0A8A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbGcJCgoMDA4OEBAREhITGRgWfxcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCfBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFbMTExE
REQ8PDg0MCwkJCgcEAwIBWYIDBQYICsNDQ8RERMUFUXGBGZDRobG0cTExiQEA4NDAoJCAYFBAl
BrAE7ATsAAAAAaaaaAvDhAAiAEUakAAAAATmFDROBDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBGcICgoLDA0ODxBjXhAPDg4MCwkJCAcGBAQDAQEBAgMEBQc
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCAgHBWYGBQQAeAwMCAQECAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAgQGCAoLDhAREhQVFXga9wHIAQIDBAUFBwCICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAd4BTgEBAgMDBAUGBwCJCQkLCwwPDQwMCwoJCQcHBQQAeAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZEXMSEBAPDg0MCgoIBwUEAwMFBwCICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABAYETJzmfDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ0OEAKKAXL+IAkJCAChBwUf1hkFCgkGBQIBAxXMAwICAQIFBgkKCwYGHAEbBQC
ICgsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwCQAwcGBAIBfQEBAwQFBGcI/tMCCzo
CBwkKCwYG/UoBmgCHBwCGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwCQAwcGBAIBAQIAAAAABgA
AAAADaQ08AAMABwALAB8AIwBeAAAlMxEjAzMRIwMzESM1EQ8HIS8GNRElFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMhAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jCBAQI
DAwUEBgYGBwCICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBwV/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwC
ICAgJAn83PgSKAgGBAIBAAABAAAAA0A8A7wAXgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQgKdG4cGhoZFxcVFBMQEDfegQ00EBITFBWGBGZGhsbGxwaGhoZGRcXfHUUFBIREA4ODAOJCAY
FAGEBAGUGCAkKDA4OEBESFBQVFhcXGQwaGRsdEBAQE8PDw8PDg4ODQ0MDAwLCwsKChIIBwCHBgU
ENgUGBwCICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhAREXQWGBgad984GRcXFRQSEQ8ODAo
JBgUDAgEYCBQYHCQsMDQ8QERITFRUWfxcZGRkaGxobGrkYGBcWFRQTEExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBWgICQkKCgoMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKChIIBwCHBgUEAwM
CAQEDBACJCwwNDxESExUWFxkZGhsCHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODAoJBwQDAQMFAAAAAGa
AAAADFQ08AAMAAaAAANyE1IREfhjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwCGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhcWfHqVEXERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwCICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww
NDxERExQVfHYB/wAAAAEAAAAAArEDvAADAALMwEjAU86ASg6RAN4AAADAAAAAAOQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR

kZL8HBw0LCQcFAwEBAwUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEExER/vUPDw8NDgwMDAsLCgkJCAcHBwUFawMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHawEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwc
ICQoJCwsMDAwODQ8PDxAQEBARERASERARERAQEAJkZGRkZGQOCakRERMTFRUWFRUVExMREREODQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTEExEREQ4NCwkHBQMBAQMFBwkLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQEA8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAQMEBwgLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqFhYXGBgZGRsRERAREBAQEA8PDw0ODA0LCwsKCQkIBwCHBQUdAwICAQI
DAwUFAMAAAAAA5ADkAADAEQAYwAAASE1ISufCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwc
vHisBDwUBOAes/tQBIwcHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTExIREA8MCwkHBQMBAQM
FBwkLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUdAwICAQIDAUFBwcHCAk
JCgsLCw0MDg0PDhAQEBAREBERERsZGRgYFxyWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4MCwgHBAM
BAQECawQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBAwUHCQsNDhERERMUFBUWFRUVExMREREODQsJBwUDAQEDBQcJCw2RBwcICQk
KCwsLDQwODQ8PDxAQEBAREBEREhARERAQEBAPDw8NDgWNCwsLCgkJCAcHBwUFawMCAgEDBacICww
OqQQDAgICAQICAwJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBAPDw8NDgWNCwsLCgkJCAcHBwU
FAwMCAgICAwMFBQAAAgAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARc7AR8KDxArAS8WPwg
nNw8BJymfCRUFgJ8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAGICAwP8+gMDAg8
HOGUFBGkJAwQDAgULAQEDBAIFBwLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQFbgYHCAgKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLcwoSEA0LBgYHBQIDAQEIawEBAGQBBIKCwsMAGMKOCN1LM4CAwNJAwMCAgIBAQICAgMDSQM
DAgICAQECAPMBAGIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJkXIAGfAgIEAQIDJgcEAQYuAwMEBAQFBBEL4jgfgHhODg0MDAsKCgkICQcIBgcFBQQAeAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFBUBAQCcAgIQGwEFBQAEAAAAAAQA5AAAwAjACc
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAQM
EBAUFBBQFAwQCAgICBAMFBAUFBBQBm/7UZDIyCQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAQMDAQEBAMDBAQEBQUBAQDAgIBA+WWlpYBBQFBBgYHCAj+opaWAV4HCAsGBwUEAvo
AAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBacICQoL+QIEBwgJCgtjCgoJCAcEAvkLCgkIBwQCAgQHCakKC/kCBacICQoKXgsKCgOABQk
KCvoCBacICQoLYwoKCQgHBAL5CwoJCAcEAgIEBwgJCgv5AgQHCakKC2MKCgkIBwQC+goKCQgHBAl
BAWUAAAAABQAAAAADwgPCAAMABwAJAFUAmwAAARUHNQEVIZUHNSMVHw8hPw81FxEjNS8PIQ8PFMS
RNQ8PER8PIT8PETUvDzECyP5wASyWlMQBAQIEBAUGBgICAKJCgoKASwKCgoJCQgIBwYGBQDAwE
BljIBAQMDBAUGBgICAKJCgoK/nAKCgoJCQgIBwYGBQDAwEBMgoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICAKJCgoKArwKCgoJCQgIBwYGBQQAeAgEBAGIDBAQGBp8HBwcICAQJCgFqyMgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAgHBgYFBAMDAQEBAQM
DBAUGBgICAKJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAgHBgYFBQAQAEBAQIEBAU
GBgcICAKJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAAQA5AAbQCxAAABHwQPC8IPQE
PFhUfAQ8ELw4/Fz0BPwgfAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAGMEuAUFBgGAWgFAwMCAgE
jHxsYCwoJCQgIBgcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDCMRDQsIAwMBAQECawIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAgIDAUFBwcGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECawUGCAkKCwwNDA4ODw/+DA8PDg4NDADwM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEB1MBAGQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBSaGgwNDAwbHRSOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAgEBAGMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAGMFBggJCgsMDQ0NDg8PaFQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAgEBAGQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUFCABBBQiHhEQ2Q+iaioZEwkGAQECBQCBQMDAwUaRQHxyRcXFhUWFRUUEX
QBw4MCwkDBAICAgEBawQGBwkLDQ0PEBAREhMTDScTFQkIBgYFBQQAeAwEBAQMEBggJCwsNDQ8PERA
RERIREKECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBawUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCHBQMCAQEBawEUAwQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgcICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCAcGBQYUCw8IBwcICQoLDAw
MDhMSEhAQDg0MCgoJCAcGBQDAgEBAAAAAAMAAAAA/QDcAAqAFYAUQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbwUFBAMCAgEBawSaCag
KDAsMCww9wAYFAwMDAQIDBjOICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCAcGAWL+bhISEhM
SEA4NhgIEBQcJCQ1NCAgFBQDAQEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAGIEBQYHCAkJCgoKCwwM/uMCAgQFBgcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQBAGMDAwUEBQYGVggHBwUFBaIBAU8CBAUICakJLAoJCAcGAWICBAU
ICakJWQEEBgcKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQEBCACJCgw

2846

```

        font-weight: normal;
        font-style: normal;
    }
    .e-ddb-icons {
        font-family: 'e-ddb-icons';
        speak: none;
        font-size: 16px;
        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-basic::before {
        content: "\e726";
    }
    .e-flow::before {
        content: "\e724";
    }
    .e-connector::before {
        content: "\e725";
    }
}
</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 symbol description for symbols in the palette

The diagram provides support to add symbol description below each symbol of a palette. This descriptive representation of each symbol will enhance the details of the symbol visually. The height and width of the symbol description can also be set individually.

- The property `getSymbolInfo`, can be used to add the symbol description at runtime.

The following code is an example to set a symbol description for symbols in the palette.

INDEX.JS

```

//Initialize the basicshapes for the symbol palette
function getBasicShapes() {
    var basicShapes = [
        { id: 'Rectangle', shape: { type: 'Basic', shape: 'Rectangle' },
        style: { strokeWidth: 2 } },
        { id: 'Ellipse', shape: { type: 'Basic', shape: 'Ellipse' },
        style: { strokeWidth: 2 } },
        { id: 'Hexagon', shape: { type: 'Basic', shape: 'Hexagon' },
        style: { strokeWidth: 2 } },
    ];
}

```

```

        return basicShapes;
    }
    //Initialize the flowshapes for the symbol palette
    function getFlowShapes() {
        var flowShapes = [
            { id: 'Process', shape: { type: 'Flow', shape: 'Process' },
            style: { strokeWidth: 2 } },
            { id: 'Document', shape: { type: 'Flow', shape: 'Document' },
            style: { strokeWidth: 2 } },
        ];
        return flowShapes;
    }
    var palette = new ej.diagrams.SymbolPalette({
        expandMode: 'Multiple',
        palettes: [
            { id: 'flow', expanded: true, symbols: getFlowShapes(), title:
            'Flow Shapes' },
            { id: 'basic', expanded: true, symbols: getBasicShapes(), title:
            'Basic Shapes' },
        ],
        width: '100%', height: '100%', symbolHeight: 80, symbolWidth: 80,
        //Defines the symbol description for the symbols in the palette
        getSymbolInfo: function (symbol) {
            return { width: 75, height: 40, description: { text:
            symbol.shape['shape'] } };
        }
    });
    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>
<style>
  @font-face {
    font-family: 'e-ddb-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+1DAAACIAAAAIJnbHlmlw/g
RIAAAavgAACw0aGVhZBGJTLcAAADQAAAAANmhoZWEIXQqpAAAArAAAACRobXR4oAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAD4AAABCAAAACBuYW1ldAwInAAALywAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAAAAAABAAAAAQAAJo24vV8
PPPUACwQAAAAAANc1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAAAAAAEAAAAoAOWABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAAQACAAMABAFAAYABwAIAAKACgALAaw
ADQAOAA8AEAARABIAEwAUABUAFgAXABgAGQAaABsAHAAdAB4AHwAgACEAIgAjACQAjQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEBZwGkgd8B+YH/ghMCMIIJaAnaClYLMauQC7gMpg2ODmQ
Owg8ad9IQoBF6ElYTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBp6XfoYegkICQcIBgYGBQOEAwMCAQEBAgMEBQUGBwgICQoKCwwMDA0ODg4PDxAPER
ERESERESEREBEQEBAPDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAdICAQECAgMDBAoLDg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUExEPDQwKAwQDAGEBABEQAI
DBAMKDA0PERMUFRCLDawMDQwNEhERERAREAE8PDw4ODg4MDAwLCgoJCAgHBgYUFBAMCAgECawMDBQU
FBw0QEhMy+176EwsLDawNDQ4ODg8ODw8PEA8REhEQERAQEAE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXwDjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAwwQCkCw4PERIUfHYMCw0
MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDawNCww
WFhQSEQ8OCwoEawMCAgH9rgEBAGQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBp6XvOYEWkJCAgHBwYFBQUDAwMCAQICAwQFBQYHCAgJCgoLDawMDQ4ODg8PDxAREBERERI
REhEQERAQEAE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDawNCwwWFhQSEQ8OCwoEawMCAgE
BAgIDAwwQCkCw4PERIUfHYMCw0MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxE
TFBUXCwwMDA0MDRIREREREQERAPDw8ODg4NDawMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBAAPDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEpDy8PayOJAu79Ej8BAgMDBQqGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDawLCwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBAREREShERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAgGBwUFBQDAgEBAGMEBAUFBwYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTExERDw4OCwsJBwYFAGEBAGUGBwkLCw0PBxAREhMUAcAAAAABAAAAAD9AO
1AAMABAvADMAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBoF7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUZNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGdFX0BtT4+/kp9ft4BeHwFBAoLCgkHBQICBQcJCgSKBAX9kAUECgSKCQcFAsi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQQKCwoJBwUCAgUHCQoLCgQAAAAAAGAAAAADTQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgSGBwYC7gYHBgSKAcFAQE
BAQEBBQcICgSGBwZ9Pv2QPn0GBwYLCgGHBQEB+X58frwCvP20BgYGCwoJBgUCAQEGBQYJCgSGBwY
CcGyGBgSKCQYFAGF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAADAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMfDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCAcIBgYGBAQEAgEBAQEGBAQEBgYGCACJCAk
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgcHCAgICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
```

EBAYGCAgJCQsLCwwNExAPBgYFBQQDAwIBAQECBAcICgwNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICaCHBgUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQN3/RJ9AQIDBAUGCAg
JCQoLDAwMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+iX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAgMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCAcFAQH9kLv6Au76+okGBwYLCggHBQEBAQEBAQUHCAoLBgcGAj07fx0/AAA
AAQAAAAADdwN3ANEAABMhJz8LowEfHR0BDx0jLw8jHx47AT8dPQEvHSPMDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUFBAMCAgEBAgIDBAUFBQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXFxYUFBMSEA8NDAsIB14EBAQFBgCHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxACEBE
SEhISEhMTEhISEhESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAgIEAwUFBQCACGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQCJCwwODxESExU
VFhCQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTEhMTEhISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAwAAAAADvAO
8AAsAbADWAAABIXuzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwcBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAgIDAwQFCg00EBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBGcJCQoMDA4ODxARERISFBMVFBVFBQCPzhvzbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAwQFCww0EBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFBQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEhIREQ8QDg0NCwsJCAGGBAQCAgQAAAAAAwAAAAADuQO8AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEhAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUEXiQDgWLBQEAgICAQECAgMEBAoLDg8
REhQVFWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCgsNDQ4PEBAREhM
TEhQVFRoaGhkZFxyWAVEEBQUFBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBGcICQoMDA0PDw8RER
SExQUFBVFBQCbzLFbGsODxESFBYWDawMDAwNDQwMDA0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0MDAwMFhUUEhEPDQWJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVFBMTEhIREQ8QDg0NDAoKCAcGBQQAQEEBQgKDA4Q/qseAgICAgQEBQUFBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAO8AAMAiWArAC8
ASgAAARUhNScPAh0BHwU7AT8FPQEvBSsBDwELESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFBQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFBQYFBQQEAgICAgQEBQUGBQUFBQAQAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA08A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/mA4Ahw4/mABoDgBoAAEAAAAA08A7wABwALABgAMwAAARUjNSMViZUBESERiXehETMRIxehESM
nESMRfYE/BhEvBiEPBgJvpzc4Ab391DcMjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAc
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVArX9Lm8CBAYICQoKAXYKCGkIBgQCAQMFBwgKCGAAAAA
DAAAAA08A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbGcJCgoMDA4OEBAREhITGRgWfXcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBWUfHgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFBMTEhE
REQ8PDg0MCwkJCgcEawIBWyIDBQYICsNDQ8RERMUFURXGBgZDRobG0cTEhIQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUAKAAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgWLCgkIBgYEAwICAwQFBGcICgoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCAGHBwYGBQQEAgMCAQECAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAgQGAoLDhAREhQVFxga9wHIAQIDBAUFBwCICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAD4BTgEABgMDBAUGBwcJCQkLCwwPDQwMCwoJCQkHBQQEAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFbWcICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABAYETJzmFDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ0OEAKKAXL+IAkJCAChBWUfLhkFCgkGBQIBAXXMAwICAQIFBgkKCwYGHAEBBQc
ICgsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwcQAwcGBAIBfQEBAwQFBGcI/tMCCzo
CBwkKCwYG/UoBmgCHBwcGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwcQAwcGBAIBAQIAAAAABgA
AAAADaQO8AAMABwALAB8AIwBeAAALMxEjAzMRIwMzESM1EQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI

DAWUEBgYGBwCICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBvW/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwC
ICAgJAn83PgSKCAgGBAIBAAABAAAAA08A7wAxAQAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQGDg4cGhoZFxcVFBMQEDfegQ00EBITFBWUGBgZGhsbGxwaGhoZGRcXFhUUFBIREA40DAoJCAy
FAgEBAgUGCAkKDA40EBESFBQVFcXGQwaGRsdEBAQEASPDw8PDg4ODQ0MDAwLCwsKChIIBwCHBgU
ENgUGBwCICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxFYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4da60FBASMDhARExQWGBgad984GRcXFRQSEQ80DAo
JBgUDAQECBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwGICQkKCgoMCwwMGg4ODg8PDw80EhIREBEQDw8PDg4NDQwLCwsKCQkIBwCHBQUEAwM
CAQEDBACJCwwNDxESExUWFxkZGhscHR0eHh4eHR0cGxoZGRcWFBQSEQ80DAoJBwQDAQMFAAAAAGa
AAAADFQ08AAMaAAANyE1IREfHjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwCGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhCWfHqVExERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwCICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWFhUUExERDw0MCwgHBAMDBACICww
NDxERExQVfHqYB/wAAAAEAAAAAARedVaADAAALMwEjAU86ASg6RAN4AAADAAAAAAQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBAwUHCQsNDhERERMUFBUFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEER/vUPDw8NDgMDAsLCgkJCAcHBwUFAwMCAgICAwMFBQcHBwGJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwC
ICQoJCwsMDAwODQ8PDxQEBAERERASERARERAQEAJkZGRkZGQCAkRERMTFRUFRUVExMREREODQs
JBwUDAQEDBQcJCw00ERERExMVFRUWFRUTEEREQ4NCwkHBQMBAMFBWkLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQEASPDw0ODA0LCwsKCQkIBwCHBQUdAwICAQMEBwGLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqFhYXGBgZGRsRERAREBAQEASPDw0ODA0LCwsKCQkIBwCHBQUdAwICAgi
DAwUFAAAAAA5ADkaADAEQAYwAAASE1ISUfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwc
vHisBDwUBOAes/tQBIwCHDQsJBwUDAQEDBQcJCw00ERERExQUFRYVFRUTExIREA8MCwkHBQMBAM
FBWkLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUdAwICAgiDAwUFBwCHCAk
JCgsLCw0MDg0PDhAQEBAQERARERSZGRgYFXYWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4MCwgHBAM
BAQECawQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBAwUHCQsNDhERERMFRUFRUVExMREREODQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxQEBAEREBERARERAQEBAPDw8NDgwnCwsLCgkJCAcHBwUFAwMCAgEDBACICww
OqQDDAgICAQICAwCJCgUFDA0MBQUKqHYWFxgYGRkbEREQERAQEBAPDw8NDgwnCwsLCgkJCAcHBwU
FAwMCAgICAwMFBQAAAgAAAAADkAQABsAtgAANw8CFR8FIT8FNS8FIQ8BARc7AR8KDxArAS8WPwg
nNw8BJyMfCRUFgJ7WlWm1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAgICAwP8+gMDAg8
HOGUFBgkJAwQDAgULAQEDBAIFBwCLCw8SDA00GBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFBQAFBgYHCAgKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBIKCwsMAGMKOCN1LM4CAwNJAwmCAgIBAQICAgMDSQM
DAgICAQECapMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJkXIagFagIEAQIDJgcEAQYuAwMEBAQFBBe14jgfgHhODg0MDAsKCgkICQcIBgcFBQQEAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFBUBAQcCAgIQGwEFBQAEAAAAAAQA5AAAwAjACc
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgiM
EBAUFBBQFAwQCAgICBAMFBAUFBBQBM/7UZDIYcQ0HBGUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBBQEBAQMDAQEBAMDBAQEBQUFBQDAgIBA+WWlpYBBQFBBgYHCAj+opaWAV4HCAsGBwUEAvo
AAAAEAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBACICQoL+QIEBwgJCgtjCgoJCACeAvkLCgkIBwQCAgQHCAkKC/kCBACICQoKXgsKCgOABQk
KCvoCBACICQoLYwoKCQgHBAL5CwoJCACeAgIEBwgJCgv5AgQHCAkKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwgPCAAMABwAJAFUAmwAAARUHNQEVIZUHNSMVHw8hPw81FxEjNS8PIQ8PFMS
RNQ8PER8PIT8PETUvDzECyP5wASyWlMQBAQIEBAUGBgCICAKJCgoKASwKCgoJCQgIBwYGBQDQDAwE
BljIBAQMDBAUGBgCICAKJCgoK/nAKCgoJCQgIBwYGBQDQDAwEBMgoKCgkJCAgHBgYFBAQCAQEBAQIEBAU
DBAUGBgCICAKJCgoKArwKCgoJCQgIBwYGBQDQEAQIEBAQIDBAQGBp8HBwCICAgJCgFqYmgB9MjIyMj
ICgoKCQkICAcGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAgHBgYFBAQCAQEBAQIEBAU
DBAUGBgCICAKJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAgHBgYFBAQCAQEBAQIEBAU
GBgcICAKJCgoKAhIKCQkJCQgHCKkHBQUFAwMCAgAAAAACAAAAAAQA5AABQCxAAABHwQPCC8IPQE
PFhUfAQ8ELw4/Fz0BPwgFAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAgMEuAUFBgGAWgFAwMCAgE
jHxsYCwoJCQgIBgcGBgYFBAQDAgIBAQIFAQIEBgQDBAMDChMRDQsIAwMBAQECAwIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAgIDAUFBwCGBQX+JgoJCAyFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECAwUGCAkKCwwNDA4ODw/+DA8PDg4NDAwDM7gFBQYHBwYFBBgEawIBAQEDAwM
EBAUEBlMBAgQFBAMEBQUGBgCICQoLDA00DxAREhIpLwUFAwIBAQECAG8CHBSaGgwNDawbHRSOHw8

PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAgEBAgMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAgMFBggJCgsMDQ0NDg8PafQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BjYMHhwkTDwg3Fz8VLw8/Di8TAhEkFhcLCgk
JCQkJCAkHCAUEBAMCAgEBAgQFBwgKDA00EBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBgcJCgwODg4QEBIUFCZBBQiHhEQ2Q+iAiozEwkgAQECBQQCBQMDAwUaRQHxyRcXFhUWFRUUExE
QBw4MCwkDBAICAgEBAwQGBwkLDQ0PEBAREhMTDSctTFQkIBgYFBQQEAWEBABQMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBAwUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwcFBAMCAQEBawEUAWQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgcICgoMDQcPERMUCwsMDAwZExMREBAPDg4MCwsJCACGBQYUCw8IBwcICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQDQAgEBAAAAAAMAAAAA/QDCAaQAFYAUQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbWUFBAMCAgEBAwSaCag
KDasMCwv9wAYFAwMDAQIDBjOICAOmCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACgAwL+bhISEhM
SEA4NhgIEBQcJCQlNCAgFBQDQAEBAQMEBQUICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMADAICAgQ
DBwkKDAwNDmsBAgIEBQYHCAkJCgoKCwwM/uMCAGQFBgcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQBAGMDAwUEBQYGVggHBwUFBABAU8CBAUICakJLAoJCACgAwICBAU
ICakJWQEEBgcKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQEBCACJCgw
MxQgIBwgICAgICQkJCQYKCQgHBAQBVAMCwoKCgkJCACGBQDQAEQDAwLCgoKCQkIBwYFBAMBAQE
BAwQFBgcAAAAABQAAAAADxgOQACEAQwBLAGkAXQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUBAQ
DAgEBAQECawQEBQUBQQAeAwIBfAEBAgMEBAUFBUQEBAMCAQEBQIDBAQFBQUBAQDAgF8AQECawQ
EBQUBQQAeAwIBAQBAGMEBAUFBUQEBAMCAbAU1xRCIn0FBQQAeAwIBAQBAGMEBAUFQgIBAwMEBAU
FBgYHBwcHCAHCCaCHBwcGBgUFBAQDAwECGQUBAQDAgEBAQECawQEBQWWIgQFBwcICakKvwkKCAg
HBwUCCp68BgQEBAMDAQEBAMDBAQEBgFEBgQEBAMDAQEBAMDBAQEBv68BgQEBAMDAQEBAMDBAQ
EBgFEBgQEBAMDAQEBAMDBAQEBv68BgQEBAMDAQEBAMDBAQEBgFEBgQEBAMDAQEBAMDBAQEBzZ
yJFYBAQIDBAQFBRkFBQQAeAwIBaf3zCACHBwcGBgUFBAQDAwECAGEDAwQEBQUGBgCHBwcIAg0BAQI
DBAQFBRkFBQQAeAwIBAVYICACFBQMCAQECawUFBwgAAAAAQAAAAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHiSBDwUBbIRERAPEA4OSAQFBAUEBQOEBAAMCAgE
BAgMEBQYGBuoFBQQAEBAMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBwgHBgYKBQM
DAwEBAQEBAMDAwUKBgYHCACJCQkJCgsLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDAwMCgsMAgQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWFhYWFBUQEXQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBAM
DBAYGACAgJCgoLDawNDg4PEBAREhITEhMUEXQUBMTEhITEhIDcwcJCQoKCw0MRgMCAGEEAwMEBAQ
FBukGBwUFBQMCAQICAwQECgQFBQQAEBUSrdgwKCAyEAQEBQIDBAQFDAYHbwgJCAkKCgsKDasZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBgGHCQkKCwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQEACgQEBgYICakKCgsMDA0ODg8QEBESEhITEhXQTFBQUBQEXQ
TEhISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICAwQFBgABAAAAAAMKA48AKAAAAATmfBBUHCwEPbjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwM1QwUGBg8QRgl7giwiJgYCYAEIWRkIBAtjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRITcgEGBAIBGBAPLwGZiiEKBB0YFggBBwAABAAAAAAEAQAAMABwA
LACMAAAEVITUhfSE1ARUhNQmZFSERixEhESM1IRUjESERixEhNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAUAAAAAAQAQAAAAEAQ
AAHYAAAEHIREhLwcPDx8PPw8hETMfDz8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAChBQQ
DAgEBAgMEBQcHCAoJCwsMDA0NDQ0MDAsLCQoIBwcFBAMCAQFAwAECawQFBwcICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBAgMEBQcHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAChBQQDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAAwAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
HCAoKDAwNDw8PEjP92QEcAkABBAUICsNDxAREhQUFhYXFxYVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVfHCxYFhYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwcFBQEAwMBAgECAGMDBAKMDQ8RExQVFxgzDA0S/QABYgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLfhYUFBIREA8NCwkIBQQAQFCakLDQ8QERIUFBYWFxcWfHQUEhE
QDw0LCQgFBAEBBAUICsNDxAREhQUFhYCCf7+AwQFBgcICQoLDawNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUBwCHCAkJCQoLCwsMDAwNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
AABIA3gABAAAAAAAEAAAAABAAAAAABABsAAQABAAAAAAACAaHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAFAAsAWQABAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRG1hZ3JhbUJ1aWwXkZXJSZWd1bGFyTmV3IE1hdGVyaWFsX0RyYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RyYWdyYW1CdWlsZGVyVmVyc2lubiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ1


```

1aWxkZXJgb250IGd1bmVYyYXR1ZCB1c2l1uZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHcAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQB
nAHIAAYQBtAEIAdQBpAGwAZABlAHIAATgBlAHcAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQBnAH
AYQBtAEIAdQBpAGwAZABlAHIAVgBlAHIAcwBpAG8AbGAgADEALGAAAE4AZQB3ACAATQBhAHQAZQBy
yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIABnAGUAbgBlAH
AYQB0AGUAZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgBlAHMAaQBVAg4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuAGMAZgBlAHMAaQBVAg4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAACgBAGEDAQQBBQEGAQcBCAEJAQoBCwEMAQQ0BDgEPARA
BEQESARMBFAEVARYBFWEYARkBGGEBARwBHQEeAR8BIAEHASIBIWEkASUBJgEnASgBKQAHWm9vbU1
uTQhab29tT3V0TQpVbmR1cmxpbmVNB1ByaW50TQROZXNdNBVNhdmVNB0V4cG9ydE0FQm9sZE0LT3B
lbkZvbGRlck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pzb21JbkYlWm9vbU91dEYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkQgtPcGVuRm9sZGVyRgdEZWxldGVGCFFJlZnJlc2hGC1V
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFpzb21PdXRCClVuZGVybgGluZUIGUHHJpbnRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVCCFFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXBlcwldDb25uZWN0b3ILQmFzaWNTaGFwZXMAAAAAAA==) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-ddb-icons {
    font-family: 'e-ddb-icons';
    speak: none;
    font-size: 16px;
    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-basic::before {
    content: "\e726";
}
.e-flow::before {
    content: "\e724";
}
.e-connector::before {
    content: "\e725";
}
}
</style>
</script>
var ele = document.getElementById('container');
if(ele) {
    ele.style.visibility = "visible";
}
</script>
<script src="index.js" type="text/javascript"></script>
</body></html>

```

Appearance of symbol description

The appearance of a symbol description in the palette can be customized by changing its `color`, `fontSize`, `fontFamily`, `bold`, `italic`, `textDecoration`, and `margin`.

The following code is an example to change the color of a symbol description for symbols in the palette.

INDEX.JS

```

//Initialize the basicshapes for the symbol palette
function getBasicShapes() {
    var basicShapes = [
        { id: 'Rectangle', shape: { type: 'Basic', shape: 'Rectangle' },
        style: { strokeWidth: 2 } },
        { id: 'Ellipse', shape: { type: 'Basic', shape: 'Ellipse' }, style:
        { strokeWidth: 2 } },
        { id: 'Hexagon', shape: { type: 'Basic', shape: 'Hexagon' }, style:
        { strokeWidth: 2 } },
    ];
    return basicShapes;
}

//Initialize the flowshapes for the symbol palette
function getFlowShapes() {
    var flowShapes = [
        { id: 'Process', shape: { type: 'Flow', shape: 'Process' }, style: {
        strokeWidth: 2 } },
        { id: 'Document', shape: { type: 'Flow', shape: 'Document' }, style:
        { strokeWidth: 2 } },
    ];
    return flowShapes;
}

var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [
        { id: 'flow', expanded: true, symbols: getFlowShapes(), title: 'Flow
        Shapes' },
        { id: 'basic', expanded: true, symbols: getBasicShapes(), title:
        'Basic Shapes' },
    ],
    width: '100%', height: '100%', symbolHeight: 80, symbolWidth: 80,
    //Defines the symbol description for the symbols in the palette
    getSymbolInfo: function (symbol) {
        return { width: 75, height: 40, description: { text:
        symbol.shape['shape'], color : 'red', bold: true, fontSize: 15, fontFamily :
        'Arial', italic : true, textDecoratation : 'Underline',margin : {top : 30,
        left : 0, right : 0, bottom :30}} };
    }
});
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>
<style>
    @font-face {
        font-family: 'e-ddb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+lDAAACIAAAAIJnbHlw/g
RIAAAAvgAACw0aGVhZBGJTLcAAADQAAAAANmhoZWEIXQQpAAAArAAAACRobXR4oAAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAD4AAABCAAAACBuYW1ldAwInAAALywAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAAFwEAAAAAAEAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAAAABAAAA
PPPUACwQAAAAAANC1g90AAAAA1zWD3QAAAAAEAAQAAAAACAACAAAAAABAAAAAABAAAAAABAAAA
AAgAAAAoACgAAAP8AAAAAQAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAABAAAAAABAAAAAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
AAAAAABAAAAAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQAQA
UAAQAbgAAAAQABAABAADnJv//AADnAP//AAAAQAQAAAAQAQACAAMABAFAAYABwAIAAkACgALAAw
ADQAOAA8AEAARABIAEwAUABUAFgAXABgAGQAAABSAHAAdAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEbZwGkgd8B+YH/ghMCMIIJaAnaClYLMauqC7gMpg2ODmQ
Owg8aD9IQoBF6ElYTRhRGFIQUwBVMFhoAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFbY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVFwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBP6XfoYEGkICQcIBgYGBQQEAwMCAQEBAgMEBQUGBwgICQoKCwMDA00Dg4PDxAPER
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAGHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAGMDBAOLdg8REhQWFgWLDQwMDQ0NDA0MDAwLFxUUEXEPDQwKAwQDAgEBAQEBAQI
DBAMKDA0PERMUFRC LDAwMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAGHBgYUFBAMCAgECAwMDBQU
FBw0QEhMy+176EwsLDAwNDQ4ODg8ODw8PEA8REhEQERAQEAE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAAdzgPOAAMAXwDjAAATITUhbRU
PFcsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAQKcW4PERIUfHYMCw0
MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxETFBUXCwwMDA0MDQ0NDAwNCww
WFhQSEQ8OCwoEAWMCAgH9rgEBAGQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBP6XvOYEWKJCAGHBwYFBQUDAwMCAQICAwQFBQYHCAgJCgoLDAwMDQ4ODg8PDxAREBERERI
REhEQERAQEAE8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBALc/Hw0NDAwNCwwWFhQSEQ8OCwoEAWMCAgE
BAgIDAQKcW4PERIUfHYMCw0MDA0NDQwNDAwMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCgwNDxET
FBUXCwwMDA0MDRIREREQERAPDw8ODg4NDAwMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
```

RIxEPDy8PAYOJAu79Ej8BAgMdBQQBgCICAKJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDAwLCwoKCQkICACGBgQFAwMCAXwCAwUHCAoLDQ4OEBARERESEhERERAQDg4NCwUJCAyEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAGGBwUFBAQDAGEBAGMEBAUFwYICAKJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTEExERDw4OCwsJBwYFAgEBAGUGBwkLCw0PBxAREhMUAcAAAAAABAAAAAAD9AO
1AAMABwAvADMAAAEVITULFSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCggJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECawQGBgCICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBof7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9fT4BeHwFBALCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFasi7AbU
+Pvr6/c59ATn+xwIFBwkKCwoEBQJwBQQKCwoJBwUCAGUHCQoLCgQAAAAAagAAAAADtQP0ADcAPgA
AExEfCTMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBwYC7gYHBgsKCACFAQE
BAQEBBQcICgsGBwZ9Pv2QPn0GBwYLCggHBQEB+X58fwrCvP2OBgYGCwoJBgUCAQEBCQYJCgsGBgY
CcgyGBgsKCQYFAgF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAAADAAAAAMoA3cAIgBFAIUAAAEfDw8
OKwE1EzMfDR0BDw4jNQMHpw8vDz8MLw8hAi8KCQkJCACIBgYGBAQEAgEBAQEBCAQEBgYGCACJCAk
JCpx9CQoJCAgIBwcGBQUEAwMBAQMDBAUFBgCHAgICQoJfwbBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAgJCQsLCwwNExAPBgYFBQDawIBAQEBCACICGwNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICAKKCQoJCQkICACHBgUFBAMCARwBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAAACAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCag
JCQoLDawMDQLuDQwMDAsKCQkICAYFBAMCAQECawQFBggICQkKCwwMDA3+ix36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAGMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCACFAQH9kLv6Au76+okGBwYLCggHBQEBQEBAQUHCAoLBgcGAj07fX0/AAA
AAQAAAAADdwN3ANEAAABMhJz8LoweFHR0BDx0jLw8jHx47AT8dPQEvHSMPPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQUFBAMCAgEBAGIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXfXyUFBMSEA8NDAsIB14EBAQFBgCHCagJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgICAgQEBQYHBwgKCQsLDAwNDg4PDxAQEBE
SEhISExMTExISExESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAGIEAwUFBQcGCACJCQkKCgo
LDAwMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQUEAwICAQEDBQcJCwwODxESExU
VFhCQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQEAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEB
REhISEhMTExISEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQEAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAwAAAAADvAO
8AAsAbADWAAABIXuzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBAGIDAwQFCg0OEBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU
GBQUFBQAQAgICBP6sEA4MCggGAwIBAgMFBgCJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAGIDAwQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLFhUTEhAODAsFBAMDAgIBAQICAwMEPAYICAKLCw0NDhAPERESExMUFbQVFRQ
VExQSEhERE8ODgWMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUExIQDgwLBAQEAgICAQECAgMEBAoLDg8
REhQVfwwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAGECBAQGBwgKCgsNDQ4PEBAREhM
TExQVFRoaGhkZFxyWAVEEBQUFBgUEBQMDAGICBP6vEA4NCggGAwIBAgMFBgCICQoMDA0PDw8RERI
SExQUFBVFBQCbzfLBgsODxESFBYWDawMDAwNDQwNDaw0MCwwLFhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVFBMTExIREQ8QDg0NDAoKCAcGBQCAQEEBQgKDA4Q/qseAgICAgQEBQYFBQY
EBQFVFRYyGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvAO8AAMAIAwARc8
ASgAAARUyHNScPAh0BHwU7AT8FPQEvBSSBDwE1ESM1IRUjEQERIREDKwEPBhEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sjenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFbQYFBQQEAgICAgQEBQUGBQUFBQAQAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAAABAAAAA0A8A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/ma4Ahw4/maBoDgBoAAEAAAAA0A8A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYE/BhEvBiEPBgJvpzc4Ab391DcMjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCAC
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVarX9Lm8CBAYICQoKAXYKCGkIBgQCAQMFBwgKCgAAAAA
DAAAAA0A8A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8

XfHfYWFhUWFRUVFBQUExMFBgcJCgoMDA4OEBAREhITGRgWFxcXND0ODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBUWFhgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCAGIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEAWIBWyIDBQYICQsNDQ8RERMUFUXGBgzDRobG0cTExIQEA4NDAoJCAYFBAI
BrAE7ATsAAAMAAAAAAvoDhAAiAEUAKAAAATMfDR0BDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgwLCgkIBgYEAwICAwQFBgcICGoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQC
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCAgHBWYGBQQAawMCAQECBAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAGQGCAoLDhAREhQVFXga9wHIAQIDBAUFBwcICQoLCw0NDQwLCwoJCQgHBgUEBAI
BA4BTgEBAGMDBAUGBwcJCQkLCwwPDQwMCwoJCQCHBQQAeAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwcICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBAAAABayETJzmfDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ0OEAKKAXL+IAkJCACHBwUf1hkFCgkGBQIBAXXMAwICAQIFBgkKCwYGHAEBBQC
ICGsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwcQAwcGBAIBfQEBAwQFBgcI/tMCCzo
CBwkKCwYG/UoBmgcHBwCGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwcQAwcGBAIBAQIAAAAAABgA
AAAADAQ08AAMABwALAB8AIwBeAAALMxEjAzMRiWmZESMLEQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwcICAgJAbwJCAgIBwcGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBvW/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICACHBgYGBAQEAwIBAQIDBAQFBQYGBwc
ICAgJAn83PgSKAgGBAIBAAABAAAAA08A7wAxgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQgKDg4cGhoZFxcVFBMQEDfegQ0OEBITFBWGBgZGhsbGxwaGhoZGRcXfHUUFBIREA4ODA0JCAY
FAGEBAGUGCAkKDA4OEBESFBQVFhcxGQwaGRsdEBAQEASPDw8PDg4ODQ0MDAwLCwsKChIIBwcHBgU
ENgUGBwcICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4dA60FBASMDhARExQWGBgad984GRcXFRQSEQ8ODA0
JBgUDAQECEBQYHCQsMDQ8QERITFRUWFxcZGRkaGxobGRkYGBcWFRQTExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwgICQkKCgoMCwwMGg4ODg8PDw8OEHIREBEQDw8PDg4NDQwLCwsKCQkIBwcHBQUEAwM
CAQEDBACJCwwNDxESExUWFxkZGhsCHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODA0JBwQDAQMFAAAAAAgA
AAAADFQ08AAMAAaAANYe1IREfhjSBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ4NDg0PDg4ODg4NDQ0NDQwLDA0LCgkKCAkHBwcGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWFhcxWFhQVExERDw0MCgkHBABIN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICACHBgYFBAQDAgI
BAQICAwQEBQYGBwcICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWFhUUEXERDw0MCwgHBAMDBACICww
NDxEREXQVFhYB/wAAAAEAAAAAArEDvAADAAALMwEjAU86ASg6RAN4AAADAAAAAAQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBNGrkZGR
kZL8HBw0LCQcFAwEBAwUHCQsNDhERERMUFBUWFRUVEExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTExER/vUPDw8NDgWMDAsLCgkJCACHBwUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWFqoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwc
ICQoJCwsMDAwODQ8PDxAQEBAERERASERARERAQEAJkZGRkZGQOCAkRERMTFRUWFRUVEExMREREODQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTExEREQ4NCwkHBQMBAQMFBwkLDZEHBwgJCQoLCwsNDA4
NDw8PEBAQEBEQERESEBEREBAQEASPDw0ODA0LCwsKCQkIBwcHBQUDAwICAQMEBwgLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqqFhYXGBgZGRsRERAREBAQEASPDw0ODA0LCwsKCQkIBwcHBQUDAwICAgI
DAwUFAAMAAAAA5ADKAADAEQAYwAAASE1ISUfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPw
vHisBDwUBOAEs/tQBIwcHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTExIREA8MCwkHBQMBAQM
FBwkLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwcHBQUDAwICAgIDAUFwBwCHCAk
JCgsLCw0MDg0PDhAQEBAQERARERsZGRgYfXyWqgQFBgUGBg0MBQUKQCQDAQMDAQMHqQ4MCwgHBAM
BAQECawQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREBIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBAwUHCQsNDhERERMTRUFVhUVEExMREREODQsJBwUDAQEDBQcJCw2RBwcICQk
KCwsLDQwODQ8PDxAQEBAEREBERehARERAQEBApDw8NDgWNCwsLCgkJCACHBwUFAwMCAgEDBACICww
OqQODAgICAQICAwJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBApDw8NDgWNCwsLCgkJCACHBwU
FAwMCAgICAwMFBQAAAAGAAAAADKAQABsAtgAANw8CFR8FIT8FNS8FIQ8BARC7AR8KDxArAS8WPwg
nNw8BJyMfCRufGj8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAgEBAgICAwP8+gMDAg8
HOGUFBgkJAQWDAQULAQEDBAIFBwcLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUKBgUEAwMDAgEHAQM
DAWQECg0pHwEBpCyCJAImGg4MBQUCAwMCAgMFAQBFBgYHCAgKCgsMDQ4PEBAsEhMTAFM1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBIKCwsMAGMKOCN1LM4CAwNJAwMCAgIBAQICAgMDSQM
DAgICAQECapMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJKxIAGfAgIEAQIDJgcEAQYuAwMEBAQFBBE14jgfgHoODg0MDAsKCgkICQcIBgcFBQQAeAgIBAQE
EAgMEBAkKBgcHBw8QEBENDxoYESUqMLYYFRAFBUBAQCcAgIQGwEFBQAEAAAAAAQA5AAAwAjACc
ARQAAARUhNScfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgM
EBAUFBQQFAwQCAgICBAMFBAUFBQQBm/7UZDIyCQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBQQEBAQMDAQEBAQMDAQEBAQUBAQDAgIBA+WWlpYBBQQFBgYHCAj+opaWAV4HCAsGBwUEAvo

Copyright © 2001 -2024 Syncfusion Inc.

KCQsLDAwNDQ0NDawLCwkKCAcHBQQDAgEBAgMEBQcHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
 BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAcHBQQDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAWAFc
 AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
 HCAoKDAwNDw8PejP92QEcAkABBAUICQsNDxAREhQUFhYXFxYVFRQSERAPDQsJCAUEAQEEBQgJCw0
 PEBESFBUVFhcXFhYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
 NDg0NDawMCwsKCwkJCQgHBwcFBQUEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
 SEhIQEA8PDQ0MCwphAQIAoAwLfhYUFBIREA8NCwkIBQQBAQQFCAkLDQ8QERIUFBYWFxcWFhQUEhE
 QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBgcICQoLDAwNDg4PFqf/AAIA/cD+gIMCAQE
 CAwMEBQUFBwcHCAkJCQoLCwsMDawNDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAAA
 AABIA3gABAAAAAAAEAAAAABAAAAABABsAAQABAAAAAAACAACAHAAABAAAAAADABsAIwABAAA
 AAAEABsAPgABAAAAAAFAAsAWQABAAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqwA
 DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
 BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
 0ZXJpYWxfRG1hZ3JhbUJ1aWxkZXJSZWdlbGZyTmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
 3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lvbiAxLjBOZXcgTWF0ZXJpYWxfRG1hZ3JhbUJ1
 1aWxkZXJGbz250IGdlbmVyYXRlZCB1c2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
 jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
 AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQB
 nAHIAAYQBtAEIAdQBpAGwAZABlAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQBnAHIA
 AYQBtAEIAdQBpAGwAZABlAHIAVgBlAHIAcWBPAG8AbgAgADEALgAwAE4AZQB3ACAATQBhAHQAZQBy
 yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIABnAGUAbgBlAHIA
 AYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
 TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
 AAAoAAAAAAACAAAAAAACGABgEDAQQBBQEGAQCBCAEJAQoBCwEMAQ0BDgEPARA
 BEQESARMBFAEVARYBFwEYARKBGgEbARwBHQEeAR8BIAEhASIBIwEkASUBJgEnASgBKQAHWm9vbU1
 uTQhab29tT3V0TQpVbmRlcXpmbmVNB1Byaw50TQROZXdnbnVhdmVNB0V4cG9ydE0FQm9sZE0LT3B
 1bkZvbGR1ck0HRGVsZXRLTQhSZWZyZXNoTQdJdGFsaWNNB1pvb21JbkYlWm9vbU9ldEYGUHJpbnR
 GBE5ld0YFU2F2ZUYHRXhwb3J0RgVCb2xkRgtPcGVuRm9sZGVyRgdEZWxldGVVCFJlZnJlc2hGC1V
 uZGVybGluZUYHSXRhbGljRgdab29tSW5CCFpzb21PdXRCClVuZGVybGluZUIGUHVpbnRCBE5ld0I
 FU2F2ZUIGUHVpbnR3J0QgVCb2xkQgtPcGVuRm9sZGVyQgdEZWxldGVVCFJlZnJlc2hCB0l0YWxpY0I
 KRmxvd1NoYXB1c25uZW50b3JlQmFzaWNTaGFwZXMAAAAAA==) format('trueType');

```

    font-weight: normal;
    font-style: normal;
  }
  .e-ddb-icons {
    font-family: 'e-ddb-icons';
    speak: none;
    font-size: 16px;
    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-basic::before {
    content: "\e726";
  }
  .e-flow::before {
    content: "\e724";
  }
  .e-connector::before {
    content: "\e725";
  }
</style>
</script>

```

```

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

```

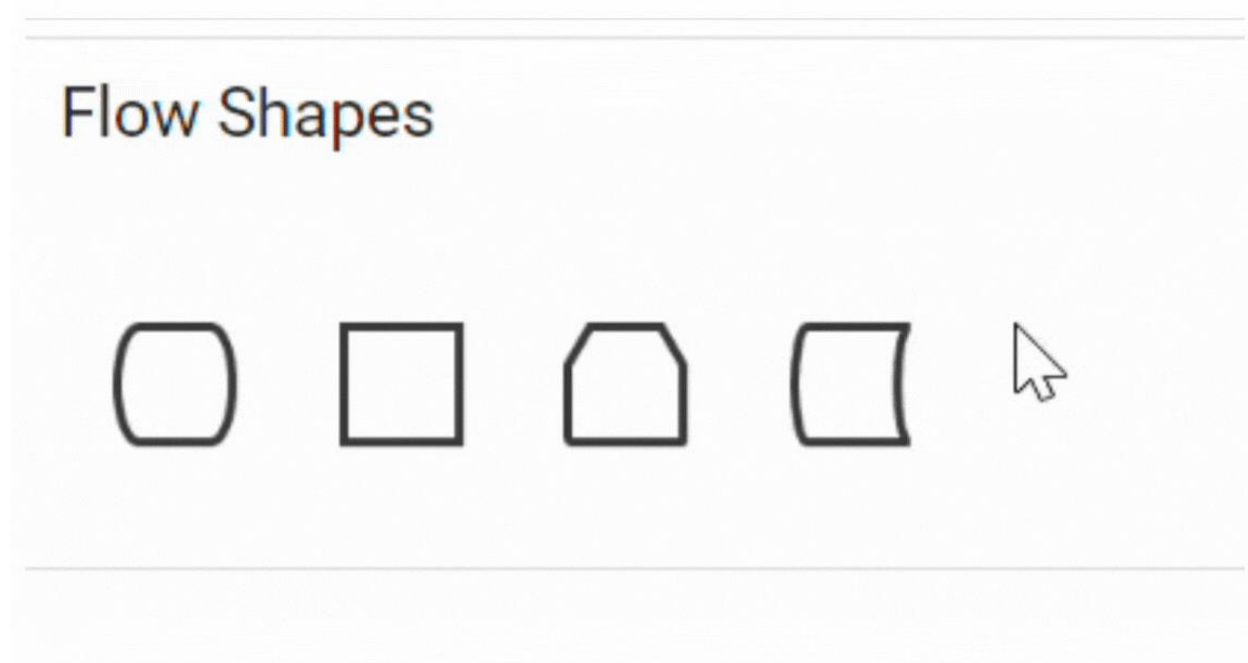
Tooltip for symbols in symbol palette

The Symbol palette supports displaying tooltips when mouse hovers over the symbols. You can customize the tooltip for each symbol in the symbol palette.

Default tooltip for symbols

When hovering over symbols in the symbol palette, the default tooltip displays the symbol's ID.

Refer to the image below for an illustration of the tooltip behavior in the symbol palette.



Custom tooltip for symbols

To customize the tooltips for symbols in the symbol palette, assign a custom tooltip to the 'Tooltip' content property of each symbol. Once you define the custom tooltip, enable the Tooltip constraints for each symbol, ensuring that the tooltips are displayed when users hover over them.

Here, the code provided below demonstrates how to define tooltip content to symbols within a symbol palette.

INDEX.JS

```

/**
 * Default symbol palette sample
 */
//Initialize the basicshapes for the symbol palatte
var basicShapes = [{

```



```

        id: 'Rectangle',
        shape: {
            type: 'Basic',
            shape: 'Rectangle'
        },
        tooltip:{
            content:"Rectangle Tooltip",
        }
    },
    {
        id: 'Ellipse',
        shape: {
            type: 'Basic',
            shape: 'Ellipse'
        }
    },
    {
        id: 'Hexagon',
        shape: {
            type: 'Basic',
            shape: 'Hexagon'
        },
        tooltip: {
            content: 'Hexagon Tooltip',
        },
        //customized content of the Tooltip is enabled by Node Tooltip
        Cosnstraints
        constraints: ej.diagrams.NodeConstraints.Default |
ej.diagrams.NodeConstraints.Tooltip
    }
];
//Initializes the symbol palette
var palette = new ej.diagrams.SymbolPalette({
    expandMode: 'Multiple',
    palettes: [{
        id: 'basic',
        expanded: true,
        symbols: basicShapes,
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic'
    }],
    symbolHeight: 80,
    symbolWidth: 80,
    symbolPreview: {
        height: 100,
        width: 100,
        offset: {
            x: 0.5,
            y: 0.5
        }
    },
    symbolMargin: {
        left: 12,
        right: 12,
        top: 12,
        bottom: 12
    },
},

```

```

    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>
<style>
    @font-face {
        font-family: 'e-ddb-icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShgAAAEoAAAAVmNtYXDon+lDAAACIAAAAIJnbHlmw/g
RIAAAavgAACw0aGVhZBGJTLcAAADQAAANmhoZWEIXQpAAAArAAAACRobXR4oAAAAAAAYAAAC
gbG9jYdYyye4AAAKkAAAAUmlheHABOAd4AAABCAAAACBuYWlldAwInAAALywAAAMVcG9zdNAiwIs
AADJEAABuQABAAAEAAAAFwEAAAAAAEAAABAAAAAAAAAAAAAAAAAKAABAAAAAQAAJo24vV8
PPPUACwQAAAAAANclg90AAAAA1zWD3QAAAAEAAQAAAAACAACAAAAAAAAAAEAAAAoAOwABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnJgQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAA
UAAQAbgAAAAQABAABADnJv//AADnAP//AAAAQAEAAAAAQACAAMABAAFAAYABwAIAAkACgALAAw

```

ADQAOAA8AEAARABIAEwAUABUAFgAXABgAGQAAABsAHAAdAB4AHwAgACEAIgAjACQAJQAmACcAAAA
AAAABBAICAn4CxcgLeAyYDeAQUBHAEoAWEBZwGkgd8B+YH/ghMCMIJAnaClYLMauqC7gMpg2ODmQ
Owg8aD9IQoBF6ElYTRhRGFIQUwBVMFhoAAAAADAAAAAPOA84ACwBnAOsAAAEjFTMVMzUzNSM1IwU
VDxQrAS8VPxYfFQUVHx07AT8LFxUXNycjJz8ONS8fDx4Ban19P319PwEZAQICAwMECQwNEBESFBY
WDAsMDQwNDQwNDQwMDAsXFRQTEQ8NDakEBAMCAQEBAQEBAgMEBAkMDQ8RExQVfwsMDAwNDQwNDQw
NDAsMFhYUEhEQDQwJBAMDAgIB/a8BAwMEBAYGBwgICQoKCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4
PDg4NDhoZGBp6XfoYegkICQcIBgYGBQQAeAwMCAQEBAgMEBQUGBwgICQoKCwMDA0ODg4PDxAPERA
RERESERESEBEQEBApDw4ODQ0NDAsLCgoJCAgHBgYEBAMDAQKWP319P32cDQ0MDA0LDBYWFBIrDw4
LCgQDAwICAQECAgMDBAOLdg8REhQWfGwLDQwMDQ0NDA0MDAwLFxUUEXEPDQwKawQDAgEBAQEBAQI
DBAMKDA0PERMUFRCLDawMDQwNEhERERAREA8PDw4ODg0MDAwLCgoJCAgHBgYUFBAMCAgECawMDBQ
FBw0QEhMy+176EwsLDawNDQ4ODg8ODw8PEA8REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAI
BAQEBAgQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIAAwAAAAADzgPOAAMAXWdjAAATITUhbRU
PFCsBLxU/Fh8VBR8eOwE/CxcVFzcnIyc/Dj0BLx4PHu0BOP7IAZYBAgIDAwQKCw4PERIUfHYMCw0
MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCGwNDxETFBUXCwwMDA0MDQ0NDawNCww
WFhQSEQ8OCwoEawMCAgH9rgEBagQDBQYGBwcJCQkLCwsMDQ0NDg4PDxAQEBEQERIRDw8PDw4PDg4
NDhoZGBp6XvOYewkJCAgHBwYFBQUDawMCAQICAwQFBQYHCAgJCgoLDawMDQ4ODg8PDxAREBERERI
REhEQERAQEA8PDg4NDQ0MCwsLCQkJBwcGBgUDBAIBAlc/Hw0NDawNCwwWFhQSEQ8OCwoEawMCAgE
BAgIDAwQKCw4PERIUfHYMCw0MDA0NDQwNDawMCxcVFBMRDw0MCgMEAwIBAQEBAQECAwQDCGwNDxET
FBUXCwwMDA0MDRIREREQERAPDw8ODg4NDawMCwoKCQgIBwYFBQQDAgIBAgMDAwUFBQcNEBITMvp
e+hMLCwwMDQ0ODg4PDg8PDxAPERIREBEQEBApDw4ODQ0NDAsLCwkJCQcHBgYFAwQCAQEBAQIEAwU
GBgcHCQkJCwsLDA0NDQ4ODw8QEBAREBESAAAAAIAAAAAA3cD1AADAGkAADchNSETFR8dOwE/HTU
RIxEPDy8PayOJAu79Ej8BAgMDBQQGBgcICakJCgoLCwwMDQ0ODQ8ODw8PEBAQEBAQDw8PDg8NDg0
NDawLCwoKCQkICAcGBgQFAwMCAXwCAwUHCAoLDQ4OEBARERESEhERERAQDg4NCwUJCAYEAgF8K30
BdxAQDxAPDw4ODg4NDA0LDAsKCgkJCAgGBwUFBAQDAgEBAgMEBAUFBwYICakJCgoLDAsNDA0ODg4
ODw8QDxAQAbb+ShQTExERDw4OCwsJBwYFAgEBagUGBwkLCw0PBxAREhMUAcAAAAAABAAAAAAD9AO
1AAMABwAvADMAAAEVITU1FSM1IREzFSE1MxEvDyEPDjchNSECVp6IAjN9/RK8AnC8AQIDBAUGBwg
JCgoLDAsNDf0SDQwMDAsKCgkJBwYFBAMCuwJw/ZABg7u7u3x8/si8vAE4DQ0MCwsKCgkIBwYGBAM
CAQECAwQGBgcICQoKCwwMDK+8AAAAAQAAAAADdwN3AAsAAAEhFSERMxEhNSERIwHC/scBOXwBof7
HfAI+fP7HAT18ATkABAAAAAADdwN3AAMABwALADIAACUzNSMBFSM1IxUhNSMRfZMRIRE7AT8HNRE
1LwcjISMPBwGDFX0BtT4+/kp9ft4BeHwFBAoLCgkHBQICBQcJCgsKBAX9kAUECgsKCQcFAsi7AbU
+Pvr6/c59AtN+XwIFBwkKCwoEBQJwBQKQcwoJBwUAgUHCQoLCgQAAAAAAGAAAAADtQP0ADcAPgA
AExEfctMhMz8JES8JKwEVMxEhETM1KwEPCDczETMRMydKAQEBBQcICgsGBWY7gYHBgsKCACFAQE
BAQEBBQcICgsGBWz9Pv2QPn0GBwYLCgghBQEB+X58fRwCvP20BgYGCwoJBgUCAQECEBQYJCgsGBgY
CcgYGBgsKCQYFAgF9/gwB9H0BAgUGCQoLBgZ2/ooBdrwAAAAADAAAAAAMoA3cAIGBFAIUAAAEfDw8
OKwE1EzmfDR0BDw4jNQMhPw8vDz8MLw8hAi8KCQkJCAcIBgYGBAQEAQEBAQEBAQEBAgYGCACJCAk
JCpx9CQoJCAgIBwcGBQEAwMBAQMDBAUFBgCHCAgICQoJfbwBhxQVExMRERAODQwKCQcFAwEBAQM
EBAYGCAgJCQsLCwwNExAPBgYFBQQDAwIBAQEBCACICgWNDxASEhQVFRb+nQHCAQEDAwQEBgYHBwg
ICakKCQoJCQkICAcHBgYUFBAMCArWBOAICAwQFBQYHBwgICQkJCgkKCQgJBwgGBgYEBAMDAQG8/Y8
BAwUHCQoLDg4QEBITEQVDw8ODg4NDQwLCwsJCQgIBg8PEggKCQoKCQsKCgoLFhYUFBMREA8NDAo
JBgQDAACAAAAAAP0A5YAAwBJAAABESERJxEfDjMhMz8OES8OIyEnKwEPDQn3/RJ9AQIDBAUGCAg
JCQoLDawMDQLuDQwMDAsKCQkICAYFBAMCAQECAwQFBggICQkKCwwMDA3+iX36DQwMDAsKCQkICAY
FBAMCApz+SwG1ff3ODQwMDAsKCgkIBwYFBQMCAgMFBQYHCAkKCgsMDAwNAbUNDawMCwoKCQgHBgU
FAwJ9AgMFBQYHCAkKCgsMDAwAAgAAAAADdw01ABkAIQAANxUfCSE/CTURITcjFSE1IzUjyAEBBQc
ICgsGBwYB9AYHBgsKCACFAQH9kLv6Au76+okGBwYLCgghBQEBQEBAQEHCAoLBgcGAj07fx0/AAA
AAQAAAAADdwN3ANEAAABMhJz8LowEfHR0BDx0jLw8jHx47AT8dPQEvHSMPDyeJATmKCxYXGQwNDQ0
NDg0ODg8ODg4ODQ0NDA0LDAsKCwkKCAkIBwcGBQYUFBAMCAgEBagIDBAUFbQYHBwgJCAoJCwoLDAs
NDA0NDQ4ODg4PGBgXFxYUFBMSEA8NDAsIB14EBAQEFGCHCAgJCQoLCwsMDA0ODQ4PDw8PEBAREBE
SERMTEXiSEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQAeAgICAgQEBQYHBwgKCQsLDawNDg4PDxAQEBE
SEhISExMTEXiSExESERERE8QDg8NDXECPOoJEQ8NBQUFAwQCAgEBAgIEAwUFBQcGCACJCQkKCgo
LDawMDA0NDQ4ODg8ODw4ODg4NDQ0MDQsMCwoLCQoICQgHBwYFBQQAeAwICAQEDBQcJCwwODxESExU
VFhCQEBAPDw8PDg4ODQwNCwwKCwkKCAgIBwYFBQQAeAgICAgIEBAUGBwcICgkLCwwMDQ4ODw8QEBA
REhISEhMTExMTEXiSEhIREBAQDw8ODg0MDAsLCQoIBwcGBQQAeAgIBAQIEBAUHBggJCQoLCwwNcQA
AAQAAAAADdwN3AAsAAAEzAyMVITUjEzM1IQGDoeS3AfSh5Lf+DAL6/gx9fQH0fQAAAwAAAAADvAO
8AAsAbADWAAABIXUzFTM1MzUjNSM3Hw8dAQ8VKwEvFDUnNzU/FDsBHwUnDxIdAR8WPwCBHwI7AT8
FPQEvAgE/By8WKwEPAQFZb284b284fQwKFRMSEA4NCgUEAwMCAgEBagIDAwQFCg00EBITFRYLDaw
MDAwNDQ0MDQwMDAwLFhUTEREODAsFBAMDAgIBAQICAwMEBQsMDhERExUWCwwMDAwNDA0NDQwMDAw
MpxMTEhERDxAODQ0LCwkICAYEBAICBAQGBwkJCwsNDQ4PEBEREhMTFBQUFRsaGhkYGBYVAVUEBQU

GBQUFBAQCAgICBP6sEA4MCggGAwIBAgMFBgcJCQoMDA4ODxARERISFBMVFBVFBQCPzhvbzhvWwU
GDA4QEhMVFGsMDAwMDQwNDQwNDawMDAsWFRMSEA4MCwUEAwMCAgEBAgIDAQFCwwOEBITFRYLDaw
MDA0MDQ0MDQwMDAwLfhUTEhAODAsFBAMDAgIBAQICAwMEPAYICakLCw0NDhAPERESExMUFbQVFRQ
VExQSEhEREASODGwMCgkJBwYFAwIBAgMGCAoMDhD+rAQCAgICBAQFBQUGBQUEAVUVFhgYGRoaGxU
UFBQTEExIREQ8QDg0NCwsJCAgGBAQCAgQAAAAAAwAAAAADuQ08AAMAYQDLAAATITUhNx8OHQEPFSs
BLxU9AT8UHwYnDxMVHxY/BwEfAjsBPwU9AS8CAT8HLxYrAQ8B7AEW/urtDBUTEExAPDgsKBAMDAgE
BAQICAwMEBQsMDxASExQWDAsMDA0MDQwNDQwMDAwMCxYUExIQDgwLBAQEAgICAQECAgMEBAoLDg8
REhQVfWwMDAwMDRkNDA0MCwymExMREhAQDw4ODQsLCQgIBgUDAgECBAQGBwgKCGsNDQ4PEBAREhM
TEExQVFRoaGhkZFxYWAVEEBQUBgUEBQMDAgICBP6vEA4NCggGAwIBAgMFBgcICQoMDA0PDw8RERI
SExQUFBVFBQCbzfLBgsODxESFBYWDawMDAwNDQwNDA0MCwwLfhUTERAODQoFBAMDAgEBAQICAwM
EBQsMDxASExQWDAsMDA0MDA0NDQwMDAwMFhUUEhEPDQwJBAMDAgIBAQEBAgMEBD0GBwgJCwsMDg4
PEBAREhIUExQVFBVFBMTExIREQ8QDg0NDAoKCAcGBQCAQEEBQgKDA4Q/qSEAgICAgQEBQUBQY
EBQFVFRYYGBkZGhsVFBQUEXMSEREPDw8NDQsLCQkHBgUDAwIEAAAABQAAAAADvA08AAMAIwArAC8
ASgAAARUHNscPAh0BHwU7AT8FPQEVBSSBDwE1ESM1IRUjEQERIREDKwEPbHEzFSE1MxEvBiMRIQK
n/rKeBAICAgIEBAUFbQYFBQQEAgICAgQEBQUGBQUFAsan/kSnAiz+sJenBgoKCQgGBALeAbzeAgQ
GCAkKC6z+RAFZ3t6fBAUFbQYFBQQEAgICAgQEBQUGBQUFBAQCAgICPP6yp6cBTgFN/uoBFv7qAgU
GBwkKC/52b28BigsKCQgFBQIBTQAAAAABAAAAA0A8A7wACwAAASEVIREzESE1IREjAeT+YAGgOAG
g/mA4Ahw4/mABoDgBoAAEAAAAAA0A8A7wABwALABgAMwAAARUjNSMVIzUBESERIXehETMRIxehESM
nESMRfYyE/BhEvBiEPBgJvpzc4Ab391DcCmjje/ntSVTdvAtgKCgkIBgQCAgQGCakKCvzwCwoKCac
FAwFZ3qen3gIs/rMBTf57AYX89gEW/upVarX9Lm8CBAYICQoKAXYKCGkIBgQCAQMFBwgKCGAAAAA
DAAAAA0A8A5EABwAyAGAAADchNQcVIREjBQc1Iw8OPxUzNQcrAQ8WFT8PFQkBRAKwOv3DOQMnsU8
XFhYWFhUWFRUVFBQUEXMFbGcJCgoMDA4OEBAREhITGRgWfXcXNDODRsbGhkYGBcWFBQTEREPDgw
LCQgEBQMCFBWUfHgYGRkaGhsbGxwcHQE7/sVvrDo5AgRWsVsCagIEBAYGBggICQoLCwwUFBMTExE
REQ8PDg0MCwkJCgcEAWIBWYIDBQYICQsNDQ8RERMUFUXGBgZDRobG0cTEExIQEA4NDAoJCAYFBAI
BrAE7ATsAAAAAaaaaAvDhAAiAEUakAAAAATmFDROBDw4jNRMfDw8OKwE1AzsBPxU1Lw41Pw81Lw4
jAckSERAPDgWLCgkIBgYEAwICAwQFBgcICGoLDA0ODxBjXhAPDg4MCwkJCACGBAQDAQEBAgMEBQc
HCQsKDA0ODhAQVG/tDhsaGRgWFRQTCagHBwYGBQQEAWMCAQECAUGCAoKDA0ODw8REhIPDg4NDAs
KCQkHBgUEAwEBAgQGCALDhAREhQVFXga9wHIAQIDBAUFbwcICQoLCw0NDQwLCwoJCQgHBgUEBAI
BAAd4BTgEBAgMDBAUGBwcJCQkLCwwPDQwMCwoJCQcHBQQEAgLe/WUCBAYICQwNEAgICQkKCQoLCgs
LCwwZExMSEBAPDg0MCgoIBwUEAwMFBwcICQoLDAwNDg4PDxAQChMSERAPDg0NCgoHBgUDAgAAAwA
AAAAD9AN3AAMAHwBUAAABAYETJzMfDCEVIQ8HAXEnDwYRIRM/Aj0BLwgjNS8IJS8MIw8BA7a8/WS
8JAgHBgYLCgoVBQ0OEAKKAXL+IAkJCAChBwUfLhkFCgkGBQIBAXXMAwICAQIFBgkKcWYGHAEbBQc
ICgsGB/6LBwYGCwoKFQUNDhAJCr0GBgI+/okBd/oBAQIFBwcQAwcGBAIBfQEBAwQFBgcI/tMCCzo
CBwkKCwYG/UoBmgCHBwcGBgYLCgkGBQIBgwcGCwoIBwUBAQEBAQIFBwcQAwcGBAIBAQIAAAAABgA
AAAADAQ08AAMABwALAB8AIwBeAAALMxEjAzMRIwMzESM1EQ8HIS8GNRELFSM1Jw8FFSMVMxEfDjM
hMz8OETM1IzUvBiMHAlM4OG84OG84OAGFAQEDAwUEBQb+RAYFBAUDAwIBTaYWBQkHBgQD3jcBAQI
DAwUEBgYGBwcICAgJAbwJCAgIBwCGBgYEBQMDAgEBN94DBAYHCQoLrAzqAb3+QwG9/kMBwV/9gQY
FBAUDAwEBAQEDAwUEBQYCF284ODMCBggJCgo+N/2BCQgICAcHBgYGBAQEAwIBAQIDBAQFBQYGBwc
ICAgJAn83PgSKAgGBAIBAAABAAAAA0A8A7wAXgAAAQ8MNSMVMzUjPw8fFw8XLx4HHx4zPxcvFyM
PAQGDg4cGhoZFxcVFBMQEDfegQ0OEBITFBWUGBgZGhsbGxwaGhoZGRcXFhUUFBIREA4ODAoJCAY
FAGEBAGUGCAkKDA4OEBESFBQVfHcXGQwaGRsdEBAQEASPDw8PDg4ODQ0MDAwLCwsKChIIBwcHBgU
ENgUGBwcICQkKCwsLDA0NDQ4PDg8QEBAREREREhISEhITHh4dHRwbGhkZFxYUFBIRDw4MCgkHBAM
BAQMFBgkLDA0PERIUFBYXGRkaGxwdHR4eHh4dA60FBASMDhARExQWGBgad984GRcXFRQSEQ8ODAo
JBgUDAQECEBQYHCQsMDQ8QERITFRUWfXcZGRkaGxobGRkYGBcWFRQTEExEQDg4MCgkIAwUEAgEBAQI
DBAQFBgYGBwgICQkKCgoMCwwMGg4ODg8PDw8OEhIREBEQDw8PDg4NDQwLCwsKCQkIBwcHBQUEAwM
CAQEDBACJCwwNDxESExUWFxkZGhsCHR0eHh4eHR0cGxoZGRcWFBQSEQ8ODAoJBwQDAQMFAAAAAGa
AAAADFQ08AAMAAaAAANYE1IREfHjsBPx4RIxEPDiMvDgMj6gIs/dQBAQEDAwMFBQYGBggHCAkJCgo
KCwsMDA0MDQ0NDg0PDg4ODg4NDQ0NDQwLDAoLCgkKCAkHBwcGBgUEBAMDAQEBOAIFBgkLDA0PEBI
TFBUWfHcWFhQVEExERDw0MCgkHBAIBN0Q3AU0ODg4ODQ0NDQwMDAsLCwoJCQkICAcHBgYFBAQDAgI
BAQICAwQEBQYGBwcICAKJCQoLCwsMDAwNDQ0NDg4ODgH0/gEWfHUUExERDw0MCwgHBAMDBACICww
NDxERExQVfHYB/wAAAAEAAAAAArEDvAADAALMwEjAU86ASg6RAN4AAADAAAAAAQA5AACwBMANM
AAAEjFTMVMzUzNSM1IzcfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwcvHisBDwUBnGrkZGR
kZL8HBw0LCQcFAwEBAwUHCQsNDhERERMUFBUWFRUVExMSERAPDAsJBwUDAQEDBQcJCwwPEBESExM
VFRUWFRUTEExER/vUPDw8NDgwMDAsLCgkJCAChBwUFAwMCAgICAwMFBQcHBwgJCQoLCwsNDA4NDw4
QEBAQEBEQEREbGRkYGBcWfQoEBQYFBgYNDAUFCgkHAWEDAwEDB6kODAsIBwQDAQEBAgMEBAYGBwc
ICQoJCwsMDAwODQ8PDxAQEBARERASERARERAQEAJkZGRkZGQOCakRERMTFRUWFRUVExMREREODQs
JBwUDAQEDBQcJCw0OERERExMVFRUWFRUTEExEREQ4NCwkHBQMBAMQMBwLkDZEHBwgJCQoLCwsNDA4

NDw8PEBAQEBEQERESEBEREBAQEAE8PDw0ODA0LCwsKCQkIBwCHBQUDAwICAQMEBwgLDA6pBAMCAgI
BAgIDBwkKBQUMDQwFBQqqFhYXGBgZGRsRERAREBAQEAE8PDw0ODA0LCwsKCQkIBwCHBQUDAwICAgI
DAwUFAAMAAAAA5ADkAADAEQAYwAAASE1ISUfCA8PLw8/Dx8GJQ8WHQEfHTM/Bx8GMz8INS8EPwc
vHisBDwUBOAEs/tQBIwcHDQsJBwUDAQEDBQcJCw0OERERExQUFRYVFRUTExIREA8MCwkHBQMBAQM
FBwkLDA8QERITExUVFRYVFRMTERH+9Q8PDw0ODAwMCwsKCQkIBwCHBQUDAwICAgIDAUFBwCHCAk
JCgsLCw0MDg0PDhAQEBAQERARERsZGRgYfXyWqgQFBgUGBg0MBQUKQCcDAQMDAQMHqQ4MCwgHBAM
BAQECaWQEBgYHBwgJCgkLCwwMDA4NDw8PEBAQEBEREIREBEREBAQAgBkcggJERETExUVFhUVFRM
TERERDg0LCQcFAwEBAwUHCQsNDhERERMTFRUVFhUVExMREREODQsJBwUDAQEDBQcJCw2RBwCICQk
KCwsLDQwODQ8PDxAQEBAEREIREARERAQEBAPDw8NDgWNCwsLCgkJCACHBwUFAwMCAGEDBACICwW
OqQQDAgICAQICAwCJCgUFDA0MBQUKqhYWFxgYGRkbEREQERAQEBAPDw8NDgWNCwsLCgkJCACHBwU
FAwMCAGICAwMFBQAAAgAAAAADkAOQABsAtgAANw8CFR8FIT8FNS8FIQ8BARC7AR8KDxArAS8WPwg
nNw8BJyMfCRUfGj8WLwM1PwUzPwMvAQcjJyN1AgIBAQICAgMDAwYDAwICAQEBAGICAwP8+gMDAg8
HOgUFBGkJAwQDAgULAQEDBAIFBwCLCw8SDA0OGBgZGwsMDAsMCwwLCA4HBgUEAwMDAqEHAQM
DAwQECg0pHwEBpCyCJAImGg4MBQUCAwMCAGMFBAGQFBgYHCAgKCgsMDQ4PEBASEhMTFRU1IhEPDw8
bGAWLCwoSEA0LBgYHBQIDAQEIawEBAGQBBiIKCwsMAGMKOCN1LM4CAwNJAwMCAGIBAQICAgMDSQM
DAgICAQECaPMBAgIFCAMJCw89fVYjHhgLDw8OEwwNDAGBQYFAwECAwMEBQYECwYGBg8KDAwNDQ4
PEJKxIAGFAgIEAQIDJgcEAQYuAwMEBAQFBBEL4jgfgHooDg0MDAsKCgkICQcIBGcFBQQEAgIBAQE
EAgMEBAkKBGcHBw8QEBENDxoYESUqMLYYFRAFBUBAQcCagIQGwEFBQAEAAAAAQA5AAAwAjACc
ARQAAARUHNscfAh0BDwYvBj0BPwU7AR8BJRUhNQcrAQ8IETMVITUzES8HIzUhApb+1GsDAgICAgM
EBAUFBQQFAwQCAgICBAMFBAUFBQQBm/7UZDIYcQ0HBgUEAwIBlgH0lgEBBQUGCAkKaf4MAZzIyKg
EBAUFBQQEBAMDAQEBAQMDBAQEBQUFBAQDAgIBA+WWlpYBBQQFBgYHCAj+opaWAV4HCAsGBwUEAvo
AAAEAAAAA48DkABEAAABDwMVIw8GFR8GMxUfBjM/BjUzPwY1LwYjNS8GIw8CAawDBwQC+QsKCQg
HBAICBACICQoL+QIEBwgJCgtjCgoJCACeAvkLCgkIBwQCAgQHCAkKC/kCBACICQoKXgsKCgOABQk
KCvoCBACICQoLYwoKCQgHBAL5CwoJCACeAgIEBwgJCgv5AgQHCAkKC2MKCgkIBwQC+goKCQgHBAI
BAwUAAAAABQAAAAADwgPCAAMABwAJAFUAmwAAARUHNQEVIZUHNSMVHw8hPw81FxEjNS8PIQ8PFsM
RNQ8PER8PIT8PETUvDzECyP5wASyWlmQBAQIEBAUGBgICakJCgoKASwKCgoJCQgIBwYGBQQDAwE
BljIBAQMDBAUGBgICakJCgoK/nAKCgoJCQgIBwYGBQQDAwEBMgoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKArwKCgoJCQgIBwYGBQQEAgEBAGIDBAQGBp8HBwCICAgJCgFqYmgB9MjIyMj
ICgoKCQkICACGBgUEAwMBAQEBAwMEBQYGBwgICQkKCgq+oP3uyAoKCgkJCAGHBgYFBAMDAQEBAQM
DBAUGBgICakJCgoKyAK8ZAEBAGQEBQYGBwgICQkKCgr9RAoKCgkJCAGHBgYFBAAQCEBAQIEBAU
GBgICakJCgoKAhIKCQkICQgHCKkHBQUFBAwMCAGAAAAAQA5AAAbQcXAAABHwPCC8IPQE
PFhUfAQ8ELw4/Fz0BPwgAiUPBxEfDyE/DxEvDyEPBgJ7uAQDAgEBAGMEuAUFBGcGawGFAwMCAGe
jHxsYCwoJCQgIBGcGBgYFBAMDAgIBAQIFAQIEBgQDBAMDChMRDQsIAwMBAQECAwIHBQUGBwgKCgw
NDw8REhQWGBocHB8BAGIDAUFBwCGBQX+JgoJCAYFAwIBAQIDBQYICQoLDAwNDg4PDwH0Dw8ODgw
NDAsKCQgGBQMCAQECAwUGCAkKCwwNDA4ODw/+DA8PDg4NDawDM7gFBQYHBwYFBbgEawIBAQEDAwM
EBAUEBlMBAGQFBAMEBQUGBgICQoLDA0ODxAREhIpLwUFAwIBAQECAG8cHBsaGgwNDawbHRsOHw8
PDQ0NDA0MDAsJCQgHBgYEAwIBUwUFBQQDBAMCAGEBAGMtCwwNDQ0ODw/+DA8PDg0NDQwLCgkIBgU
DAgEBAGMFBggJCgsMDQ0NDg8PafQPDw4NDQ0MCwoJCAYFAwIBAQIDBQYICQAAwAAAAADbgOPADE
AVgC4AAABMx8TFQ8PLwYTPwITHwsPDy8BAz8BMx8BJyMHHwkTDwg3Fz8VLw8/Di8TAhEKfHcLCgk
JCQkJCAkHCAUEBAMCAGEBAGQFBwgKDA0OEBITFRYYERITEwEDBAEEERdUDw4ODQ0LCQgHBQMBAQM
EBGcJCgwODg4QEBIUFCAZBBQiHhEQ2Q+iAioZEwGAQECCBQCBQMDAwUaRQHxyRcXfHwFRUUEX
QBw4MCwkDBAICAgEBAwQGBwkLDQ0PEBAREhMTDSCTfQkIBgYFBQQEAWEBAGMEBggJCwsNDQ8PERA
RERIREkECBwMFAwMEBQYGBwkJCgsJCgoLDQ0NDxUUEhEQDg0MCgkHBgUDAgEBAwUIAhAyAQQBawE
BSwQFBggICgsNDhAQEHUTEhAODQsJBwCFBAMCAQEBAwEUAWQBazUGKwQEBAMEAgILVv4rIR4ICAc
BCA0xCwICAgMEBgICGoMDQcPERMUCwsMDAwZEXMREBAPDg4MCwsJCACGBQYUCw8IBwCICQoLDAw
MDhMSEhAQDg0MCgoJCACGBQQDAgEBAAAAAAMAAAAA/QDCAaQAFYAUQAAAR8GFQ8MJS8FPQE/CwM
zHwYVHwYhHwYVIQ8IET8GJw8HER8PJT8OPQEvCiM1Lw8hPQEvDiMPBgOVbwUFBAMCAGEBawSaCag
kDAsMCww9wAYFAwMDAQIDBJoICAoMCwwLCjIFCgkIBwYDAgIEBQgICQkBOAoJCACGAwL+bhISEhM
SEA4NhgIEBQJCQ1NCAgFBQQDAQEBAQMEBQICAgKCQsKCwsMAkMSEhMTEQ8NoQYEBQMDAQICAgQ
DBwkKDAwNDmsBAGIEBQYHCAkJCgoKCwwM/uMCAGQFBGcICQkKCgsLCwyoCwwLCgsJCgHfAQEBAGM
DAwUEBQYFvggHBwYFBAIBAQEBAgMDAwUEBQYGVggHBwUFBABAUA8CBAUICakJLAoJCACGAwICBAU
ICakJWQEEBgKCwwNpQHECQkJBwUEAiAJCQoKCgsMDP4KDAwLCgoKCQkIBwYFBAMBAQEBCACJCgw
MxQgIBwgICAgICQkJCQYKQgHBAQBVawMCwoKCgkJCACGBQQDAQEQAawLCgoKCQkIBwYFBAMBAQE
BAwQFBGcAAAAABQAAAAADXgOQACEAQwBLAGkAXQAAAREPBy8HET8HHwYHEQ8HLwCRPwcfBgCRDwc
vBxE/Bx8GNxcjNycHIw8HFR8HMxEVHw0zITM/DTURMz8HNS8HIy8IIw8GApYBAQIDBAQFBQUBAQ
DAgEBAQECAwQEBQUBQQEAWIBfAEBAgMEBAUFBUQEBAMCAQEBAQIDBAQFBQUBAQDAgF8AQECAwQ
EBQUBQQEAWIBAQEBAgMEBAUFBUQEBAMCAbAU1xRCIn0FBQQEAWIBAQEBAgMEBAUFQIBAwMEBAU

```

FBgYHBwcHCAHCCAcHBwcGBgUFBAQDAwECGQUFBAQDAgEBAQECawQEBQWWIqQFBwcICakKvwkKCAg
HBwUCCp68BgQEBAMDAQEBAMDBAQEBgFEBgQEBAMDAQEBAMDBAQEBv68BgQEBAMDAQEBAMDBAQ
EBgFEBgQEBAMDAQEBAMDBAQEBv68BgQEBAMDAQEBAMDBAQEBgFEBgQEBAMDAQEBAMDBAQEzzI
yJFYBAQIDBAQFBRkFBQQEAwIBAf3zCacHBwcGBgUFBAQDAwECAGEDAwQEBQUGBgCHBwcIAg0BAQI
DBAQFBRkFBQQEAwIBAVYICacFBQMCAQECawUFBwgAAAAAAQAAAAADjwOPAOGAAAEpBy8DKwEPBx0
BHwY7Aj8ILwQ/Bx8dDx4vESsBDwUVHxAzPx4vHisBDwUBbIRERAPEA40SAQFBAUEBQoEBAMCAgE
BAgMEBQYGBuofBQqEBAMDBAEBAQECA0sTFBUXGBgZGQ0ODQ0NDA0MGAsLCwoJCQkJBwgHBgYKBQM
DAwEBAQEBAQMDAwUKBgYHCacJCQkJCgSLCwwMDA0MDQ0NDg0PEA8ODw4ODg4NDawMCgSMagQDBAQ
DAkgDAQMPDxARERMTFBQUFRUWFhYWFBUExQTEhMSEhEQEA8ODg0MDAsKCgkICAYGBAMDAQEBQM
DBAYGCAGJCgoLDawNDg4PEBAREhITEhMUExQUFBMTExITEhIDcwcJCQoKCw0MRgMCAGEEAwMEBAQ
FBukGBwUFBQMCAQICawQECgQFBQqEBUsRDgwkCAYEAQEBAQIDBAQFDAYHBwgJCAkKCgSKDAsZDA0
NDQ0NDg0ODQ0NDA0YDAsLCwoJCggJBwgHBgYGBAUDAwMBAQEBAQIDBAUFBggHCQkKcwsOAgIBAQJ
IBQYGBhAQDw4NCwsKCQgGBgQDAQECAgQEBgYICakKCgSMDA0ODg8QEBESEhITExQTFBQUFBQExQ
TExISEhEQEA8ODg0MDAsKCgkICAYGBAQCAgICawQFBgABAAAAAAMKA48AKAAAATMfBBUHCwEPbjc
fAj8CLwE3Ez8GBysBLwEBkAYiGg8HBwMlQwUGBg8QRgl7giwiJgYCYAEIWRkIBATjBgSNGR8gjAN
aAwQDAwMNF/7x/soPDAoHBRItCgEGBAIBGBAPLwGZiiEKBBOYFggBBwAABAAAAAAAEAAQAAAMABwA
LACMAAAEVITUhFSE1ARUhNQmzFSERIXehESM1IRUjESERIXehNTMRIQPA/wD+gP8AAkD+wEDA/sC
AAYDAAoDAAYCA/oDA/kABAMDAwMACwMDA/wCA/wD+wAFawMD+wAFAAQCAUAAAAAAQAAAAEAAQ
AAHYAAAEHIREhLwcPDx8PPw8hETmFdZ8PLw8PBgMSAf7v/u8LCwwNDw8REQ0NDawLCwkKCAcHBQq
DAgEBAgMEBQcHCAoJCwsMDA0NDQ0MDAsLCQoIBwcFBAMCAQFAwAECawQFBwcICgkLCwwMDQ0NDQw
MCwsJCggHBwUEAwIBAQIDBAUHBwgKCQsLDAwNDRERDw8NDAsDwgL9ABAMCgkHBgMBAQIDBAUHBwg
KCQsLDAwNDQ0NDawLCwkKCAcHBQqDAgEBAgMEBQcHCAoJCwsMDA0NAwANDQwMCwsJCggHBwUEAwI
BAQIDBAUHBwgKCQsLDAwNDQ0NDawLCwkKCAcHBQqDAgEBAwYHCQoMAAAAAAQAAAAA/8EAAAwAFc
AbQCrAAABDwEVHxAFAQUVDw8vDz8PHw4DEQ8PJwMjEQMzAyEnHwEzPx09AS8TESEBwgEBAQIDBQY
HCAoKDAwNDw8PEjP92QECakABBAUICQsNDxAREhQUFhYXFxYVFRQSERAPDQsJCAUEAQEEBQgJCw0
PEBESFBUVFhcXfYUFBIREA8NCwkIBQT/FxESEBEPEA4ODQ0LCwsJC1uMtEDS0gMARxUSDw4PDg4
NDg0NDawMCwsKCwkJCQgHBwcFBQEAwMBAgECAGMDBAKMDQ8RExQVFxgZDA0S/QABwgCNDhQUFBM
SEhIQEA8PDQ0MCwphAQIAoAwLFhYUFBIREA8NCwkIBQQAQOFCakLDQ8QERIUFBYWFxcWFhQUEhE
QDw0LCQgFBAEBBAUICQsNDxAREhQUFhYCCf7+AwQFBgcICQoLDawNDg4PFqf/AAIA/cD+gIMCAQE
CAwMEBQUFBwcHCAkJCQoLCwsMDA0NDQ0ODg4PDg8ODQ0ODA0NGBcWFBMSEA4MCggDAwIBQgAAAA
AABIA3gABAAAAAAAEAAABAAAAAABABsAAQABAAAAAAACAAcAHAABAAAAAADABsAIwABAAA
AAAAEABsAPgABAAAAAAAFAsAWQABAAAAAAGABsAZAABAAAAAAAKACwAfwABAAAAAALABIAqWA
DAAEECQAAAAIAvQADAAEECQABADYAvwADAAEECQACAA4A9QADAAEECQADADYBAwADAAEECQAEADY
BOQADAAEECQAFABYBbwADAAEECQAGADYBhQADAAEECQAKAFgBuwADAAEECQALACQCEyBOZXcgTWF
0ZXJpYWxfRGllhZ3JhbUJ1aWxkZXJSZWdlbGFiYmV3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyTmV
3IE1hdGVyaWFsX0RpYWdyYW1CdWlsZGVyVmVyc2lubiAxLjBOZXcgTWF0ZXJpYWxfRGllhZ3JhbUJ
1aWxkZXJGbz250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAE4AZQB3ACAATQBhAHQAZQByAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHU
AaQBsAGQAZQByAFIAZQBnAHUAbABhAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQB
nAHIAIYQBTAEIADQBPAGwAZABlAHIAATgBlAHCAIABNAGEAdABlAHIAaQBhAGwAXwBEAGkAYQBnAH
IAYQBTAEIADQBPAGwAZABlAHIAVgBlAHIAcWBPAG8AbGAgADEALgAwAE4AZQB3ACAATQBhAHQAZQB
yAGkAYQBsAF8ARABpAGEAZwByAGEAbQBCAHUAaQBsAGQAZQByAEYAbwBuAHQAIAABnAGUAbgBlAH
IAYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAEQBuAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAACgBAGEDAQBBQEGAQCBCAEJAQoBCwEMAQ0BDgEPARA
BEQESARMBFAEVARYBFwEYARKBGgEbARwBHQEeAR8BIAEHASIBIwEkASUBJgEnASgBKQAHWm9vbU
1uTQhab29tT3V0TQpVbmRlcmxpbmVNB1ByaW50TGROZXdnbnVhdmVNB0V4cG9yde0FQm9sZE0LT3B
1bkZvbGR1ck0HRGVyZXRLTQhSZWZyZXNoTQdJdGFSaWNNNB1pvb21JbkYIWM9vbU91dEYGUHJpbnR
GBE5ld0YFU2F2ZUYHRXhwb3J0RgVVCB2xkRgtPcGVuRm9sZGVyRgdEZWxlZGVGVCFJlZnJlc2hGClV
uZGVybgGluZUYHSXRhbG1jRgdab29tSW5CCFPvb21PdXRCClVuZGVybgGluZUIGUHVJpbnRCBE5ld0I
FU2F2ZUIHRXhwb3J0QgVVCB2xkQgtPcGVuRm9sZGVyQgdEZWxlZGVCCFJlZnJlc2hCB0l0YWxpY0I
KRmxvd1NoYXBlcwldB25uZWN0b3ILQmFzaWNTaGFwZXMAAAAAA==) format('trueType');

font-weight: normal;
font-style: normal;
}
.e-ddb-icons {
font-family: 'e-ddb-icons';

```

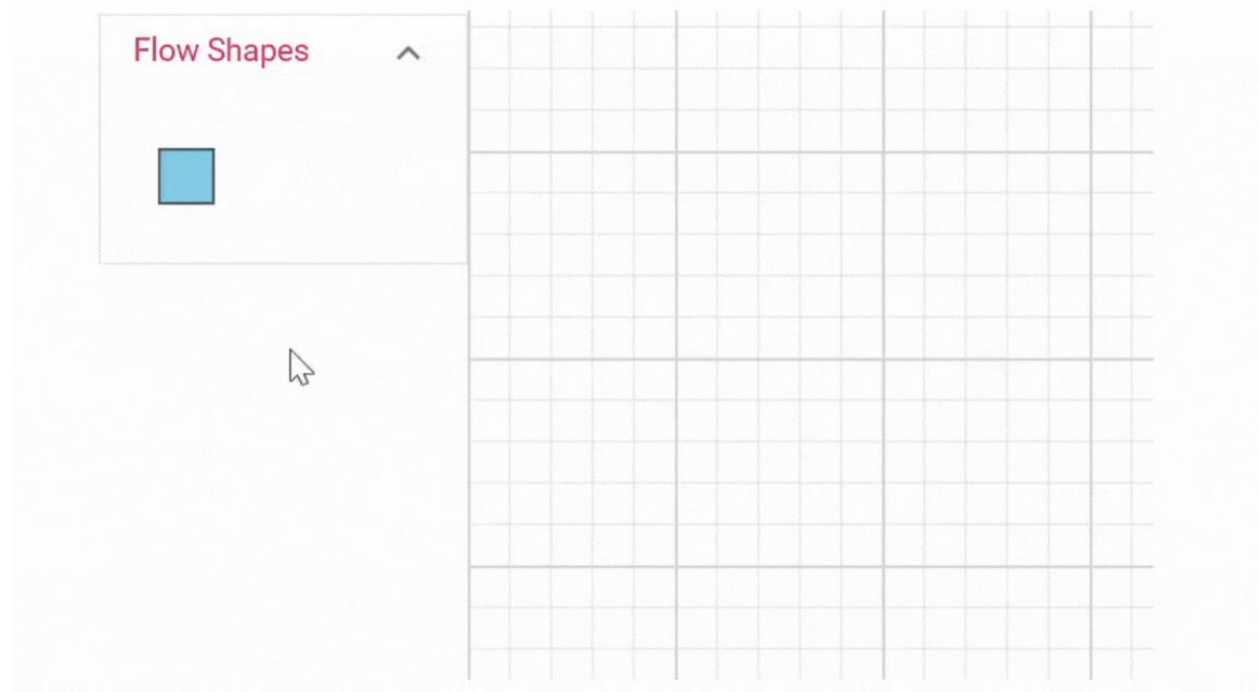
```
    speak: none;
    font-size: 16px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: 1;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }
  .e-basic::before {
    content: "\e726";
  }
  .e-flow::before {
    content: "\e724";
  }
  .e-connector::before {
    content: "\e725";
  }
}
</style>
<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 provide different tooltip for Symbol palette and diagram elements.

Differentiate the tooltips between symbols in the symbol palette and dropped nodes by utilizing the `dragEnter` event. When a custom tooltip is defined for a symbol, it will be displayed for both the symbol and the dropped node in the diagram canvas. However, to provide distinct tooltips for symbols in the palette and dropped nodes, capture the `dragEnter` event and assign specific tooltips dynamically.

When a symbol is dragged from the symbol palette and enters the diagram canvas, the `[DragEnter]` [IDragEnterEventArgs](#) event is triggered. Within this event, you can define a new tooltip for the dropped node. By assigning custom tooltip content to the `Tooltip` property of the node, you can provide a distinct tooltip that is specific to the dropped node.

The following image illustrates the differentiation of tooltips displayed in the Symbol Palette and the Diagram.



The following code snippet will demonstrate how to define two different tooltip for symbol in the symbol palette and dropped node in the diagram canvas.

```
`ts
let diagram: Diagram = new Diagram({
width: '100%', height: '500px',
connectors: connectors, nodes: nodes,
//event to change tooltip content while dragging symbols into Diagram
dragEnter: dragEnter,
});
diagram.appendTo('#diagram');
function dragEnter(args:IDragEnterEventArgs)
{
//enable tooltip constraints for the dragged symbol
args.dragItem.constraints = NodeConstraints.Default | NodeConstraints.Tooltip;
//change the tooltip content of the dragged symbol
args.dragItem.tooltip.content='This is Diagram Tooltip';
}
`
```

Palette interaction

Palette interaction notifies the element enter, leave, and dragging of the symbols into the diagram.

DragEnter

[DragEnter] [IDragEnterEventArgs](#) notifies, when the element enter into the diagram from symbol palette.

DragLeave

[DragLeave] [IDragLeaveEventArgs](#) notifies, when the element leaves from the diagram.

DragOver

[DragOver] [IDragOverEventArgs](#) notifies, when an element drag over another diagram element.

Note: The diagram provides support to cancel the drag and drop operation from the symbol palette to the diagram when the ESC key is pressed

See Also

- [How to add the symbol to the diagram](#)

Overview in EJ2 JavaScript 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.TS

```
import {
```

```

    Diagram,
    ConnectorModel,
    Overview,
    OverviewModel
} from '@syncfusion/ej2-diagrams';
import {
    DataManager,
    Query
} from '@syncfusion/ej2-data';
import {
    TreeInfo,
    Node,
    StackPanel,
    ImageElement,
    Container,
    TextElement,
    DataBinding,
    HierarchicalTree
} from '@syncfusion/ej2-diagrams';
/**
 * Overview
 */
let diagram: Diagram;
let overview: Overview;
Diagram.Inject(DataBinding, HierarchicalTree);
let data: object[] = [{
    'Id': 'parent',
    'Name': 'Maria Anders',
    'Designation': 'Managing Director',
    'IsExpand': true,
    'RatingColor': '#C34444'
},
{
    'Id': 1,
    'Name': 'Ana Trujillo',
    'Designation': 'Project Manager',
    'IsExpand': false,
    'RatingColor': '#68C2DE',
    'ReportingPerson': 'parent'
},
{
    'Id': 2,
    'Name': 'Anto Moreno',
    'Designation': 'Project Lead',
    'IsExpand': false,
    'RatingColor': '#93B85A',
    'ReportingPerson': 'parent'
},
{
    'Id': 3,
    'Name': 'Thomas Hardy',
    'Designation': 'Senior S/w Engg',
    'IsExpand': false,
    'RatingColor': '#68C2DE',
    'ReportingPerson': 1
},
{

```

```

        'Id': 4,
        'Name': 'Christina kaff',
        'Designation': 'S/w Engg',
        'IsExpand': 'false',
        'RatingColor': '#93B85A',
        'ReportingPerson': 2
    },
    {
        'Id': 5,
        'Name': 'Hanna Moos',
        'Designation': 'Project Trainee',
        'IsExpand': 'true',
        'RatingColor': '#D46E89',
        'ReportingPerson': 2
    },
    ],
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
// Initializes the diagram control
diagram = new Diagram({
    snapSettings: {
        constraints: 0
    },
    layout: {
        type: 'OrganizationalChart',
        margin: {
            top: 20
        },
    },
    getLayoutInfo: (node: Node, tree: TreeInfo) => {
        if (!tree.hasSubTree) {
            tree.orientation = 'Vertical';
            tree.type = 'Alternate';
        }
    },
    },
    dataSourceSettings: {
        id: 'Id',
        parentId: 'ReportingPerson',
        dataManager: items
    },
    getNodeDefaults: (node: NodeModel) => {
        node.height = 50;
        node.style.fill = '#6BA5D7';
        node.style.borderColor = 'white';
        node.style.strokeColor = 'white';
        return node;
    },
    getConnectorDefaults: (obj: ConnectorModel): ConnectorModel => {
        obj.style.strokeColor = '#6BA5D7';
        obj.style.fill = '#6BA5D7';
        obj.style.strokeWidth = 2;
        obj.targetDecorator.style.fill = '#6BA5D7';
        obj.targetDecorator.style.strokeColor = '#6BA5D7';
        obj.targetDecorator.shape = 'None';
        obj.type = 'Orthogonal';
        return obj;
    },
    },

```

```
setNodeTemplate: (obj: Node, diagram: Diagram): Container => {
    let content: StackPanel = new StackPanel();
    content.id = obj.id + '_outerstack';
    content.style.strokeColor = 'darkgreen';
    content.style.fill = '#6BA5D7';
    content.orientation = 'Horizontal';
    content.padding = {
        left: 5,
        right: 10,
        top: 5,
        bottom: 5
    };
    let innerStack: StackPanel = new StackPanel();
    innerStack.style.strokeColor = 'none';
    innerStack.style.fill = '#6BA5D7';
    innerStack.margin = {
        left: 5,
        right: 0,
        top: 0,
        bottom: 0
    };
    innerStack.id = obj.id + '_innerstack';
    let text: TextElement = new TextElement();
    text.content = obj.data['Name'];
    text.style.color = 'white';
    text.style.strokeColor = 'none';
    text.style.fill = 'none';
    text.id = obj.id + '_text1';
    let desigText: TextElement = new TextElement();
    desigText.margin = {
        left: 0,
        right: 0,
        top: 5,
        bottom: 0
    };
    desigText.content = obj.data['Designation'];
    desigText.style.color = 'white';
    desigText.style.strokeColor = 'none';
    desigText.style.fill = 'none';
    desigText.style.textWrapping = 'Wrap';
    desigText.id = obj.id + '_desig';
    innerStack.children = [text, desigText];
    content.children = [innerStack];
    return content;
}
});
diagram.appendTo('#element');
// Initializes the overview control
let options: OverviewModel = {};
// Relates diagram with overview
options.sourceID = 'element';
overview = new Overview(options);
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></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 EJ2 JavaScript Diagram control

This article describes the API migration process of Diagram component from Essential JS 1 to Essential JS 2.

Background

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Defines the background color of diagram elements	Property: backgroundColor <code>\$("#diagram").ejDiagram({ backgroundColor: "red"});</code>	Property: backgroundColor <code>var diagram = new ej.diagrams.Diagram({ backgroundColor: 'red'}); diagram.appendTo('#diagram');</code>
Defines how to align the background image over the diagram area	Property: backgroundImage.alignment <code>\$("#diagramcontent").ejDiagram({ backgroundImage: { alignment: ej.datavisualization.Diagram.ImageAlignment.XMidYMid }});</code>	Property: background.align <code>var diagram = new ej.diagrams.Diagram({ pageSettings: { background: { align: 'XMidYMid' } }}); diagram.appendTo('#diagram');</code>
Defines how the background image should be scaled/stretched	Property: backgroundImage.scale <code>\$("#diagramcontent").ejDiagram({ backgroundImage: { scale: ej.datavisualization.Diagram.ScaleConstraints.Meet }});</code>	Property: background.scale <code>var diagram = new ej.diagrams.Diagram({ pageSettings: { background: { scale: 'Meet' } }}); diagram.appendTo('#diagram');</code>
Sets the source path of the background image	Property: backgroundImage.source <code>\$("#diagramcontent").ejDiagram({ backgroundImage: { source: "Syncfusion.png" }});</code>	Property: background.source <code>var diagram = new ej.diagrams.Diagram({ pageSettings: { background: { source: 'Syncfusion.png' } }}); diagram.appendTo('#diagram');</code>

Bridging

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Sets the direction of line bridges	Property: bridgeDirection <code>\$("#diagramcontent").ejDiagram({ bridgeDirection: ej.datavisualization.Diagram.BridgeDirection.Bottom});</code>	Property: bridgeDirection <code>var diagram = new ej.diagrams.Diagram({ bridgeDirection: 'Top'}); diagram.appendTo('#diagram');</code>

CommandManager

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Stores the multiple command names with the corresponding command objects	<pre>Property:commandManager.commands\$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.Shift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length > 0; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } } });</pre>	<pre>Property:commandManager.commands var diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.nodes.length > 0 diagram.selectedItems.connectors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes.length; i++) { diagram.selectedItems.nodes[i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }] } }); diagram.appendTo('#diagram');</pre>
The command is executable at the moment or not.	<pre>Property:commandManager.commands.canExecute \$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.Shift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length > 0; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } } });</pre>	<pre>Property:commandManager.commands.canExecute var diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.nodes.length > 0 diagram.selectedItems.connectors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes.length; i++) { diagram.selectedItems.nodes[i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }] } }); diagram.appendTo('#diagram');</pre>
Defines what	<pre>Property:commandManager.commands.execute \$("#diagramcontent").ejDiagram({</pre>	<pre>Property:commandManager.commands.execute var diagram =</pre>

to be executed when the key combination is recognized	<pre>commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.Shift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length ; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } } } };</pre>	<pre>new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.nodes.length > 0 diagram.selectedItems.connectors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes.length; i++) { diagram.selectedItems.nodes[i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }] });diagram.appendTo('#diagram');</pre>
Defines a combination of keys and key modifiers, on recognition of which the command will be executed	<pre>Property:commandManager.commands.gesture\$\$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.Shift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length ; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } } } };</pre>	<pre>Property:commandManager.commands.gesturevar diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.nodes.length > 0 diagram.selectedItems.connectors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes.length; i++) { diagram.selectedItems.nodes[i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }] });diagram.appendTo('#diagram');</pre>
Sets the key value, on recognition of which	<pre>Property:commandManager.commands.gesture.key\$\$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.Shift }, canExecute: function(args) { var</pre>	<pre>Property:commandManager.commands.gesture.keyvar diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.nodes</pre>

the command will be executed	<pre> diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length ; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } })); </pre>	<pre> s.length > 0 diagram.selectedItems.conne ctors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes .length; i++) { diagram.selectedItems.nodes [i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }} });diagram.appendTo('#diag ram'); </pre>
Sets a combination of key modifiers, on recognition of which the command will be executed.	<pre> Property:commandManager.commands.gesture.key Modifiers\$ \$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.S hift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length ; }, execute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } })); </pre>	<pre> Property:commandManager.com mands.gesture.keyModifiersvar diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.node s.length > 0 diagram.selectedItems.conne ctors.length > 0) { return true; } return false; }, execute: function() { for (var i = 0; i < diagram.selectedItems.nodes .length; i++) { diagram.selectedItems.nodes [i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }} });diagram.appendTo('#diag ram'); </pre>
Defines any additional parameters that are required at	<pre> Property:commandManager.commands.parameter\$ \$("#diagramcontent").ejDiagram({ commandManager: { commands: { "clone": { parameter : "node", gesture: { key: ej.datavisualization.Diagram.Keys.C, keyModifiers: ej.datavisualization.Diagram.KeyModifiers.S hift }, canExecute: function(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); return diagram.model.selectedItems.children.length ; }, execute: function(args) { var diagram </pre>	<pre> Property:commandManager.com mands.parametervar diagram = new ej.diagrams.Diagram({ commandManager: { commands: [{ name: 'customCopy', parameter: 'node', canExecute: function() { if (diagram.selectedItems.node s.length > 0 diagram.selectedItems.conne ctors.length > 0) { return true; } return false; }, execute: function() { for </pre>

runtime	<pre>= \$("#diagramcontent").ejDiagram("instance"); diagram.copy(); diagram.paste(); } } } }));</pre>	<pre>(var i = 0; i < diagram.selectedItems.nodes .length; i++) { diagram.selectedItems.nodes [i].style.fill = 'red'; } diagram.dataBind(); }, gesture: { key: Keys.G, keyModifiers: KeyModifiers.Shift } }) });diagram.appendTo('#diag ram');</pre>
---------	---	--

Connectors

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Allows the user to save custom information/data about a connector	Property:connectors.addInfo <pre>var addInfo = { Description: "Bidirectional Flow"};var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, addInfo: addInfo};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.addInfo <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Defines the bridgeSpace of connector	Property:connectors.bridgeSpace <pre>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, bridgeSpace: 15, targetPoint: { x: 200, y: 200 },};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.bridgeSpace <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, bridgeSpace: 15, targetPoint: { x: 600, y: 200 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Enables or disables the behaviors of	Property:connectors.constraints <pre>var ConnectorConstraints = ej.datavisualization.Diagram.ConnectorConstra ints;var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, constraints: ConnectorConstraints.Default &</pre>	Property:connectors.constraints <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, constraints: ConnectorConstraints.Defa</pre>

connectors	~ConnectorConstraints.Select}};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	ult ConnectorConstraints.Drag }};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');
Defines the radius of the rounded corner	Property:connectors.cornerRadiusvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, cornerRadius: 10, segments:[{ type: "orthogonal" }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.cornerRadiusvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, cornerRadius: 10, type: 'Orthogonal', }];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');
Customize connectors appearance using user-defined CSS	Property:connectors.cssClassvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, cssClass: "hoverConnector"};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Not applicable
Sets the horizontal alignment of the connector	Property:connectors.horizontalAlignvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, horizontalAlign:ej.datavisualization.Diagram.HorizontalAlignment.Right}};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Not applicable
A collection of JSON objects where each object represents	Property:connectors.labelsvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, labels:[{ text:"connector" }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.annotationvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, annotations: [{ id: 'label', content: 'Text', offset: 0.5 }]}];var diagram = new

ents a label		<code>ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Stroke color of the connector	<code>Property:connectors.lineColor</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, lineColor: "blue"};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	<code>Property:connectors.style.strokeColor</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, style: { strokeColor: 'blue' }},];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Sets the pattern of dashes and gaps used to stroke the path of the connector	<code>Property:connectors.lineDashArray</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, lineColor: "blue", lineDashArray: "2,2"};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	<code>Property:connectors.style.strokeDashArray</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, style: { strokeColor: 'blue', strokeWidth: 3, strokeDashArray: '2,2' }},];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Sets the width of the line	<code>Property:connectors.lineWidth</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, lineWidth: 10};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	<code>Property:connectors.style.strokeWidth</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, style: { strokeColor: 'blue', strokeWidth: 3, strokeDashArray: '2,2' }},];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Defines the padding	<code>Property:connectors.lineHitPadding</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 },</code>	<code>Property:connectors.hitPadding</code> <code>var connectors = [{ id: 'connector', type:</code>

g value to ease the interaction with connectors	<pre> lineHitPadding: 15};\$("#diagramcontent").ejDiagram({ connectors: [connector]}); </pre>	<pre> 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, hitPadding: 10}};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram'); </pre>
Defines the minimum space to be left between the bottom of parent bounds and the connector	<pre> Property:connectors.marginBottomvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, marginBottom: 15};\$("#diagramcontent").ejDiagram({ connectors: [connector]}); </pre>	<p>Property:connectors.margin.bottom</p> <pre> var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, margin: { bottom: 3 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram'); </pre>
Defines the minimum space to be left between the top of parent bounds and the connector	<pre> Property:connectors.marginTopvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, marginTop: 15};\$("#diagramcontent").ejDiagram({ connectors: [connector]}); </pre>	<p>Property:connectors.margin.top</p> <pre> var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, margin: { top: 3 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram'); </pre>
Defines the minimum space	<pre> Property:connectors.marginLeftvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, marginLeft: </pre>	<p>Property:connectors.margin.left</p> <pre> var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, </pre>

to be left between in the left of parent bounds and the connector	<pre>15};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>targetPoint: { x: 600, y: 200 }, margin: { left: 3 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Defines the minimum space to be left between in the right of parent bounds and the connector	<pre>Property:connectors.marginRightvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, marginRight: 15};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.margin.righ htvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, margin: { right: 3 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Sets a unique name for the connector	<pre>Property:connectors.namevar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 },};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.idvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Defines the transparency of the connector	<pre>Property:connectors.opacityvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, opacity: 0.5};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.style.opaci tyvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, style: { opacity: 0.5 },},];var diagram = new ej.diagrams.Diagram({ connectors:</pre>

		<code>connectors});diagram.appendTo('#diagram');</code>
Sets the parent name of the connector.	Property: <code>connectors.parent</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, parent: "group"};var group = { name: "group", children: ["connector"]};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [group]});</code>	Not applicable
An array of JSON objects where each object represents a segment	Property: <code>connectors.segments</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "straight", point: { x: 75, y: 150 } }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	Property: <code>connectors.segments</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Orthogonal', length: 30, direction: 'Bottom' }, { type: 'Orthogonal', length: 80, direction: 'Right' }]}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Sets the direction of orthogonal segment	Property: <code>connectors.segments.direction</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "straight", point: { x: 75, y: 150 }, direction: "bottom" }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	Property: <code>connectors.segments.direction</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Orthogonal', length: 30, direction: 'Bottom' }, { type: 'Orthogonal', length: 80, direction: 'Right' }]}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Describes the length of orthogonal	Property: <code>connectors.segments.length</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "straight", point: { x: 75, y: 150 }, length: 50 }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</code>	Property: <code>connectors.segments.length</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Orthogonal', length: 30,</code>

segment		<pre>direction: 'Bottom' }, { type: 'Orthogonal', length: 80, direction: 'Right' }]]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Describes the end point of bezier/straight segment	<pre>Property:connectors.segments.pointvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "straight", point: { x: 75, y: 150 } }]]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.segments. pointvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Straight', point: { x: 800, y: 50 } }]]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Defines the first control point of the bezier segment	<pre>Property:connectors.segments.point1var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "bezier", point1: { x: 150, y: 50 } }]]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.segments. point1var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 600, y: 300 }, point1: { x: 525, y: 475 } }]]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append ndTo('#diagram');</pre>
Defines the second control point of bezier segment	<pre>Property:connectors.segments.point2var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "bezier", point1: { x: 150, y: 50 }, point2: { x: 150, y: 150 } }]]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.segments. point2var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 600, y: 300 }, point1: { x: 525, y: 475 }, point2: { x: 575, y: 475 } }]]];var diagram = new ej.diagrams.Diagram({ connectors:</pre>

		connectors});diagram.appendTo('#diagram');
Sets the type of the segment	Property:connectors.segments.type <pre>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: ej.datavisualization.Diagram.Segments.Bezier }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.segments.type <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 600, y: 300 }, point1: { x: 525, y: 475 }, point2: { x: 575, y: 475 } }]}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</pre>
Describes the length and angle between in the first control point and the start point of bezier segment	Property:connectors.segments.vector1 <pre>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "bezier", vector1: { distance: 75, angle: 0 }, vector2: { distance: 75, angle: 180 } }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.segments.vector1 <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 900, y: 160 }, vector1: { angle: 20, distance: 75 }, vector2: { angle: 20, distance: 75 } }],};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</pre>
Describes the length and angle between in the second control point and end	Property:connectors.segments.vector2 <pre>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, segments: [{ type: "bezier", vector1: { distance: 75, angle: 0 }, vector2: { distance: 75, angle: 180 } }]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.segments.vector2 <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 900, y: 160 }, vector1: { angle: 20, distance: 75 }, vector2: { angle: 20, distance: 75 } }],};var diagram = new ej.diagrams.Diagram({</pre>

point of bezier segment		connectors: connectors});diagram.appendTo('#diagram');
Sets the type of the connector	Property:connectors.shape.type var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, shape: { type: "bpmn" }, segments: [{ type: "straight"}]};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.shape.type var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, segments: [{ type: 'Bezier', point: { x: 600, y: 300 }, point1: { x: 525, y: 475 }, point2: { x: 575, y: 475 } }], shape: { type: 'Bpmn', flow: 'Message', message: 'InitiatingMessage' } }];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');
Defines the source decorator of the connector	Property:connectors.sourceDecorator var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow" } };\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.sourceDecorator var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');
Sets the border color of the source decorator	Property:connectors.sourceDecorator.borderColor var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow", borderColor: "red" } };\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.sourceDecorator.style.strokeColor var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', style: { strokeColor: 'red' }, }, }];var diagram = new ej.diagrams.Diagram({ connectors:

		<code>connectors});diagram.appendTo('#diagram');</code>
Sets the border width of the decorator	Property: <code>connectors.sourceDecorator.borderWidth</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow", borderWidth: 5 } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });</code>	Property: <code>connectors.sourceDecorator.style.strokeWidth</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', strokeWidth: 5 }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Defines to customize source Decorator appearance using user-defined CSS	Property: <code>connectors.sourceDecorator.cssClass</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow", cssClass: "hoverDecorator" } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });</code>	Not applicable
Sets the fill color of the source decorator	Property: <code>connectors.sourceDecorator.fillColor</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow", fillColor: "red" } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });</code>	Property: <code>connectors.sourceDecorator.style.fill</code> <code>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', fill: 'black' }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</code>
Sets the height of the	Property: <code>connectors.sourceDecorator.height</code> <code>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { width: 10, height:</code>	Property: <code>connectors.sourceDecorator.height</code> <code>var connectors = [{ id: 'connector', type: 'Straight',</code>

source decorator	<pre>10 } };\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', height: 10, width: 10 },,}};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Defines the custom shape of the source decorator	<pre>Property:connectors.sourceDecorator.pathDatavar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "path", pathData: "M 376.892, 225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.sourceDec orator.pathDatavar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Custom', pathData: "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" }},,}};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Defines the shape of the source decorator.	<pre>Property:connectors.sourceDecorator.shapevar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: ej.datavisualization.Diagram.DecoratorShapes. Circle } };\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.sourceDec orator.shapevar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, sourceDecorator: { shape: 'Arrow', },,}};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Defines the width of the source	<pre>Property:connectors.sourceDecorator.widthvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, sourceDecorator: { shape: "openarrow", width: 10, height: 10 }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre>Property:connectors.sourceDec orator.widthvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 },</pre>

decorator		<pre>sourceDecorator: { shape: 'Arrow', width: 10, height: 10 },},},var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</pre>
Sets the source node of the connector	Property:connectors.sourceNode <pre>var node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50};var node2 = { name: "target", offsetX: 300, offsetY: 300, width: 50, height: 50};var connector = { name: "connector", sourceNode: "source", targetNode: "target"};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes:[node1, node2]});</pre>	Property:connectors.sourceID <pre>var nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90 }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90 }];var connectors = [{ id: 'connector', type: 'Straight', sourceID: 'source', targetID: 'target'}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram');</pre>
Defines the space to be left between in the source node and the source point of a connector	Property:connectors.sourcePadding <pre>var node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50};var node2 = { name: "target", offsetX: 300, offsetY: 300, width: 50, height: 50};var connector = { name: "connector", sourceNode: "source", targetNode: "target", sourcePadding: 2, targetPadding: 2};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [node1, node2]});</pre>	Property:connectors.hitPadding <pre>var nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90 }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90 }];var connectors = [{ id: 'connector', type: 'Straight', hitPadding: 2, sourceID: 'source', targetID: 'target'}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram');</pre>
Describes the start point of the connector	Property:connectors.sourcePoint <pre>var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 } };\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Property:connectors.sourcePoint <pre>var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 },},},},var diagram = new ej.diagrams.Diagram({</pre>

		connectors: connectors});diagram.append ndTo('#diagram');
Sets the source port of the connec tor	Property:connectors.sourcePort var node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50, ports: [{ name: "port", offset: { x: 1, y: 0.5 } }]};var node2 = { name: "target", offsetX: 200, offsetY: 200, width: 50, height: 50, ports: [{ name: "port1", offset: { x: 0, y: 0.5 } }]};var connector = { name: "connector", sourceNode: "source", targetNode: "target", sourcePort: "port", targetPort: "port1"};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [node1, node2]});	Property:connectors.sourcePort IDvar nodeport1 = { id: 'port', shape: 'Square', offset: { x: 1, y: 0.5 }};var nodeport2 = { id: 'port1', shape: 'Square', offset: { x: 0, y: 0.5 }};var nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90, ports: [nodeport1] }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90, ports: [nodeport2] }];var connectors = [{ id: 'connector', type: 'Straight', sourceID: 'source', targetID: 'target', sourcePortID: 'port', targetPortID: 'port1',}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram');
Defines the target decora tor of the connec tor	Property:connectors.targetDecorator var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow" }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.targetDeco ratorvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', },}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');
Sets the border color of the target	Property:connectors.targetDecorator.borderColor var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow", borderColor: "red" }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});	Property:connectors.targetDeco rator.style.strokeColorvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: {

decorator		<pre> shape: 'Arrow', style: { strokeColor: 'red' }, },,]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append nTo('#diagram');</pre>
Sets the border width of the decorator	<pre> Property:connectors.targetDecorator.borderWidthvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow", borderWidth: 5 }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre> Property:connectors.targetDecorator.style.strokeWidthvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', strokeWidth: 5 },,]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append nTo('#diagram');</pre>
Defines to customize target Decorator appearance using user-defined CSS	<pre> Property:connectors.targetDecorator.cssClassvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow", cssClass: "hoverDecorator" }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	Not applicable
Sets the fill color of the target decorator	<pre> Property:connectors.targetDecorator.fillColorvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow", fillColor: "red" }};\$("#diagramcontent").ejDiagram({ connectors: [connector]});</pre>	<pre> Property:connectors.targetDecorator.style.fillvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', fill: 'black' },,]];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append nTo('#diagram');</pre>

Sets the height of the target decorator	Property:connectors.targetDecorator.height var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { width: 10, height: 10 } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });	Property:connectors.targetDecorator.height var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', height: 10, width: 10 }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors });diagram.appendTo('#diagram');
Defines the custom shape of the target decorator	Property:connectors.targetDecorator.pathData var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "path", pathData: "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });	Property:connectors.targetDecorator.pathData var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Custom', pathData: "M 376.892,225.284 L 371.279,211.95 L 376.892,198.617 L 350.225,211.95 L 376.892,225.284 Z" }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors });diagram.appendTo('#diagram');
Defines the shape of the target decorator.	Property:connectors.targetDecorator.shape var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: ej.datavisualization.Diagram.DecoratorShapes.Circle } };\$("#diagramcontent").ejDiagram({ connectors: [connector] });	Property:connectors.targetDecorator.shape var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 }, targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', }, }];var diagram = new ej.diagrams.Diagram({ connectors: connectors });diagram.appendTo('#diagram');
Defines the width of the target	Property:connectors.targetDecorator.width var connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, targetDecorator: { shape: "openarrow", width: 10, height: 10 } }	Property:connectors.targetDecorator.width var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 500, y: 100 },

decorator	<pre> });\$("#diagramcontent").ejDiagram({ connectors: [connector]}); </pre>	<pre> targetPoint: { x: 600, y: 200 }, targetDecorator: { shape: 'Arrow', width: 10, height: 10 },,}};var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram'); </pre>
Sets the target node of the connector	<pre> Property:connectors.targetNodevar node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50};var node2 = { name: "target", offsetX: 300, offsetY: 300, width: 50, height: 50};var connector = { name: "connector", sourceNode: "source", targetNode: "target"};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [node1, node2]}); </pre>	<pre> Property:connectors.targetIDva r nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90 }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90 }];var connectors = [{ id: 'connector', type: 'Straight', sourceID: 'source', targetID: 'target'}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram'); </pre>
Defines the space to be left between in the target node and the target point of a connector	<pre> Property:connectors.targetPaddingvar node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50};var node2 = { name: "target", offsetX: 300, offsetY: 300, width: 50, height: 50};var connector = { name: "connector", sourceNode: "source", targetNode: "target", sourcePadding: 2, targetPadding: 2};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [node1, node2]}); </pre>	<pre> Property:connectors.hitPadding var nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90 }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90 }];var connectors = [{ id: 'connector', type: 'Straight', hitPadding: 2 sourceID: 'source', targetID: 'target'}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram'); </pre>
Describes the start point of the	<pre> Property:connectors.targetPointvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }};\$("#diagramcontent").ejDiagram({ connectors: [connector]}); </pre>	<pre> Property:connectors.targetPoint var connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 },,}};var diagram = </pre>

connector		<pre>new ej.diagrams.Diagram({ connectors: connectors});diagram.appendTo('#diagram');</pre>
Sets the target port of the connector	<pre>Property:connectors.targetPortvar node1 = { name: "source", offsetX: 100, offsetY: 100, width: 50, height: 50, ports: [{ name: "port", offset: { x: 1, y: 0.5 } }]};var node2 = { name: "target", offsetX: 200, offsetY: 200, width: 50, height: 50, ports: [{ name: "port1", offset: { x: 0, y: 0.5 } }]};var connector = { name: "connector", sourceNode: "source", targetNode: "target", sourcePort: "port", targetPort: "port1"};\$("#diagramcontent").ejDiagram({ connectors: [connector], nodes: [node1, node2]});</pre>	<pre>Property:connectors.targetPortl Dvar nodeport1 = { id: 'port', shape: 'Square', offset: { x: 1, y: 0.5 }};var nodeport2 = { id: 'port1', shape: 'Square', offset: { x: 0, y: 0.5 }};var nodes = [{ id: 'source', width: 60, height: 60, offsetX: 75, offsetY: 90, ports: [nodeport1] }, { id: 'target', width: 75, height: 70, offsetX: 210, offsetY: 90, ports: [nodeport2] }];var connectors = [{ id: 'connector', type: 'Straight', sourceID: 'source', targetID: 'target', sourcePortID: 'port', targetPortID: 'port1',}];var diagram = new ej.diagrams.Diagram({ connectors: connectors, nodes: nodes});diagram.appendTo('#diagram');</pre>
Defines the tooltip that should be shown when the mouse hovers over connector	<pre>Property:connectors.tooltipvar tooltip = { templateId: "mouseovertooltip", alignment: { horizontal: "center", vertical: "bottom" }};var ConnectorConstraints = ej.datavisualization.Diagram.ConnectorConstra ints;var connector = { name: "flow", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, constraints: ConnectorConstraints.Default & ~ConnectorConstraints.InheritTooltip, tooltip: tooltip};\$("#diagramcontent").ejDiagram({ connectors: [connector]}) < script type = "text/x-jsrender" id = "mouseovertooltip" > < div style = "background-color: #F08080; color: white; white-space: nowrap; height: 20px" > < span style = "padding: 5px;" > < /span> < /div> < /script>var tooltip = { templateId: "mouseovertooltip", alignment: { horizontal: "center", vertical: "bottom" }};var ConnectorConstraints = ej.datavisualization.Diagram.ConnectorConstra</pre>	<pre>Property:connectors.tooltipvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, constraints: ConnectorConstraints.Defa ult ConnectorConstraints.Tool tip, tooltip: { content: 'Connector', position: 'TopCenter', showTipPointer: true, },}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.append To('#diagram');</pre>

	<pre>ints;var connector = { name: "flow", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, constraints: ConnectorConstraints.Default & ~ConnectorConstraints.InheritTooltip, tooltip: tooltip};\$("#diagramcontent").ejDiagram({ connectors: [connector]});;</pre>	
Sets the vertical alignment of connector	<pre>Property:connectors.verticalAlignvar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 150, y: 150 }, verticalAlign: ej.datavisualization.Diagram.VerticalAlignmen t.Bottom};\$("#diagramcontent").ejDiagram({ connectors: [connector]});;</pre>	Not applicable
Enables or disables the visibility of connector	<pre>Property:connectors.visiblevar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, visible: true};\$("#diagramcontent").ejDiagram({ connectors: [connector]});;</pre>	<pre>Property:connectors.visiblevar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, visible: true}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Sets the z-index of the connector	<pre>Property:connectors.zOrdervar connector = { name: "connector", sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, zOrder: 1000};\$("#diagramcontent").ejDiagram({ connectors: [connector]});;</pre>	<pre>Property:connectors.zIndexvar connectors = [{ id: 'connector', type: 'Straight', sourcePoint: { x: 100, y: 100 }, targetPoint: { x: 200, y: 200 }, zIndex: -1}];var diagram = new ej.diagrams.Diagram({ connectors: connectors});diagram.appe ndTo('#diagram');</pre>
Binds the custom JSON data with connector properties	<pre>Property:connectors.connectorTemplate var data = [{"Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Id", parent: "ReportingPerson", dataSource: data }, connectorTemplate: "connectorTemplate"});function connectorTemplate(diagram, connector) { if (connector.sourceNode &&</pre>	Not applicable

	<code>connector.targetNode) { connector.linecolor = "green"; }}</code>	
Enables/Disables the default behaviors of the diagram	Property:constraints <code>var DiagramConstraints = ej.datavisualization.Diagram.DiagramConstraints;\$("#diagramcontent").ejDiagram({ constraints: DiagramConstraints.Default DiagramConstraints.Bridging});</code>	Property:constraints <code>var diagram = new ej.diagrams.Diagram({ constraints: DiagramConstraints.Default DiagramConstraints.Bridging});diagram.appendTo('#diagram');</code>

ContextMenu

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the collection of context menu items	Property:contextMenu.items <code>var menuitems = [{ "name": "hyperLink", "text": "Hyperlink", "image": "", "style": ""}];var contextMenu = { items: menuitems};\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	Property:contextMenuSettings.items <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'delete', id: 'delete', target: '.e-diagramcontent', iconCss: 'e-copy' }] },});diagram.appendTo('#diagram');</code>
Defines the text for the collection of context menu item	Property:contextMenu.items.text <code>var menuitems = [{ "text": "ZoomIn"}];var contextMenu = { items: menuitems};\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	Property:contextMenuSettings.items .text <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'ZoomIn' }] },});diagram.appendTo('#diagram');</code>
Defines the name for the collection of context menu items	Property:contextMenu.items.name <code>var menuitems = [{ "name": "hyperLink"}];var contextMenu = { items: menuitems};\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	Property:contextMenuSettings.items .id <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'delete', id: 'delete' }] },});diagram.appendTo('#diagram');</code>
Defines the image url for the collection of context	Property:contextMenu.items.imageUrl <code>var menuitems = [{ "name": "zoomin", "text": "ZoomIn", "imageUrl": "Images/zoomin.png", "style": ""}];var contextMenu = { items:</code>	Property:contextMenuSettings.items .url <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ 'id':</code>

menu items	<code>menuitems});\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	<code>'zoomin', 'text': 'ZoomIn', 'url': 'Images/zoomin.png', }, }, });diagram.appendTo('#diagram');</code>
Defines the cssClass for the collection of context menu items	Property:contextMenu.items.cssClass <code>var menuitems = [{ "name": "zoomin", "text": "ZoomIn", "imageUrl": "Images/zoomin.png", "cssClass": "menu", "style": ""}];var contextMenu = { items: menuitems};\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	Property:contextMenuSettings.items <code>.iconCssvar diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'delete', id: 'delete', target: '.e-diagramcontent', iconCss: 'e-copy' }], }, });diagram.appendTo('#diagram');</code>
Defines the collection of sub items for the context menu items	Property:contextMenu.items.subItems <code>\$("#diagramcontent").ejDiagram({ contextMenu: { items: [{ name: "zoom", text: "Zoom", subItems: [{ name: "zoomIn", text: "ZoomIn" }, { name: "zoomOut", text: "ZoomOut" }] }] } });</code>	Property:contextMenuSettings.items <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'Zoom', id: 'zoom', items: [{ name: "zoomIn", text: "ZoomIn" }, { name: "zoomOut", text: "ZoomOut" }] }, { showCustomMenuOnly: false, }, });diagram.appendTo('#diagram');</code>
Set whether to display the default context menu items or not	Property:contextMenu.showCustomMenuItemsOnly <code>var contextMenu = { showCustomMenuItemsOnly: true};\$("#diagramcontent").ejDiagram({ contextMenu: contextMenu});</code>	Property:contextMenuSettings.showCustomMenuOnly <code>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { showCustomMenuOnly: false, }, });diagram.appendTo('#diagram');</code>
Specifies separator between the menu items	Not applicable	Property:contextMenuSettings.items <code>.separatorvar diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'Save', id: 'save', target: '.e-diagramcontent', iconCss: 'e-save', separator: true }, { text: 'Load', id: 'load', target: '.e-diagramcontent', iconCss: 'e-load' }, }, }, });diagram.appendTo('#diagram');</code>

Define the target to show the menu item.	Not applicable	Property:contextMenuSettings.items.target <pre>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'delete', id: 'delete', target: '.e-diagramcontent', iconCss: 'e-copy' }]}, showCustomMenuOnly: false, },});diagram.appendTo('#diagram');</pre>
Enables/Disables the context menu items	Not applicable	Property:contextMenuSettings.show <pre>var diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true, items: [{ text: 'delete', id: 'delete', target: '.e-diagramcontent', iconCss: 'e-copy' }]}, showCustomMenuOnly: false, },});diagram.appendTo('#diagram');</pre>

DataSourceSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the data source either as a collection of objects or as an instance of ej.Data Manager	Property:dataSourceSettings.dataSource <pre>var data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ dataSourceSettings: { dataSource: data }});</pre>	Property:dataSourceSettings.dataManager.items <pre>var items = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];var diagram = new ej.diagrams.Diagram({ dataSourceSettings: { dataManager: items },});diagram.appendT o('#diagram');</pre>
Sets the unique id of the	Property:dataSourceSettings.id <pre>var data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana</pre>	Property:dataSourceSettings.id <pre>var items = [{</pre>

data source items	<pre>Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Id", dataSource: data }});</pre>	<pre>"Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];var diagram = new ej.diagrams.Diagram({ dataSourceSettings: { id: 'Id', dataManager: items },});diagram.appendT o('#diagram');</pre>
Defines the parent id of the data source item	<pre>Property:dataSourceSettings.parentIdvar data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Id", parent: "ReportingPerson", dataSource: data }});</pre>	<pre>Property:dataSourceSettings.parentIdvar items = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];var diagram = new ej.diagrams.Diagram({ dataSourceSettings: { id: 'Id', parentId: 'ReportingPerson', dataManager: items },});diagram.appendT o('#diagram');</pre>
Describe s query to retrieve a set of data from the specified	<pre>Property:dataSourceSettings.query\$("#diagramcontent"). ejDiagram({ dataSourceSettings: { dataSource: ej.DataManager({ url: "http://mvc.syncfusion.com/Services/Northwnd.svc/" }), query: ej.Query().from("Employees").select("EmployeeID,R eportsTo,FirstName"), tableName: "Employees", id: "EmployeeID", parent: "ReportsTo" }});</pre>	Not applicable

datasource		
Sets the unique id of the root data source item	<pre>Property:datasourceSettings.rootvar data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ datasourceSettings: { id: "Id", parent: "ReportingPerson", root: "E1", dataSource: data } });</pre>	<pre>Property:datasourceSettings.rootvar items = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];var diagram = new ej.diagrams.Diagram({ datasourceSettings: { id: 'Id', parentId: 'ReportingPerson', dataManager: items, root: 'E1' }, });diagram.appendTo('#diagram');</pre>
Describes the name of the table on which the specified query has to be executed	<pre>Property:datasourceSettings.tableName\$("#diagramcontent").ejDiagram({ datasourceSettings: { dataSource: ej.DataManager({ url: "http://mvc.syncfusion.com/Services/Northwnd.svc/" }), query: ej.Query().from("Employees").select("EmployeeID, ReportsTo, FirstName"), //Table name tableName: "Employees", id: "EmployeeID", parent: "ReportsTo" } });</pre>	Not applicable
Specifies the method name which is used to get the updated data	<pre>Property:datasourceSettings.crudAction\$("#diagramcontent").ejDiagram({ datasourceSettings: { id: "Name", crudAction: { read: "http://js.syncfusion.com/demos/ejservices/api/Diagram/GetNodes" } } });</pre>	Not applicable

from client side to the server side		
Specifies the create method which is used to get the nodes to be added from client side to the server side	<pre>Property: dataSourceSettings.crudAction.create\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", crudAction: { create: "http://js.syncfusion.com/demos/ejservices/api/Diagram/AddNodes", } } });</pre>	Not applicable
Specifies the update method which is used to get the updated data from client side to the server side	<pre>Property: dataSourceSettings.crudAction.update\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", crudAction: { update: "http://js.syncfusion.com/demos/ejservices/api/Diagram/UpdateNodes", } } });</pre>	Not applicable
Specifies the destroy method which is used to get the deleted items	<pre>Property: dataSourceSettings.crudAction.destroy\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", crudAction: { destroy: "http://js.syncfusion.com/demos/ejservices/api/Diagram/DeleteNodes" } } });</pre>	Not applicable

data from client side to the server side		
Specifies the read method to get the created nodes from client side to the server side	Property: <code>dataSourceSettings.crudAction.read\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", crudAction: { read: http://js.syncfusion.com/demos/ejservices/api/Diagram/GetNodes } } });</code>	Not applicable
Defines the data source either as a collection of objects or as an instance of ej.Data Manager	Property: <code>dataSourceSettings.customFields\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: 'Name', customFields: ["Description", "Color"] } });</code>	Property: <code>dataSourceSettings.data</code> <code>var diagram = new ej.diagrams.Diagram({ dataSourceSettings: { id: 'Name', customFields: ["Description", "Color"] }, }); diagram.appendTo('#diagram');</code>
Defines the data source either as a collection of objects or as an instance of ej.Data	Property: <code>dataSourceSettings.connectionDataSource\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name" } } });</code>	Not applicable

Manager		
Sets the datasource for the connection datasource settings items	Property: <code>dataSourceSettings.connectionDataSource.dataSource</code> <pre> var data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }]; \$("#diagramcontent").ejDiagram({dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", dataSource: data } } }); </pre>	Not applicable
Sets the unique id of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.id</code> <pre> \$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name" } } }); </pre>	Not applicable
Sets the source node of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.sourceNode</code> <pre> \$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", sourceNode: "sourceNode", } } }); </pre>	Not applicable
Sets the target node of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.targetNode</code> <pre> \$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", targetNode: "targetNode" } } }); </pre>	Not applicable
Sets the sourcePointX value of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.sourcePointX</code> <pre> \$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", sourcePointX: 200 } } }); </pre>	Not applicable

Sets the sourcePointY value of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.sourcePointY</code> <pre>intY\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", sourcePointY:200 } } });</pre>	Not applicable
Sets the x point value of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.targetPointX</code> <pre>intX\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", targetPointX:200 } } });</pre>	Not applicable
Sets the y point value of the connection data source item	Property: <code>dataSourceSettings.connectionDataSource.targetPointY</code> <pre>intY\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", targetPointY:200 } } });</pre>	Not applicable
Specifies the method name which is used to get updated connectors from client side to the server side	Property: <code>dataSourceSettings.connectionDataSource.crudAction</code> <pre>on\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", sourceNode: "sourceNode", targetNode: "targetNode", crudAction: { read: http://js.syncfusion.com/demos/ejservices/api/Dia gram/GetConnectors" } } } });</pre>	Not applicable
Specifies the create method which is	Property: <code>dataSourceSettings.connectionDataSource.crudAction.create</code> <pre>on.create\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", sourceNode: "sourceNode", targetNode: "targetNode", crudAction: { create:</pre>	Not applicable

used to get the connectors to be added from client side to the server side	<code>http://js.syncfusion.com/demos/ejservices/api/Diagram/AddConnectors", } } } }));</code>	
Specifies the update method which is used to get the updated connectors from client side to the server side	Property: <code>dataSourceSettings.connectionDataSource.crudAction.update\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", crudAction: { update: http://js.syncfusion.com/demos/ejservices/api/Diagram/UpdateConnectors", } } } }));</code>	Not applicable
Specifies the destroy method which is used to get the deleted items data from client side to the server side	Property: <code>dataSourceSettings.connectionDataSource.crudAction.destroy\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", crudAction: { destroy: http://js.syncfusion.com/demos/ejservices/api/Diagram/DeleteConnectors" } } } }));</code>	Not applicable
Specifies the read method which is	Property: <code>dataSourceSettings.connectionDataSource.crudAction.read\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", crudAction: {</code>	Not applicable

used to get the data from client side to the server side	<pre>read: http://js.syncfusion.com/demos/ejservices/api/Diagram/GetConnectors" } } } }));</pre>	
Specifies the custom fields to get the updated data from client side to the server side	<pre>Property:dataSourceSettings.connectionDataSource.customFields\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Name", connectionDataSource: { id: "Name", customFields: ["Description", "Color"] } } });</pre>	Not applicable
Binds the custom data with node model	<pre>Property:dataSourceSettings.doBinding\$("#diagramcontent").ejDiagram({ width: 1500, height: 2500, layout: { type: 'HierarchicalTree', verticalSpacing: 40 }, dataSourceSettings: { id: 'Name', parentId: 'ReportingPerson', dataManager: items, doBinding: (nodeModel: NodeModel, data: object, diagram: Diagram) => { nodeModel.annotations = [{ content: data['Name'], margin: { top: 10 } }]; } }));</pre>	Not applicable

DefaultSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Initializes the default values for nodes and connectors	<pre>Property:defaultSettings.node\$("#diagramcontent").ejDiagram({ defaultSettings: { node: { fillColor:"red" } } });</pre>	<pre>Property:getNodeDefaultsvar diagram = new ej.diagrams.Diagram({ getNodeDefaults: (object: Node) => { object.style = { fill: 'lightgrey', strokeColor: 'none', strokeWidth: 2 }; return object; });diagram.appendTo('#diagram');</pre>

Initialize the default connector properties	Property: <code>defaultSettings.connector\$</code> ("#diagramcontent").ejDiagram({ defaultSettings: { connector: { lineColor:"red", lineWidth:4, lineDashArray:"2,2" } } });	Property: <code>getConnectorDefaults</code> var diagram = new ej.diagrams.Diagram({ getConnectorDefaults: (connector: ConnectorModel) => { connector= { targetDecorator:{shape 'None'}, type : 'Orthogonal'}; return connector; } }); diagram.appendTo('#diagram');
Initialize the default properties of groups	Property: <code>defaultSettings.group\$</code> ("#diagramcontent").ejDiagram({ defaultSettings: { group: { constraints: ej.datavisualization.Diagram.NodeConstraints.Default & ~ej.datavisualization.Diagram.NodeConstraints.Drag } } });	Not applicable

DrawType

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Sets the type of JSON object to be drawn through drawing tool	Property: <code>drawType\$</code> ("#diagramcontent").ejDiagram({ drawType:{type:"node"} });	Property: <code>drawingObject</code> var diagram = new ej.diagrams.Diagram({ drawingObject : {id: 'connector', type: 'Straight'} }); diagram.appendTo('#diagram');

EnableAutoScroll

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enable or disable	Property: <code>enableAutoScroll\$</code> ("#diagramcontent").ejDiagram({ enableAutoScroll: false });	Property: <code>canAutoScroll</code> var diagram = new ej.diagrams.Diagram({

s auto scroll in diagram		<pre>canAutoScroll: true});diagram.appendTo('#diagram');</pre>
--------------------------	--	--

EnableContextMenu

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enable or disable diagram context menu	<pre>Property:enableContextMenu\$("#diagramcontent").ejDiagram({ enableContextMenu: false });</pre>	<pre>Property:contextMenuSettings.showvar diagram = new ej.diagrams.Diagram({ contextMenuSettings: { show: true }});diagram.appendTo('#diagram');</pre>

GetCustomCursor

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enable or disable rendering component with custom cursor	Not applicable	<pre>Property:getCustomCursorfunction getCustomCursor(action: string, active: boolean): string {var cursor;if (active && action === 'Drag') {cursor = '-webkit-grabbing';} else if (action === 'Drag') {cursor = '-webkit-grabbing'}return cursor;}var nodes = [{ id: 'node1', width: 100, height: 100, offsetX: 100, offsetY: 100, }, { id: 'node2', width: 100, height: 100, offsetX: 300, offsetY: 100, shape: { type: 'Basic', shape: 'Ellipse' }, },];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes, getCustomCursor: getCustomCursor});diagram.appendTo('#diagram');</pre>

GetCustomProperty

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Allows to get the custom properties that have to be serialized	Not applicable	Property: <code>getCustomProperty</code> <pre>var nodes = [{ id: 'node1', width: 100, height: 100, offsetX: 100, offsetY: 100, }, { id: 'node2', width: 100, height: 100, offsetX: 300, offsetY: 100, shape: { type: 'Basic', shape: 'Ellipse' } },];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes, getCustomProperty: (key: string) => { if (key === 'nodes') { return ['description']; } return null; });diagram.appendTo('#diagram');</pre>

GetCustomTool

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Allows to get the custom tool	Not applicable	Property: <code>getCustomTool</code> <pre>function getTool(action: string): ToolBase { var tool; if (action === 'userHandle') { tool = new CloneTool(diagram.commandHandler, true); } return tool;}class CloneTool extends ToolBase { public mouseDown(args: MouseEventArgs): void { super.mouseDown(args); diagram.copy(); diagram.paste(); }}var nodes = [{ id: 'node1', width: 100, height: 100, offsetX: 100, offsetY: 100, }, { id: 'node2', width: 100, height: 100, offsetX: 300, offsetY: 100, shape: { type: 'Basic', shape: 'Ellipse' } },];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes, getCustomTool: getTool});diagram.appendTo('#diagram');</pre>

Height

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2

Specifies the height of the diagram	Property:height\$("#diagramcontent").ejDiagram({ height:"500", width:"500" });	Property:heightvar diagram = new ej.diagrams.Diagram({ height:1000});diagram.appendTo('#diagram');
-------------------------------------	--	--

HistoryManager

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
A method that takes a history entry as argument and returns whether the specific entry can be popped or not	Property:historyManager.canPopvar diagram = \$("#diagramcontent").ejDiagram("instance");var entry = { object: node, prevState: node.empInfo };diagram.model.historyManager.push(entry);var value = { role: "New role" };node.empInfo = value;if(diagram.model.historyManager.canPop(entry)){diagram.model.historyManager.pop();}	Not applicable
A method that ends grouping the changes	Property:historyManager.closeGroupActionvar group = diagram.model.selectedItems;diagram.model.historyManager.startGroupAction();for (var i = 0; i < group.children.length; i++) { var option = {}; var item = group.children[i]; // Updates the fillColor for all the child elements. option.fillColor = args.style.backgroundColor;diagram.updateNode(item.name,	Property:historyList.endGroupActionvar diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node1', offsetX: 100, offsetY: 100, width: 100, height: 100, }, { offsetX: 200, offsetY: 200, width: 100, height: 100, id: 'node2' }], connectors: [{ id: 'connector1', sourcePoint: { x: 100, y: 200 }, targetPoint: { x: 200, y: 300 }, type: 'Orthogonal' }

	option); } diagram.model.historyManager.closeGroupAction();	}}}); diagram.appendTo('#diagram'); var objects = []; objects.push(diagram.nodes[0], diagram.nodes[1], diagram.connectors[0]); diagram.historyList.startGroupAction(); diagram.distribute('Top', objects); diagram.distribute('Bottom', objects); diagram.distribute('BottomToTop', objects); diagram.historyList.endGroupAction();
A method that removes the history of a recent change made in diagram	Property: historyManager.pop var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.model.historyManager.pop();	Not applicable
A method that allows to track the custom changes made in diagram	Property: historyManager.push var diagram = \$("#diagramcontent").ejDiagram("instance"); var entry = { object: node, prevState: node.empInfo }; diagram.model.historyManager.push(entry); var value = { role: "New role" }; node.empInfo = value;	Property: historyList.push var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }] }); diagram.appendTo('#diagram'); var object = diagram.nodes[0]; object['description'] = (document.getElementById('custom') as HTMLSelectElement).value; var entry = { undoObject: object }; diagram.historyList.push(entry); diagram.dataBind();
Defines what should be	Property: historyManager.redo \$("#diagramcontent").ejDiagram({ historyManager: { undo: customUndoRedo, redo: customUndoRedo } }); function customUndoRedo(args) { var diagram =	Property: historyList.redo var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }] }); diagram.appendTo('#diagram'

happened while trying to restore a custom change	<pre>\$("#diagramcontent").ejDiagram("instance"); var node = args.object; var currentState = node.empInfo; node.empInfo = args.prevState; args.prevState = currentState;</pre>	<pre>);var node1 = diagram.nodes[0];node1['customName'] = 'customNode';entry = {undoObject: node1};diagram.historyList.push(entry);diagram.historyList.undo = function(args: HistoryEntry) { args.redoObject = cloneObject(args.undoObject) as NodeModel; args.undoObject['customName'] = 'customNodeChange';}diagram.historyList.redo = function(args: HistoryEntry) { var current = cloneObject(args.undoObject) as NodeModel; args.undoObject['customName'] = args.redoObject['customName']; args.redoObject = current;}</pre>
Gets the number of redo actions to be stored on the history manager. Its an read-only property and the collection should not be modified	<p>Property: historyManager.redoStack</p> <pre>var diagram = \$("#diagramcontent").ejDiagram("instance");diagram.model.historyManager.redoStack();</pre>	<p>Property: historyList.redoStack</p> <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');diagram.historyList.redoStack();</pre>
Restricts	<p>Property: historyManager.stackLimit</p> <pre>var diagram =</pre>	Not applicable

the undo and redo actions to a certain limit	<pre>\$("#diagramcontent").ejDiagram("instance"); diagram.model.historyManager.stackLimit();</pre>	
A method that starts to group the changes to revert/restore them in a single undo or redo	<pre>Property:historyManager.startGroupAction var group = diagram.model.selectedItems diagram.model.historyManager.startGroupAction(); for (var i = 0; i < group.children.length; i++) { var option = {}; var item = group.children[i]; option.fillColor = args.style.backgroundColor; diagram.updateNode(item.name, option); } diagram.model.historyManager.closeGroupAction();</pre>	<pre>Property:historyList.startGroupAction var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node1', offsetX: 100, offsetY: 100, width: 100, height: 100, }, { offsetX: 200, offsetY: 200, width: 100, height: 100, id: 'node2' }], connectors: [{ id: 'connector1', sourcePoint: { x: 100, y: 200 }, targetPoint: { x: 200, y: 300 }, type: 'Orthogonal' }] }); diagram.appendTo('#diagram'); var objects = []; objects.push(diagram.nodes[0], diagram.nodes[1], diagram.connectors[0]); diagram.historyList.startGroupAction(); diagram.distribute('Top', objects); diagram.distribute('Bottom', objects); diagram.distribute('BottomToTop', objects); diagram.historyList.endGroupAction();</pre>
Define what should be happened while trying to revert a custom change	<pre>Property:historyManager.undo \$("#diagramcontent").ejDiagram({ historyManager: { undo: customUndoRedo, redo: customUndoRedo } }); function customUndoRedo(args) { var diagram = \$("#diagramcontent").ejDiagram("instance"); var node = args.object; var currentState = node.empInfo; node.empInfo = args.prevState; args.prevState = currentState; }</pre>	<pre>Property:historyList.undo var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }] }); diagram.appendTo('#diagram'); var node1 = diagram.nodes[0]; node1['customName'] = 'customNode'; entry = { undoObject: node1; }; diagram.historyList.push(entry); diagram.historyList.undo = function(args: HistoryEntry) { args.redoObject = cloneObject(args.undoObject) as NodeModel; args.undoObject['customName'] = 'customNodeChange'; } diagram.historyList.redo = function(args:</pre>

		<pre>HistoryEntry) { var current = cloneObject(args.undoObject) as NodeModel; args.undoObject['customName'] = args.redoObject['customName']; args.redoObject = current;}</pre>
Gets the number of undo actions to be stored on the history manager. Its an read-only property and the collection should not be modified	<pre>Property:historyManager.undoStackvar diagram = \$("#diagramcontent").ejDiagram("instanc e");diagram.model.historyManager.undoSt ack();</pre>	<pre>Property:historyList.undoStackvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');diagram.historyList.undoStack();</pre>
Set the current entry object	Not applicable	<pre>Property:historyList.currentEntryvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');diagram.historyList.currentEnt ry();</pre>
Set the history entry	Not applicable	<pre>Property:historyList.canUndovar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram'</pre>

can be undo		<code>);diagram.historyList.canUndo = true;</code>
Set the history entry can be redo	Not applicable	Property: <code>historyList.canRedo</code> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');diagram.historyList.canRedo = true;
Used to decide to stored the changes to history	Property: <code>historyManager.canLog</code> var diagram = \$("#diagramcontent").ejDiagram("instance");diagram.model.historyManager.canLog();	Property: <code>historyList.canLog</code> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');diagram.historyList.canLog = function (entry: HistoryEntry) { entry.cancel = true; return entry; }

LabelRenderingMode

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the type of the rendering mode of label	Property: <code>labelRenderingMode</code> \$("#diagramcontent").ejDiagram({ labelRenderingMode: "svg" });	Not applicable

Layout

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Specifies the custom bounds to arrange/align the layout	Property: <code>layout.bounds</code> \$("#diagramcontent").ejDiagram({ layout: { bounds: { x: 0, y: 0, width: 1000, height: 1000 } } });	Property: <code>layout.bounds</code> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { bounds: new Rect(0, 0, 500, 500) } });diagram.appendTo('#diagram');
Defines the fixed	Property: <code>layout.fixedNode</code> \$("#diagramcontent").ejDiagram({ fixedNode: "node" });	Property: <code>layout.fixedNode</code> var diagram = new

node with reference to which, the layout will be arranged and fixed node will not be repositioned		<pre>ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { fixedNode: 'node' } }); diagram.appendTo('# diagram');</pre>
Customizes the orientation of trees/sub trees	Property: layout.getLayoutInfo <pre>function getLayoutInfo(diagram, node, options) { options.orientation = "vertical"; options.type = "left"; offset = 10; }; \$("#diagramcontent").ejDiagram({ layout: { getLayoutInfo: getLayoutInfo } });</pre>	Property: layout.getLayoutInfo <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { getLayoutInfo: (node: Node, tree: TreeInfo) => { if (!tree.hasSubTree) { tree.orientation = 'vertical'; } }}}); diagram.appendTo('#diagram');</pre>
Defines a method to customize the segments based on source and target nodes	Property: layout.getConnectorSegments <pre>function getConnectorSegment(diagram, node, options) { }; \$("#diagramcontent").ejDiagram({ layout: { getConnectorSegments: getConnectorSegment } });</pre>	Property: layout.connectorSegments <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { connectorSegments: 'Default' }}); diagram.appendTo('#diagram');</pre>
Sets the space to be horizontally left between nodes	Property: layout.horizontalSpacing <pre>\$("#diagramcontent").ejDiagram({ layout: { horizontalSpacing: 50 } });</pre>	Property: layout.horizontalSpacing <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { horizontalSpacing: 30 }}); diagram.appendTo('#diagram');</pre>
Defines the space to be left between layout	Property: layout.margin <pre>\$("#diagramcontent").ejDiagram({ layout: { margin: { left: 10, right: 10, top: 10, bottom: 10 } }});</pre>	Property: layout.margin <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, layout: { margin: { left: 50, top: 50, right: 0, bottom: 0 } }</pre>

bounds and layout		<pre> }); diagram.appendTo('#diagram'); </pre>
Defines how to horizontally align the layout within the layout bounds	<pre> Property:layout.horizontalAlignment\$("#diagramcontent").ejDiagram({ layout: { horizontalAlignment:ej.datavisualization.Diagram.HorizontalAlignment.Center } }); </pre>	Property:layout.horizontalAlignment <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { horizontalAlignment: 'Center' }}); diagram.appendTo('#diagram'); </pre>
Defines how to vertically align the layout within the layout bounds	<pre> Property:layout.verticalAlignment\$("#diagramcontent").ejDiagram({ layout: { verticalAlignment:ej.datavisualization.Diagram.VerticalAlignment.Center } }); </pre>	Property:layout.verticalAlignment <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { verticalAlignment: 'Center' }}); diagram.appendTo('#diagram'); </pre>
Sets the orientation/direction to arrange the diagram elements	<pre> Property:layout.orientation\$("#diagramcontent").ejDiagram({ layout: { orientation: ej.datavisualization.Diagram.LayoutOrientations.LeftToRight } }); </pre>	Property:layout.orientation <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { orientation: 'TopToBottom', }}); diagram.appendTo('#diagram'); </pre>
Sets the type of the layout based on which the elements will be arranged	<pre> Property:layout.type\$("#diagramcontent").ejDiagram({ layout: { type: ej.datavisualization.Diagram.LayoutTypes.HierarchicalTree } }); </pre>	Property:layout.type <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { type: 'OrganizationalChart' }}); diagram.appendTo('#diagram'); </pre>
Sets the space to be vertically left between nodes	<pre> Property:layout.verticalSpacing\$("#diagramcontent").ejDiagram({ layout: { verticalSpacing: 50 } }); </pre>	Property:layout.verticalSpacing <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { verticalSpacing: 30 }}); diagram.appendTo('#diagram'); </pre>

Sets the value is used to define the root node of the layout	Property: layout.root <code>\$("#diagramcontent").ejDiagram({ layout: { root: 'rootNode' } });</code>	Property: layout.root <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { root: 'rootNode' } }); diagram.appendTo('#diagram');</code>
Defines how long edges should be, ideally. This will be the resting length for the springs	Property: layout.springFactor <code>\$("#diagramcontent").ejDiagram({ layout: { springFactor: 0.442 } });</code>	Property: layout.springFactor <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { type: 'SymmetricalLayout', springLength: 80, springFactor: 0.8, maxIteration: 500, } }); diagram.appendTo('#diagram');</code>
Defines how long edges should be, ideally. This will be the resting length for the springs	Property: layout.maxIteration <code>\$("#diagramcontent").ejDiagram({ layout: { maxIteration: 442 } });</code>	Property: layout.maxIteration <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { type: 'SymmetricalLayout', springLength: 80, springFactor: 0.8, maxIteration: 500, } }); diagram.appendTo('#diagram');</code>
Defines how long edges should be, ideally. This will be the resting length for the springs	Property: layout.springLength <code>\$("#diagramcontent").ejDiagram({ layout: { springLength: 80 } });</code>	Property: layout.springLength <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { type: 'SymmetricalLayout', springLength: 80, springFactor: 0.8, maxIteration: 500, } }); diagram.appendTo('#diagram');</code>
Sets how to define the connection direction (first segment)	Not applicable	Property: layout.connectionDirection <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { connectionDirection: 'Auto', type: 'SymmetricalLayout',</code>

direction & last segment direction)		springLength: 80, springFactor: 0.8, maxIteration: 500, });diagram.appendTo('#diagram');
Enables/Disables animation option when a node is expanded/collapsed	Not applicable	Property:layout.enableAnimation var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { enableAnimation: true, orientation: 'TopToBottom', type: 'OrganizationalChart', margin: { top: 20 }, horizontalSpacing: 30, verticalSpacing: 30, });diagram.appendTo('#diagram');
Defines whether an object should be at the left/right of the mind map. Applicable only for the direct children of the root node	Not applicable	Property:layout.getBranch var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }], layout: { type: 'MindMap', });diagram.appendTo('#diagram'); diagram.layout.getBranch = (node: NodeModel, nodes: NodeModel[]) => { return 'Left'; }diagram.dataBind();

Nodes

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Array of JSON objects where each object represents a node	Property:nodes var nodes = [{ name: "node1", width: 175, height: 60, offsetX:100, offsetY:100}];\$("#diagramcontent").ejDiagram({ nodes:nodes });	Property:nodes var node = { offsetX: 250, offsetY: 250, width: 100, height: 100,};var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendTo('#diagram');
Defines the type of	Property:nodes.activity var nodes = [{ type: "bpmn", shape:	Property:nodes.shape.activity var node = { offsetX:

BPMN Activity. Applicable, if the node is a BPMN activity	<pre>ej.datavisualization.Diagram.BPMNShapes.Activity, activity: ej.datavisualization.Diagram.BPMNActivity.SubProcess, width:50, height:50 }};\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	<pre>250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'Task' }, },};var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendTo('#diagram');</pre>
To maintain additional information about nodes	<pre>Property:nodes.addInfovar addInfo = { TooltipData: "Shares the information with the customer" };var node1 = { name: "node1", addInfo: addInfo, offsetX:100, offsetY:100, width:50, height:50 };var node2 = { type: "swimlane", name: "swimlane", addInfo: addInfo }};\$("#diagramcontent").ejDiagram({nodes:[no del1, node2]});</pre>	<pre>Property:nodes.addInfovar node = { offsetX: 250, offsetY: 250, width: 100, height: 100, addInfo: { "borderColor": "black", "borderWidth": '1px' }},};var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendTo('#diagram');</pre>
Defines the additional information of a process. It is not directly related to the message flows or sequence flows of the process	<pre>Property:nodes.annotationvar nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50, angle: -45, length: 150, direction: "top" } }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	<pre>Property:nodes.shape.annotati onsvar node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', angle: 45, length: 150, text: 'Left', }] } };var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendTo('#diagram');</pre>
Sets the angle between the BPMN shape and the annotation	<pre>Property:nodes.annotation.anglevar nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50, angle: -45 } }};\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	<pre>Property:nodes.shape.annotati ons.anglevar node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', angle: 45, }] } };var diagram = new</pre>

		<code>ej.diagrams.Diagram({ nodes: [node]});diagram.appendT o('#diagram');</code>
Sets the direction of the text annotation	Property: <code>nodes.annotation.direction</code> <code>var nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50, angle: -45, length: 150, direction: "top" } }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</code>	Not applicable
Sets the height of the text annotation	Property: <code>nodes.annotation.height</code> <code>var nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50, } }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</code>	Property: <code>nodes.shape.annotations.height</code> <code>var node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', text: 'Left', height: 50 }] }};var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendT o('#diagram');</code>
Sets the distance between the BPMN shape and the annotation	Property: <code>nodes.annotation.length</code> <code>var nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50, length: 150 } }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</code>	Property: <code>nodes.shape.annotations.length</code> <code>var node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', length: 150, text: 'Left', }] }};var diagram = new ej.diagrams.Diagram({ nodes: [node]});diagram.appendT o('#diagram');</code>
Defines the additional information about the flow object	Property: <code>nodes.annotation.text</code> <code>var nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape" } }</code>	Property: <code>nodes.shape.annotations.text</code> <code>var node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type:</code>

in a BPMN Process	<pre>}};\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	<pre>'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ text: 'Left', }] }};var diagram = new ej.diagrams.Diagram({ nodes: [node] });diagram.appendT o('#diagram');</pre>
Sets the width of the text annotation	<p>Property: nodes.annotation.width</p> <pre>var nodes = [{ name: "node1", width: 100, height:100, offsetX:50, offsetY:50, type:"bpmn", shape: "activity", annotation: { text: "This is a BPMN Activity shape", width: 100, height: 50 } }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	<p>Property: nodes.shape.annotations.width</p> <pre>var node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', width: 45, text: 'Left', }] }};var diagram = new ej.diagrams.Diagram({ nodes: [node] });diagram.appendT o('#diagram');</pre>
Sets the id for the annotation	Not applicable	<p>Property: nodes.shape.annotations.id</p> <pre>var node = { offsetX: 250, offsetY: 250, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: true, type: 'Input' }, annotations: [{ id: 'left', text: 'Left', }] } };var diagram = new ej.diagrams.Diagram({ nodes: [node] });diagram.appendT o('#diagram');</pre>
Defines whether the group can be ungrouped or not	<p>Property: nodes.canUngroup</p> <pre>var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, borderColor: "red", borderDashArray: "4,2"};var node2 = { name: "node2", width: 50, height: 50, offsetX: 150, offsetY: 150, borderColor: "red", borderDashArray: "4,2"};var group = { name: "group", children: [node1, node2],</pre>	Not applicable

	<pre>canUngroup: false};\$("#diagramcontent").ejDiagram({ nodes: [group]}});</pre>	
<p>Array of JSON objects where each object represents a child node/connector</p>	<pre>Property:nodes.childrenvar node1 = { name: "node1", width: 50, height:50, offsetX:50, offsetY:50, borderColor: "red", borderDashArray: "4,2"};var node2 = { name: "node2", width: 50, height: 50, offsetX: 150, offsetY: 150, borderColor: "red", borderDashArray: "4,2"};var group = { name: "group", children: [node1, node2]};\$("#diagramcontent").ejDiagram({ nodes: [group]}});</pre>	<pre>Property:nodes.childrenvar node1 = { id: 'node1', offsetX: 250, offsetY: 250, width: 100, height: 100,};var node2 = { id: 'node2', offsetX: 450, offsetY: 450, width: 100, height: 100,};var group = { id: 'group',};group.children = ['node1', 'node2'];var diagram = new ej.diagrams.Diagram({ nodes: [group]}});diagram.append To('#diagram');</pre>
<p>Sets the type of UML classifier</p>	<pre>Property:nodes.classifiervar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	Not applicable
<p>Defines the name, attributes and methods of a Class. Applicable, if the node is a Class</p>	<pre>Property:nodes.classvar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", }}];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
<p>Sets the name of class</p>	<pre>Property:nodes.class.namevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", }}];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
<p>Defines the collection of attributes</p>	<pre>Property:nodes.class.attributesvar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient",</pre>	Not applicable

	<pre>attributes: [{ name: "accepted" }] }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	
Sets the name of the attribute	<pre>Property:nodes.class.attributes.namevar nodes = [{{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", attributes: [{ name: "accepted" }} }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the data type of attribute	<pre>Property:nodes.class.attributes.typevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", attributes: [{ name: "accepted", type: "Date" }} }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Defines the visibility of the attribute	<pre>Property:nodes.class.attributes.scopevar nodes = [{{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", attributes: [{ name: "accepted", type: "Date", scope:"protected" }} }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Defines the collection of methods of a Class	<pre>Property:nodes.class.methodsvar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", methods: [{ name: "getHistory" }} }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the name of the method	<pre>Property:nodes.class.methods.namevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{{name: "Date" }} }} }]];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable

Defines the arguments of the method	<pre>Property:nodes.class.methods.argumentsvar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date", type:"String" }] }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Defines the name, attributes and methods of a Class. Applicable, if the node is a Class	<pre>Property:nodes.class.methods.arguments.namevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date" }], type: "History" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the type of the argument	<pre>Property:nodes.class.methods.arguments.typevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date" }], type: "History" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the return type of the method	<pre>Property:nodes.class.methods.typevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{name: "Date" }], type: "History" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the visibility of the method	<pre>Property:nodes.class.methods.scopevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Class, "class": { name: "Patient", methods: [{ name: "getHistory", arguments: [{name: "Date" }], type: "History", scope:"protected" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable

Defines the state of the node is collapsed/expanded	<pre>Property:nodes.collapseIcon\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10 }, expandIcon: { height: 10, width: 10, shape: "ArrowDown" } }],});</pre>	<pre>Property:nodes.collapseIconvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", fill: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp" } }]});diagram.appendTo('#diagram');</pre>
Sets the border color for collapse icon of node	<pre>Property:nodes.collapseIcon.borderColor\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, borderColor: "red" }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", borderColor: "red" } }],});</pre>	<pre>Property:nodes.collapseIcon.borderColorvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", borderColor: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", borderColor: 'red' } }]});diagram.appendTo('#diagram');</pre>
Sets the border width for collapse icon of node	<pre>Property:nodes.collapseIcon.borderWidth\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, borderWidth: "2" }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", borderWidth: "2" } }],});</pre>	<pre>Property:nodes.collapseIcon.borderWidthvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", borderWidth: '2' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", borderWidth: '2' } }]});diagram.appendTo('#diagram');</pre>
Sets the fill color for collapse	<pre>Property:nodes.collapseIcon.fillColor\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup",</pre>	<pre>Property:nodes.collapseIcon.fillvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100,</pre>

icon of node	width:10, height:10, fillColor: "green" }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", fillColor: "green" } }],));	offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", fill: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", fill: 'red' } }]);diagram.appendTo('#diagram');
Defines the height for collapse icon of node	Property:nodes.collapseIcon.height\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10 }, expandIcon: { height: 10, width: 10, shape: "ArrowDown" } }],));	Property:nodes.collapseIcon.heightvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", fill: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp" } }]});diagram.appendTo('#diagram');
Sets the horizontal alignment of the icon	Property:nodes.collapseIcon.horizontalAlignment\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, horizontalAlignment:ej.datavisualization.Diagram.HorizontalAlignment.Left }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", horizontalAlignment:ej.datavisualization.Diagram.HorizontalAlignment.Left } }],));	Property:nodes.collapseIcon.horizontalAlignmentvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", horizontalAlignment:'Center' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", horizontalAlignment:'Center' } }]});diagram.appendTo('#diagram');
To set the margin for the collapse icon of node	Property:nodes.collapseIcon.margin\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, margin:{ left: 5 } }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", margin:{ left: 5 } } }],));	Property:nodes.collapseIcon.marginvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", fill: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", fill: 'red' } }]});diagram.appendTo('#diagram');

		'red', margin:{ left: 5 } }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", margin:{ left: 5 } } }]});diagram.appendTo('#diagram');
Sets the fraction/ratio(relative to node) that defines the position of the icon	Property:nodes.collapseIcon.offset\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, offset:ej.datavisualization.Diagram.Point(0,0.5) }, expandIcon: { height: 10, width: 10, shape: "ArrowDown", offset:ej.datavisualization.Diagram.Point(0,0.5) } }],});	Property:nodes.collapseIcon.offset\$("#diagram").ejDiagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", offset: { x: 0, y: 0.5 } }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", offset: { x: 0, y: 0.5 } } }]});diagram.appendTo('#diagram');
Defines the shape of the collapsed state of the node	Property:nodes.collapseIcon.shape\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10 }, expandIcon: { height: 10, width: 10, shape: "arrowdown" } }],});	Property:nodes.collapseIcon.shape\$("#diagram").ejDiagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", fill: 'red' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp" } }]});diagram.appendTo('#diagram');
Sets the vertical alignment of the icon	Property:nodes.collapseIcon.verticalAlignment\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, collapseIcon: { shape:"arrowup", width:10, height:10, verticalAlignment:ej.datavisualization.Diagram.VerticalAlignment.Top }, expandIcon: { height: 10, width: 10, shape: "arrowdown", verticalAlignment:ej.datavisualization.Diagram.VerticalAlignment.Top } }],});	Property:nodes.collapseIcon.verticalAlignment\$("#diagram").ejDiagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", verticalAlignment: 'Center' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp",

		<pre>verticalAlignment: 'Center' } }}});diagram.appendTo('# diagram');</pre>
Defines the custom content of the icon	Not applicable	<pre>Property:nodes.collapselcon.co ntentvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "Template", content: '' + '' }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp" } }}});diagram.appendTo('# diagram');</pre>
Defines the geometry of a path	Not applicable	<pre>Property:nodes.collapselcon.p athDatavar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "Path", pathData: "M0,0 L0,100" }, collapseIcon: { height: 20, width: 20, shape: "Path", pathData: "M0,0 L0,100" } }}});diagram.appendTo('# diagram');</pre>
Defines the corner radius of the icon border	Not applicable	<pre>Property:nodes.collapselcon.co rnerRadiusvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", cornerRadius: 3}, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", cornerRadius: 3 } }}});diagram.appendTo('# diagram');</pre>

Defines the space that the icon has to be moved from the icon border	Not applicable	Property: <code>nodes.collapseIcon.padding</code> <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, expandIcon: { height: 20, width: 20, shape: "ArrowDown", padding: { left: 50 } }, collapseIcon: { height: 20, width: 20, shape: "ArrowUp", padding: { left: 50 } } }]);diagram.appendTo('# diagram');</pre>
Defines the distance to be left between a node and its connections(In coming and out going connections)	Property: <code>nodes.connectorPadding</code> <pre> \$("#diagram").ej Diagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, connectorPadding: 5 }],});</pre>	Not applicable
Enables or disables the default behaviors of the node	Property: <code>nodes.constraints</code> <pre> var NodeConstraints = ej.datavisualization.Diagram.NodeConstraint s;\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, constraints: NodeConstraints.Default & ~NodeConstraints.Select }],});</pre>	Property: <code>nodes.constraints</code> <pre> var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, constraints: NodeConstraints.Default NodeConstraints.Select }]);diagram.appendTo('# diagram');</pre>
Defines how the child objects need to be arranged(Either in any predefined manner or automatically)	Property: <code>nodes.container</code> <pre> var node1 = { name: "node1", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var node2 = { name: "node2", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var group = { name: "group", children: [node1, node2], container: { type: "stack" }, offsetX: 200, offsetY: 100};\$("#diagramcontent").ejDiagram({ nodes: [group]});</pre>	Not applicable

lly). Applicable, if the node is a group		
Defines the orientation of the container. Applicable, if the group is a container	Property: nodes.container.orientation var node1 = { name: "node1", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var node2 = { name: "node2", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var group = { name: "group", children: [node1, node2], container: { type: "stack", orientation: "horizontal" }, offsetX: 200, offsetY: 100};\$("#diagramcontent").ejDiagram({ nodes: [group]});	Not applicable
Sets the type of the container. Applicable if the group is a container.	Property: nodes.container.type var node1 = { name: "node1", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var node2 = { name: "node2", width: 50, height: 50, borderColor: "red", borderDashArray: "4,2"};var group = { name: "group", children: [node1, node2], container: { type: ej.datavisualization.Diagram.ContainerType. Stack }, offsetX: 200, offsetY: 100};\$("#diagramcontent").ejDiagram({ nodes: [group]});	Not applicable
Defines the corner radius of rectangular shapes	Property: nodes.cornerRadius \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, type: "basic", shape: "rectangle", cornerRadius: 5 }], });	Not applicable
This property allows you to customize nodes appearance using user- defined CSS	Property: nodes.cssClass \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, cssClass: "hoverNode" }], });	Not applicable
Defines the BPMN data object	Property: nodes.data.type \$("#diagram").ejDiagram({ nodes: [{ name: "dataobject", type: "bpmn", shape: ej.datavisualization.Diagram.BPMNShapes.DataObject, data: { type: ej.datavisualization.Diagram.BPMNDataObject	Property: nodes.shape.dataObject.type var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, shape: { type:

	<pre>s.Input }, width: 50, height: 50, offsetX: 100, offsetY: 100}}]);});</pre>	<pre>'Bpmn', shape: 'DataObject', dataObject: { collection: false, type: 'Input' } } }]);diagram.appendTo('#diagram');</pre>
Defines whether the BPMN data object is a collection or not	<pre>Property:nodes.data.collection\$("#diagram").ejDiagram({ nodes: [{ name: "dataobject", type: "bpmn", shape: ej.datavisualization.Diagram.BPMNShapes.DataObject, data: { type: ej.datavisualization.Diagram.BPMNDataObject.s.Input, collection: false }, width: 50, height: 50, offsetX: 100, offsetY: 100}}]);});</pre>	<pre>Property:nodes.shape.dataObject.collectionvar diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, shape: { type: 'Bpmn', shape: 'DataObject', dataObject: { collection: false, type: 'Input' } } }]});diagram.appendTo('#diagram');</pre>
Defines an Enumeration in a UML Class Diagram	<pre>Property:nodes.enumeration\$("#diagram").ejDiagram({ nodes : [{ name: "Enums", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes Enumeration, enumeration: { name: "AccountType", } }]});</pre>	Not applicable
Sets the name of the Enumeration	<pre>Property:nodes.enumeration.name\$("#diagram").ejDiagram({ nodes : [{ name: "Enums", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes Enumeration, enumeration: { name: "AccountType", } }]});</pre>	Not applicable
Defines the collection of enumeration members	<pre>Property:nodes.enumeration.members\$("#diagram").ejDiagram({ nodes : [{ name: "Enums", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes Enumeration, enumeration: { name: "AccountType", members: [{ name: "CheckingAccount" }] } }]});</pre>	Not applicable
Sets the name of the	<pre>Property:nodes.enumeration.members.name\$("#diagram").ejDiagram({ nodes : [{ name: "Enums", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type:</pre>	Not applicable

enumeration member	<pre>umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.Enumeration, enumeration: { name: "AccountType", members: [{ name: "CheckingAccount"}] }]]]);</pre>	
Sets the type of the BPMN Events. Applicable, if the node is a BPMN event	<pre>Property:nodes.event\$("#diagram").ejDiagram({ nodes : [{ name: "nodeEvent", type: "bpmn", shape: "event", event: ej.datavisualization.Diagram.BPMNEvents.Intermediate, width: 50, height: 50}]]]);</pre>	<pre>Property:nodes.shape.eventvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'Event', event: { event: 'Start', trigger: 'None' } } }]]});diagram.appendTo('# diagram');</pre>
Defines the type of the trigger	<pre>Property:nodes.event.trigger\$("#diagram").ejDiagram({ nodes : [{ name: "nodeEvent", type: "bpmn", shape: ej.datavisualization.Diagram.BPMNShapes.Event, trigger: ej.datavisualization.Diagram.BPMNTriggers.None width: 50, height: 50}]]]);</pre>	<pre>Property:nodes.shape.event.triggervar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, shape: { type: 'Bpmn', shape: 'Event', event: { event: 'Start', trigger: 'None' } } }]]});diagram.appendTo('# diagram');</pre>
Defines whether the node can be automatically arranged using layout or not	<pre>Property:nodes.excludeFromLayoutvar node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, excludeFromLayout: true};var node2 = { name: "node2", width: 50, height: 50};var node3 = { name: "node3", width: 50, height: 50};\$("#diagramcontent").ejDiagram({ nodes: [node1, node2, node3], layout: { type: "hierarchicaltree" }]);</pre>	<pre>Property:nodes.excludeFromLayoutvar diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', offsetX: 100, offsetY: 100, width: 100, height: 100, excludeFromLayout: true, }, { id: 'node1', width: 70, height: 70, annotations: [{ content: 'node1' }] }, { id: 'node2', width: 70, height: 70, annotations: [{ content: 'node2' }] }];, layout: { type: 'RadialTree', }, });diagram.appendTo('#diagram');</pre>
Defines the fill color of the node	<pre>Property:nodes.fillColorvar node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, fillColor:"red"};\$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<pre>Property:nodes.style.fillvar diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', offsetX: 100, offsetY: 100, width: 100, height:</pre>

		<pre>100, style: { fill: 'red' } },,));diagram.appendTo('#diagram');</pre>
<p>Sets the type of the BPMN Gateway. Applicable, if the node is a BPMN gateway</p>	<p>Property:nodes.gateway</p> <pre>var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, type: "bpmn", shape: "gateway" , gateway: ej.datavisualization.Diagram.BPMNGateways.Exclusive }; \$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<p>Property:nodes.shape.gateway</p> <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, shape: { type: 'Bpmn', shape: 'Gateway', gateway: { type: 'Exclusive' } } }],});diagram.appendTo('#diagram');</pre>
<p>Paints the node with linear color transitions</p>	<p>Property:nodes.gradient.type</p> <pre>var gradient = { type: "linear", x1: 0, x2: 50, y1: 0, y2: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 50 }]; var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, gradient : gradient }; \$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<p>Property:nodes.style.gradient.type</p> <pre>var stopscol = [];var stops1 = { color: 'white', offset: 0};stopscol.push(stops1) ;var stops2 = { color: 'red', offset: 50};stopscol.push(stops2);var gradient1 = { x1: 0, x2: 50, y1: 0, y2: 50, stops: stopscol, type: 'Linear'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient1 } }],});diagram.appendTo('#diagram');</pre>
<p>Defines the x1 value of linear gradient</p>	<p>Property:nodes.gradient.LinearGradient.x1</p> <pre>var gradient = { type: "linear", x1: 0, x2: 50, y1: 0, y2: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 50 }]; var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, gradient : gradient }; \$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<p>Property:nodes.style.gradient.LinearGradient.x1</p> <pre>var stopscol = [];var stops1 = { color: 'white', offset: 0};stopscol.push(stops1) ;var stops2 = { color: 'red', offset: 50};stopscol.push(stops2);var gradient1 = { x1: 0, x2: 50, y1: 0, y2: 50, stops: stopscol, type: 'Linear'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node',</pre>

		width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient1 } }],));diagram.appendTo('#diagram');
Defines the x2 value of linear gradient	Property:nodes.gradient.LinearGradient.x2var gradient = { type: "linear", x1: 0, x2: 50, y1: 0, y2: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 50 }]};var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, gradient : gradient };\$("#diagramcontent").ejDiagram({ nodes: [node1],});	Property:nodes.style.gradient.LinearGradient.x2var stopscol = [];var stops1 = { color: 'white', offset: 0};stopscol.push(stops1);var stops2 = { color: 'red', offset: 50};stopscol.push(stops2);var gradient1 = { x1: 0, x2: 50, y1: 0, y2: 50, stops: stopscol, type: 'Linear'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient1 } }],});diagram.appendTo('#diagram');
Defines the y1 value of linear gradient	Property:nodes.gradient.LinearGradient.y1var gradient = { type: "linear", x1: 0, x2: 50, y1: 0, y2: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 50 }]};var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, gradient : gradient };\$("#diagramcontent").ejDiagram({ nodes: [node1],});	Property:nodes.style.gradient.LinearGradient.y1var stopscol = [];var stops1 = { color: 'white', offset: 0};stopscol.push(stops1);var stops2 = { color: 'red', offset: 50};stopscol.push(stops2);var gradient1 = { x1: 0, x2: 50, y1: 0, y2: 50, stops: stopscol, type: 'Linear'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient1 } }],});diagram.appendTo('#diagram');
Defines the y2 value of	Property:nodes.gradient.LinearGradient.y2var gradient = { type: "linear", x1: 0, x2: 50, y1: 0, y2: 50, stops: [{ color: "white",	Property:nodes.style.gradient.LinearGradient.y2var

linear gradient	<pre>offset: 0 }, { color: "red", offset: 50 }]]];var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, gradient : gradient };\$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<pre>stopscol = [];var stops1 = { color: 'white', offset: 0};stopscol.push(stops1) ;var stops2 = { color: 'red', offset: 50};stopscol.push(stops2);var gradient1 = { x1: 0, x2: 50, y1: 0, y2: 50, stops: stopscol, type: 'Linear'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient1 } }],});diagram.appendTo('#diagram');</pre>
Defines the type of gradient	<pre>Property:nodes.gradient.RadialGradient.typevar node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }]}};\$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<pre>Property:nodes.style.gradient.typevar stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram');</pre>
Defines the position of the outermost circle	<pre>Property:nodes.gradient.RadialGradient.cxvar node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }]}};\$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	<pre>Property:nodes.style.RadialGradient.cxvar stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});</pre>

		<pre>]],));diagram.appendTo('#diagram'); </pre>
Defines the outer most circle of the radial gradient	<pre> Property:nodes.gradient.RadialGradient.cyvar node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node1],}); </pre>	<pre> Property:nodes.style.RadialGradient.cyvar stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram'); </pre>
Defines the innermost circle of the radial gradient	<pre> Property:nodes.gradient.RadialGradient.fxvar node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node1],}); </pre>	<pre> Property:nodes.style.RadialGradient.fxvar stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram'); </pre>
Defines the innermost circle of the radial gradient	<pre> Property:nodes.gradient.RadialGradient.fyvar node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node1],}); </pre>	<pre> Property:nodes.style.RadialGradient.fyvar stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram'); </pre>

		<pre>100, style: { gradient: gradient } }],));diagram.appendTo('#diagram');</pre>
Defines the different colors and the region of color transitions	Property: nodes.gradient.RadialGradient.stops <pre>var node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node],});</pre>	Property: nodes.style.RadialGradient.stops <pre>var stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram');</pre>
Sets the color to be filled over the specified region	Property: nodes.gradient.stops.color <pre>var node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node],});</pre>	Property: nodes.style.gradient.stops.color <pre>var stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram');</pre>
Sets the position where the previous color transition ends and a new color transition starts	Property: nodes.gradient.stops.offset <pre>var node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100 }] }};\$("#diagramcontent").ejDiagram({ nodes: [node],});</pre>	Property: nodes.style.gradient.stops.offset <pre>var stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node',</pre>

		width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],));diagram.appendTo('#diagram');
Describes the transparency level of the region	Property:nodes.gradient.stops.opacity <pre>var node = { name: "node", width: 50, height: 50, offsetX: 100, offsetY: 100, gradient: { type: "radial", fx: 50, fy: 50, cx: 50, cy: 50, stops: [{ color: "white", offset: 0 }, { color: "red", offset: 100, opacity: 0.5 }] } };\$("#diagramcontent").ejDiagram({ nodes: [node1],});</pre>	Property:nodes.style.gradient.stops.opacity <pre>var stops = [{ color: 'white', offset: 0}, { color: 'red', offset: 50, opacity: 0.5}];var gradient = { cx: 50, cy: 50, fx: 50, fy: 50, stops: stops, type: 'Radial'};var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, style: { gradient: gradient } }],});diagram.appendTo('#diagram');</pre>
Defines the header of a swimlane/lane	Property:nodes.header <pre>var swimlane = { type: "swimlane", name: "swimlane", header: { text: "Swimlane", fontSize: 12, bold: true } };\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
Defines the height of the node	Property:nodes.height <pre>\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, }],});</pre>	Property:nodes.height <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');</pre>
Sets the horizontal alignment of the node. Applicable, if the parent of the node is a container	Property:nodes.horizontalAlign <pre>var node1 = { name: "node1", width: 50, height: 50};var node2 = { name: "node2", width: 50, height: 50, horizontalAlign: ej.datavisualization.Diagram.HorizontalAlign ment.Right};var group = { name: "group", children: [node1, node2], container: { type: "canvas" }, offsetX: 200, offsetY: 100, minWidth: 200, minHeight: 200, fillColor: "red"};\$("#diagramcontent").ejDiagram({ nodes: [group]});</pre>	Not applicable
A read only collection	Property:nodes.inEdges <pre>var node = diagram.selectionList[0];for(var i = 0; i <</pre>	Property:nodes.inEdges <pre>var diagram = new</pre>

of the incoming connectors /edges of the node	<pre>node.inEdges.length; i++){ console.log(node.inEdges[i]);};</pre>	<pre>ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('# diagram');var node = (diagram.nodes[0] as Node).inEdges;for (var i = 0; i < node.length; i++) {console.log(node[i]);};</pre>
Defines an interface in a UML interface Diagram	<pre>Property: nodes.interfacevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface }];\$("#diagramcontent").ejDiagram({ nodes:nodes });</pre>	Not applicable
Defines the name, attributes and methods of a Interface. Applicable, if the node is a Interface	<pre>Property: nodes.interface.namevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Defines the collection of attributes	<pre>Property: nodes.interface.attributesvar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", attributes: [{ name: "accepted"}] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the name of the attribute	<pre>Property: nodes.interface.attributes.namevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", attributes: [{ name: "accepted" }]} }];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable

Sets the data type of attribute	Property: nodes.interface.attributes.type var nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.interface, "interface": { name: "Patient", attributes: [{ name: "accepted", type: "Date" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});	Not applicable
Defines the visibility of the attribute	Property: nodes.interface.attributes.scope var nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.interface, "interface": { name: "Patient", attributes: [{ name: "accepted", type: "Date", scope: "protected" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});	Not applicable
Defines the collection of methods of a interface	Property: nodes.interface.methods var nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.interface, "interface": { name: "Patient", methods: [{ name: "getHistory" }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});	Not applicable
Sets the name of the method	Property: nodes.interface.methods.name var nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date" }] }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});	Not applicable
Defines the arguments of the method	Property: nodes.interface.methods.arguments var nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShapes.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date", type: "String" }] }] } }];\$("#DiagramContent").ejDiagram({ nodes: nodes});	Not applicable
Defines the name,	Property: nodes.interface.methods.arguments.name var nodes = [{ name: "Patient", offsetX:	Not applicable

attributes and methods of a interface. Applicable, if the node is a interface	<pre>100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date" }], type: "History" }] } }};\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	
Sets the type of the argument	<pre>Property:nodes.interface.methods.typevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{ name: "Date" }], type: "History" }] }}];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the return type of the method	<pre>Property:nodes.interface.methods.typevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{name: "Date" }], type: "History" }] }}];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Sets the visibility of the method	<pre>Property:nodes.interface.methods.scopevar nodes = [{ name: "Patient", offsetX: 100, offsetY: 100, borderWidth: 2, borderColor: "black", type: "umlclassifier", classifier: ej.datavisualization.Diagram.ClassifierShap es.interface, "interface": { name: "Patient", methods: [{ name: "getHistory", arguments: [{name: "Date" }], type: "History", scope:"protected" }] }}];\$("#DiagramContent").ejDiagram({ nodes: nodes});</pre>	Not applicable
Defines whether the sub tree of the node is expanded or collapsed	<pre>Property:nodes.isExpandedvar node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, isExpanded: false};var node2 = { name: "node2", width: 50, height: 50};var connector = { sourceNode: "node1", targetNode: "node2", name: "connector"};\$("#diagramcontent").ejDiagram ({ nodes: [node1, node2], connectors:</pre>	<pre>Property:nodes.isExpandedvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, id: 'node1', isExpanded: true, }, { id: 'node2', width: 50, height: 50</pre>

	<pre>[connector], layout: { type: "hierarchicaltree" }));</pre>	<pre>}}, connectors: [{ sourceNode: 'node1', targetNode: 'node2', id: 'connector' }], layout: { type: "hierarchicaltree" });diagram.appendTo('#d iagram');</pre>
Sets the node as a swimlane	<pre>Property:nodes.isSwimlanevar swimlane = { type: "swimlane", name: "swimlane", isSwimlane: true, header: { text: "Swimlane", fontSize: 12, bold: true }};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
A collection of objects where each object represents a label	<pre>Property:nodes.labels\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", fontColor: "Red" }] }],});</pre>	<pre>Property:nodes.annotationsva r diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation' }] }]);diagram.appendTo('# diagram');</pre>
An array of objects where each object represents a lane. Applicable, if the node is a swimlane	<pre>Property:nodes.lanesvar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200 }, { name: "lane2", width: 100 }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
This property allows you to customize lanes appearance using user-defined CSS	<pre>Property:nodes.lanes.cssClassvar addInfo = { Description:"Describe the functionality" };var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200 }, { name: "lane2", width: 100, cssClass:"hoverLane", addInfo: addInfo, fillColor:"lightgrey" }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
Defines the header of the lane	<pre>Property:nodes.lanes.headervar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200 }, { name: "lane2", width: 100, header: { fillColor:"blue",</pre>	Not applicable

	<code>fontColor:"white", text:"Function 1" } }]];\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	
Defines the width of lane	Property: nodes.lanes.width <code>var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200, height: 200, zOrder:10 }, { name: "lane2", width: 100 } }];\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	Not applicable
An array of objects where each object represents a child node of the lane	Property: nodes.lanes.children <code>var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200 }, { name: "lane2", width: 100, children:[{name:"process", width: 50, height: 50 }] }];\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	Not applicable
Defines the object as a lane	Property: nodes.lanes.isLane <code>var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200, height: 200, isLane:true, orientation:"vertical" }, { name: "lane2", width: 100 } }];\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	Not applicable
Defines the minimum space to be left between the bottom of parent bounds and the node	Property: nodes.margin <code>var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 200, lanes: [{ name: "lane1", width: 200, children: [{ name: "process", width: 50, height: 50, marginBottom: 50, marginLeft: 10, marginRight: 10, marginTop: 10 }] }] }];\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	Property: nodes.margin <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, margin : { left: 15, right: 15, top: 15, bottom: 15 } }] });diagram.appendTo('# diagram');</code>
Defines the maximum height limit of the node	Property: nodes.maxHeight <code>var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, maxHeight: 100, maxWidth: 100, minHeight: 10, minWidth: 10 }];\$("#diagramcontent").ejDiagram({ nodes: nodes});</code>	Property: nodes.maxHeight <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, maxHeight: 100, maxWidth: 100, minHeight: 10, minWidth: 10 }] });diagram.appendTo('# diagram');</code>

Sets the unique identifier of the node	Property: nodes.name var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, },];\$("#diagramcontent").ejDiagram({ nodes: nodes});	Property: nodes.id var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node1' offsetX: 100, offsetY: 100, width: 100, height: 100, }]});diagram.appendTo('#diagram');
Defines the opaque of the node	Property: nodes.opacity var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, opacity: 0.5, rotateAngle: 70}];\$("#diagramcontent").ejDiagram({ nodes: nodes});	Property: nodes.style.opacity var diagram = new ej.diagrams.Diagram({ nodes: [{ id: 'node1' offsetX: 100, offsetY: 100, width: 100, height: 100, rotateAngle: 70, style: { opacity: 0.5 } }]});diagram.appendTo('#diagram');
Defines the minimum padding value to be left between the bottom most position of a group and its children. Applicable, if the group is a container	Property: nodes.paddingBottom var node1 = { name: "node1", width: 50, height: 50};var node2 = { name: "node2", width: 50, height: 50, verticalAlign: "bottom"};var group = { name: "group", children: [node1, node2], container: { type: "canvas" }, offsetX: 200, offsetY: 100, fillColor: "gray", minWidth: 200, minHeight: 200, paddingBottom: 10, paddingLeft: 10, paddingRight: 10, paddingTop: 10};\$("#diagramcontent").ejDiagram({nodes:[group]});	Not applicable
Sets the name of the parent group	Property: nodes.parent var node1 = { name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, parent: "group"};var node2 = { name: "node2", width: 50, height: 50, offsetX: 150, offsetY: 150, parent: "group"};var group = { name: "group", children: ["node1", "node2"]};\$("#diagramcontent").ejDiagram({ nodes: [node1, node2, group]});	Not applicable
Sets the path geometry that defines the	Property: nodes.pathData var nodes;nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, type: "basic", shape: "path", pathData: "M 370.9702,194.9961 L 359.5112,159.7291 L	Property: nodes.shape.data var nodes = [{ id: 'node1', width: 100, height: 100, offsetX: 300, offsetY: 100, shape: { type:

shape of a path node	<pre>389.5112,137.9341 L 419.5112,159.7291 L 408.0522,194.9961 L 370.9702,194.9961 z"}];\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>'Path', data: 'M 540.3643,137.9336 L 546.7973,159.7016 L 570.3633,159.7296 L 550.7723,171.9366 L 558.9053,194.9966 L 540.3643,179.4996 L 521.8223,194.9966 L 529.9553,171.9366 L 510.3633,159.7296 L 533.9313,159.7016 L 540.3643,137.9336 z' }, },];var diagram = new ej.diagrams.Diagram({ width: '800px', height: '500px', nodes: nodes});</pre>
An array of objects, where each object represents a smaller region(phase) of a swimlane	<pre>Property:nodes.phasesvar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 100, width: 300, orientation: "horizontal", phases: [{ name: "phase1", offset: 150, label: { text: "Phase1" } }, { name: "phase2", label: { text: "Phase2" } }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
Defines the header of the smaller regions	<pre>Property:nodes.phases.labelvar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 100, width: 300, orientation: "horizontal", phases: [{ name: "phase1", offset: 150, label: { text: "Phase1" } }, { name: "phase2", label: { text: "Phase2" } }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
Defines the line color of the splitter that splits adjacent phases	<pre>Property:nodes.phases.lineColorvar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 100, width: 300, orientation: "horizontal", phases: [{ name: "phase1", offset: 150, label: { text: "Phase1" }, lineColor:"green", lineDashArray:"2,2", lineWidth:3 }, { name: "phase2", label: { text: "Phase2" } }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable
Sets the length of the smaller region(phase) of a swimlane	<pre>Property:nodes.phases.offsetvar swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 100, width: 300, orientation: "horizontal", phases: [{ name: "phase1", offset: 150, label: { text: "Phase1" } }, { name: "phase2", label: { text: "Phase2" } }]};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</pre>	Not applicable

	<code>}};\$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	
Sets the orientation of the phase	Property: <code>nodes.phases.orientation</code> <code>var diagram = \$("#diagramcontent").ejDiagram("instance"); diagram.addPhase(diagram.selectionList[0].name, { name: "verticalPhase", type: "phase", offset: 200, orientation: "vertical", label: { text: "New Phase" } });</code>	Not applicable
Sets the height of the phase headers	Property: <code>nodes.phaseSize</code> <code>var swimlane = { type: "swimlane", name: "swimlane", offsetX: 300, offsetY: 100, width: 300, orientation: "horizontal", phaseSize: 50, phases: [{ name: "phase1", offset: 150, label: { text: "Phase 1" } }]}; \$("#diagramcontent").ejDiagram({ nodes: [swimlane]});</code>	Not applicable
Sets the ratio/fractional value relative to node, based on which the node will be transformed (positioning, scaling and rotation)	Property: <code>nodes.pivot</code> <code>var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, pivot: { x: 0, y: 0 } }]; \$("#diagramcontent").ejDiagram({ nodes: nodes});</code>	Property: <code>nodes.pivot</code> <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, pivot: { x: 0, y: 0 } }]}); diagram.appendTo('#diagram');</code>
Defines a collection of points to draw a polygon. Applicable, if the shape is a polygon	Property: <code>nodes.points</code> <code>var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, type: "basic", shape: "polygon", points: [{ x: 0, y: 12.5 }, { x: 0, y: 50 }, { x: 50, y: 50 }, { x: 50, y: 0 }, { x: 12.5, y: 0 }, { x: 0, y: 12.5 }]}]; \$("#diagramcontent").ejDiagram({ nodes: nodes});</code>	Property: <code>nodes.shape.points</code> <code>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, shape: { type: 'Basic', shape: 'Polygon', points: [{ x: 35, y: 0 }, { x: 65, y: 0 }, { x: 100, y: 35 }, { x: 100, y: 65 }, { x: 65, y: 100 }, { x: 35, y: 100 }, { x: 0, y: 65 }, { x: 0, y: 35 }] } }]}); diagram.appendTo('#diagram');</code>
An array of objects	Property: <code>nodes.ports</code> <code>var nodes = [{ name: "node1", width: 50, height: 50, offsetX:</code>	Property: <code>nodes.ports</code> <code>var port = [{ id: 'port1',</code>

where each object represents a port	<pre>50, offsetY: 50, ports: [{ name: "port1", offset: { x: 0.5, y: 0 } }, { name: "port2", offset: { x: 0.5, y: 1 } }]}};\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>visibility: PortVisibility.Visible, shape: 'Circle', offset: { x: 0, y: 0 }}];var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, ports: port }]]);diagram.appendTo('# diagram');</pre>
Sets the border color of the port	<pre>Property:nodes.ports.borderColorvar nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, ports: [{ name: "port1", offset: { x: 0.5, y: 0 }, borderColor:"yellow", borderWidth: 3, cssClass:"hoverPort", fillColor:"red", size: 10, visibility:ej.datavisualization.Diagram.Port tVisibility.Visible }, { name: "port2", offset: { x: 0.5, y: 1 } }]}};\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>Property:nodes.ports.style.stro keColorvar port = [{ id: 'port1', visibility: PortVisibility.Visible, shape: 'Circle', offset: { x: 0, y: 0 }, style: { strokeColor: 'yellow', strokeDashArray: '2 2', strokeWidth: 4, fill:'red', opacity: 0.5 }]]];var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, ports: port }]]);diagram.appendTo('# diagram');</pre>
Defines whether connections can be created with the port	<pre>Property:nodes.ports.constraintsvar nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, ports: [{ name: "port1", offset: { x: 0.5, y: 0 } }, { name: "port2", offset: { x: 0.5, y: 1 } }]}};\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>Property:nodes.ports.constrain tsvar port = [{ id: 'port1', visibility: PortVisibility.Visible, shape: 'Circle', offset: { x: 0, y: 0 }, constraints: PortConstraints.Draw}}];v ar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, ports: port }]]);diagram.appendTo('# diagram');</pre>
An array of objects where each object	<pre>Property:nodes.ports.shapevar nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, ports: [{ name: "port1", offset: { x: 0.5, y: 0 }, shape:ej.datavisualization.Diagram.PortShap es.Path, pathData: "M5,0 L10,10 L0,10 z" }, { name: "port2", offset: { x: 0.5, y: 1 } }</pre>	<pre>Property:nodes.ports.shapeva r port = [{ id: 'port1', visibility: PortVisibility.Visible, shape: 'Path', pathData: 'M5,0 L10,10 L0,10 z' offset: { x: 0, y: 0</pre>

represents a port	<pre>}}];\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>}}];var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, ports: port }]});diagram.appendTo('#diagram');</pre>
Sets the vertical alignment of the port with respect to its immediate parent	Not applicable	Property: nodes.ports.verticalAlignment <pre>var port = [{ id: 'port1', visibility: PortVisibility.Visible, shape: 'Circle', offset: { x: 0, y: 0 }, verticalAlignment: 'Top', horizontalAlignment: 'Left', width: 10, height: 10}];var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, ports: port }]});diagram.appendTo('#diagram');</pre>
Defines the opacity and the position of shadow	Property: nodes.shadow <pre>var nodes = [{ name: "node1", width: 50, height: 50, offsetX: 50, offsetY: 50, shadow: { opacity: 0.5, distance: 10, angle: 45 } }];\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	Property: nodes.shadow <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, shadow: { opacity: 0.5, distance: 10, angle: 45 } }]});diagram.appendTo('#diagram');</pre>
Defines the sub process of a BPMN Activity. Applicable, if the type of the BPMN activity is sub process	Property: nodes.subProcess <pre>var nodes = [{ name: "node1", width: 100, height: 100, offsetX: 50, offsetY: 50, type: "bpmn", shape: "activity", activity: "subprocess", subProcess: { loop: ej.datavisualization.Diagram.BPMNLoops.Standard, adhoc: true, boundary: ej.datavisualization.Diagram.BPMNBoundary.Collapsible, compensation: true, collapsed: false } }];\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	Property: nodes.shape.activity.subProcess <pre>var nodes = [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'SubProcess', subProcess: { adhoc: false, boundary: 'Default', collapsed: true } } } }];var diagram = new ej.diagrams.Diagram({</pre>

		width: '100%', height: '600px', nodes: nodes});diagram.appendTo('#diagram');
Defines the collection of events that need to be appended with BPMN Sub-Process	Property:nodes.subProcess.events var nodes = [{ name: "node1", width: 100, height: 100, offsetX: 50, offsetY: 50, type: "bpmn", shape: "activity", activity: "subprocess", subProcess: { type: "transaction", events: [{ name:"intermediate1", event: "intermediate", offset: { x: 0.25, y: 1 } }, { event: "intermediate", trigger: "error", offset: { x: 0.75, y: 1 } }] } }];\$("#diagramcontent").ejDiagram({ nodes: nodes});	Property:nodes.shape.activity.subProcess.events var nodes = [{ id: 'node1', width: 190, height: 190, offsetX: 300, offsetY: 200, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'SubProcess', subProcess: { type: 'Event', loop: 'ParallelMultiInstance', compensation: true, adhoc: false, boundary: 'Event', collapsed: true, events: [{ height: 20, width: 20, offset: { x: 0, y: 0 }, margin: { left: 10, top: 10 }, horizontalAlignment: 'Left', verticalAlignment: 'Top', event: 'Intermediate', trigger: 'Error' }] } } } }]};var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes});diagram.appendTo('#diagram');
An array of objects where each object represents a port	Property:nodes.subProcess.events.ports var nodes;nodes = [{ name: "node1", width: 100, height: 100, offsetX: 50, offsetY: 50, type: "bpmn", shape: "activity", activity: "subprocess", subProcess: { type: "transaction", events: [{ event: "intermediate", offset: { x: 0.25, y: 1 }, ports: [{ name: "port1", offset: { x: 0.5, y: 0 } }, { name: "port2", offset: { x: 0.5, y: 1 } }] } }] }];\$("#diagramcontent").ejDiagram({ nodes: nodes});	Property:nodes.shape.activity.subProcess.events.ports var nodes = [{ id: 'node1', width: 190, height: 190, offsetX: 300, offsetY: 200, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'SubProcess', subProcess: { type: 'Event', loop: 'ParallelMultiInstance', compensation: true, adhoc: false, boundary: 'Event', collapsed: true, events: [{ height: 20, width: 20, offset: { x: 0, y: 0 }, margin: { left: 10, top: 10 },

		<pre>horizontalAlignment: 'Left', verticalAlignment: 'Top', ports: [{ id: 'port1', visibility: PortVisibility.Visible, shape: 'Circle', offset: { x: 0, y: 0 } }], event: 'Intermediate', trigger: 'Error' }] } } } }];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes});diagram.appendTo ('#diagram');</pre>
A collection of objects where each object represents a label	<pre>Property:nodes.subProcess.events.labelsvar label = [];label = { "text": "Node1", "fontColor": "Red"};var nodes = [{ name: "node1", width: 100, height: 100, offsetX: 50, offsetY: 50, type: "bpmn", shape: "activity", activity: "subprocess", subProcess: { type: "transaction", events: [{ event: "intermediate", offset: { x: 0.25, y: 1 }, {labels:label} }] } }];\$("#diagramcontent").ejDiagram({ nodes: nodes});</pre>	<pre>Property:nodes.shape.activity.s ubProcess.events.annotations var nodes = [{ id: 'node1', width: 190, height: 190, offsetX: 300, offsetY: 200, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'SubProcess', subProcess: { type: 'Event', loop: 'ParallelMultiInstance', compensation: true, adhoc: false, boundary: 'Event', collapsed: true, events: [{ height: 20, width: 20, offset: { x: 0, y: 0 }, margin: { left: 10, top: 10 }, horizontalAlignment: 'Left', verticalAlignment: 'Top', annotations: [{ id: 'label3', margin: { bottom: 10 }, horizontalAlignment: 'Center', verticalAlignment: 'Top', content: 'Event', offset: { x: 0.5, y: 1 }, style: { color: 'black', fontFamily: 'Fantasy', fontSize: 8 } }], event: 'Intermediate', trigger: 'Error' }] } } }];var diagram = new ej.diagrams.Diagram({</pre>

		width: '100%', height: '600px', nodes: nodes});diagram.appendTo('#diagram');
Defines the task of the BPMN activity. Applicable, if the type of activity is set as task	Property: nodes.task var nodes = [{ name: "node1", width: 100, height: 100, offsetX: 50, offsetY: 50, type: "bpmn", shape: "activity", activity: "task", task: { compensation: true, call: true, loop: ej.datavisualization.Diagram.BPMNLoops.Standard, type: ej.datavisualization.Diagram.BPMNTasks.Service } }];\$("#diagramcontent").ejDiagram({ nodes: nodes});	Property: nodes.shape.activity.task var nodes = [{ id: 'node', width: 100, height: 100, offsetX: 100, offsetY: 100, shape: { type: 'Bpmn', shape: 'Activity', activity: { activity: 'Task', task: { call: true, compensation: false, type: 'Service', loop: 'ParallelMultiInstance' } } } }];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', nodes: nodes});diagram.appendTo('#diagram');diagram.tool = DiagramTools.ZoomPan;

NodeTemplate

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Binds the custom JSON data with node properties	Property: nodeTemplate var data = [{ "Id": "E1", "Name": "Maria Anders", "Designation": "Managing Director" }, { "Id": "E2", "Name": "Ana Trujillo", "Designation": "Project Manager", "ReportingPerson": "E1" }];\$("#diagramcontent").ejDiagram({ dataSourceSettings: { id: "Id", parent: "ReportingPerson", dataSource: data }, nodeTemplate: "nodeTemplate"});function nodeTemplate(diagram, node) { node.labels[0].text = node.Name;}	Property: setNodeTemplate function setNodeTemplate() { setNodeTemplate: (object: NodeModel, diagram: Diagram): StackPanel => { if (object.id === 'node2') { var table = new StackPanel(); table.orientation = 'Horizontal'; var column1 = new StackPanel(); column1.children = []; column1.children.push(getTextElement('Column1')); addRows(column1); var column2 = new StackPanel(); column2.children = []; column2.children.push(getTextElement('Column2')); addRows(column2); table.children = [column1, column2]; return table; } return null; }}var getTextElement = (text: string) => { var textElement = new TextElement(); textElement.width = 50; textElement.height = 20; textElement.content = text; return textElement;};var nodes = [{ id: 'node1',

		<pre>height: 100, offsetX: 100, offsetY: 100, }, { id: 'node2', width: 100, height: 100, offsetX: 300, offsetY: 100 }];var connectors = [{ id: 'connector1', type: 'Straight', sourcePoint: { x: 100, y: 300 }, targetPoint: { x: 200, y: 400 }},];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', connectors: connectors, nodes: nodes, setNodeTemplate: setNodeTemplate, });diagram.appendTo('#diagram');</pre>
--	--	---

Overview

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Allows to see a preview or an overall view of the entire content of a Diagram	Property:overview <pre>; \$("#diagram").ejDiagram({ width: "100%", height: "600px"});\$("#overview").ejOvervi ew({ // Relates Diagram with overview sourceID: "diagram", width: "100%", height: "100%" });</pre>	Property:Overview // Initializes the diagram control <pre>var diagram = new ej.diagrams.Diagram({ width: "100%", height: "600px"});diagram.appendTo('#diagram');var overview = new ej.diagrams.Overview({ width: '100%', height: '150px', sourceID: 'diagram' });overview.appendTo('#overview');</pre>

Layers

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
A collection of JSON objects where each object represents a layer. Layer is a named category of	Property:layers <pre>var layer = { name: "layer1", active: true, visible: true, print: true, lock: true, objects: ["textNode"]};\$("#diagram").ejDiagra m({ layers: [layer]});</pre>	Property:layers <pre>var nodes = [{ id: 'node1', width: 100, height: 100, offsetX: 100, offsetY: 100, }, { id: 'node2', width: 100, height: 100, offsetX: 300, offsetY: 100, shape: { type: 'Basic', shape: 'Ellipse' }, }];var connectors = [{ id: 'connector1', type: 'Straight', sourcePoint: { x: 100, y: 300 }, targetPoint: { x: 200, y: 400 },}];var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', connectors: connectors, nodes:</pre>

diagram shapes		<pre>nodes, layers: [{ id: 'layer1', visible: true, lock: true, zIndex: -1, objects: ['node1', 'node2', 'connector1'] }],});diagram.appendTo('#diagram ');</pre>
A collection of JSON objects where each object represents a layer. Layer is a named category of diagram shapes	Property:layers.print <pre>var layer = { name: "layer1", active: true, visible: true, print: true, lock: true, objects: ["textNode"] };\$("#diagram").ejDiagram({ layers: [layer]});</pre>	Not applicable

Annotations

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
A collection of objects where each object represents a label	Property:labels.text <pre>\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label" }] }],});</pre>	Property:annotations.content <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation' }] }]);diagram.appendTo('# diagram');</pre>
Offset for annotation content	Property:labels.offset <pre>\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", offset: { x: 0, y: 1}, rotateAngle: 90 }] }],});</pre>	Property:annotations.offset <pre>var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', offset: { x: 0, y: 1}, rotateAngle: 90 }] }]);diagram.appendTo('# diagram');</pre>

Sets the hyperlink for the labels	<pre>Property:labels.hyperText\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", bold:true, "hyperText": "https://www.syncfusion.com" }] }],});</pre>	<pre>Property:annotations.hyperlinkvar style = { bold: true };var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: style, hyperlink: { color: 'red', content: 'HRPortal', textDecoration: 'Overline', link: 'https://hr.syncfusion.com/home' } }] }]);diagram.appendTo('#diagram');</pre>
Enables/disables the bold style	<pre>Property:labels.bold\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", bold:true }] }],});</pre>	<pre>Property:annotations.style.boldvar style = { bold: true };var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: style }] }] }]);diagram.appendTo('#diagram');</pre>
Sets the border color of the label	<pre>Property:labels.borderColor\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", borderColor:"red", borderWidth: 2 }] }],});</pre>	<pre>Property:annotations.style.borderColorvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { strokeColor: 'black', strokeWidth: 2 } }] }] }]);diagram.appendTo('#diagram');</pre>
Sets the border width of the label	<pre>Property:labels.borderWidth\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", borderColor: 'black', borderWidth: 2 }] }],});</pre>	<pre>Property:annotations.style.borderWidthvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100,</pre>

		<pre>offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { color: 'black', fill: 'white', opacity: 0.7, strokeColor: 'black', strokeWidth: 2 } }] }); diagram.appendTo('#diagram');</pre>
This property allows you to customize labels appearance using user-defined CSS	<pre>Property:labels.cssClass\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", cssClass:"hoverText" }] }],});</pre>	Not applicable
Enables or disables the default behaviors of the label	<pre>Property:labels.constraints\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", constraints: LabelConstraints.All & ~LabelConstraints.Resizable }] }],});</pre>	<pre>Property:annotations.constraintsvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', constraints: ~AnnotationConstraints.InheritReadOnly }] }] }); diagram.appendTo('#diagram');</pre>
Sets the fill color of the text area	<pre>Property:labels.fillColor\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", fillColor: "green" }] }],});</pre>	<pre>Property:annotations.style.fillvar diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { color: 'black', fill: 'white', } }] }] }); diagram.appendTo('#diagram');</pre>

Sets the font color of the text	Property: labels.fontColor \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", fontColor: "green" }] }],});	Property: annotations.style.color var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { color: 'black', } }] }]});diagram.appendTo('#diagram');
Sets the font family of the text	Property: labels.fontFamily \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", fontColor: "green", fontFamily: "seugoe UI", fontSize: 14 }] }],});	Property: annotations.style.fontFamily var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { color: 'black', bold: true, italic: true, fontSize: '12', fontFamily: 'TimesNewRoman' } }] }]});diagram.appendTo('#diagram');
Sets the height of the label	Property: labels.height \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", height: 100, width: 100 }] }],});	Property: annotations.height var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', height: 100, width: 100 }] }]});diagram.appendTo('#diagram');
Sets the horizontal alignment of the label	Property: labels.horizontalAlignment \$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", verticalAlignment: ej.datavisualization.Diagram.VerticalAlignment.Top, horizontalAlignment: ej.datavisualization.Diagram.HorizontalAlignment.Left }] }],});	Property: annotations.horizontalAlignment var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', horizontalAlignment: 'Left', verticalAlignment: 'Top' }] }]});diagram.appendTo('#diagram');

		'Top' }] }}});diagram.appendTo('#diagram');
To set the margin of the label	Property:labels.margin\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", margin:{ left: 5 }, padding:{ left: 5 } }] }] } }]);	Property:annotations.margin var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', margin:{ left: 5 } }] }]}]);diagram.appendTo('#diagram');
Defines whether the label is editable or not	Property:labels.readOnly\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", readOnly:true }] }] } }]);	Property:annotations.constraints var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', constraints: AnnotationConstraints.ReadOnly }] }]}]);diagram.appendTo('#diagram');
Sets the id of svg/html templates . Applicable , if the node's label is HTML or native	Property:labels.templateId\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ templateId:"SvgEllipse" }] }] } }]);	Not applicable
Defines how to align the text inside the label	Property:labels.textAlign\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", textAlign:ej.datavisualization.Diagram.TextAlign.Left, textDecoration: ej.datavisualization.Diagram.TextDecorations.Underline, textOverflow:true, overflowType: ej.datavisualization.Diagram.OverflowType.Ellipsis, wrapping:ej.datavisualization.Diagram.TextWrapping.NoWrap }] }] } }]);	Property:annotations.style.textAlign var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', style: { textAlign: 'Justify', textOverflow: 'Wrap', textDecoration :

		<code>'LineThrough', textWrapping : 'WrapWithOverflow' } }] }}});diagram.appendTo('#diagram');</code>
Enables or disables the visibility of the label	<code>Property:labels.visible\$("#diagram").ejDiagram({ nodes: [{ name: "node", width: 100, height: 100, offsetX: 100, offsetY: 100, labels: [{ text: "Label", visible: false }] }],});</code>	<code>Property:annotations.visible var diagram = new ej.diagrams.Diagram({ nodes: [{ offsetX: 100, offsetY: 100, width: 100, height: 100, annotations: [{ content: 'Annotation', visible : false }] }]);diagram.appendTo('#diagram');</code>
Gets whether the label is currently being edited or not	<code>Property:labels.modevar node = diagram.selectionList[0];console.log(node.labels[0].mode);</code>	Not applicable

PageSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the size and appearance of diagram page	<code>Property:pageSettings.autoScrollBorder\$("#diagramcontent").ejDiagram({ pageSettings: { autoScrollBorder: { left: 50, top: 50, right: 50, bottom: 50 } } });</code>	Not applicable
Sets whether multiple pages can be created to fit all nodes and	<code>Property:pageSettings.multiplePage\$("#diagramcontent").ejDiagram({ pageSettings: { multiplePage:false pageWidth: 800, pageHeight: 500, pageOrientation:ej.datavisualization.Diagram. PageOrientations.Landscape } });</code>	<code>Property:pageSettings.multiplePage var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', pageSettings: { width: 800, height: 600, multiplePage: true, orientation:'Landscape' }, });diagram.appendTo('#diagram');</code>

connectors		
Defines the background color of diagram pages	Property: <code>pageSettings.pageBackgroundColor</code> <code>\$("#diagramcontent").ejDiagram({ pageSettings: { pageBackgroundColor: "lightgrey", }});</code>	Property: <code>pageSettings.background.color</code> <pre>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', pageSettings: { background: { color: 'red', source: 'Clayton.png', scale: 'Meet', align: 'XMidYMid' }},});diagram.appendTo('#diagram');</pre>
Defines the scrollable area of diagram. Applicable, if the scroll limit is not limited	Property: <code>pageSettings.scrollableArea</code> <code>\$("#diagramcontent").ejDiagram({ pageSettings: { scrollableArea: { x:0, y:0, width:1000, height:1000} }});</code>	Property: <code>scrollSettings.scrollableArea</code> <pre>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { scrollableArea: new Rect(0, 0, 300, 300), }},});diagram.appendTo('#diagram');</pre>
Defines the draggable region of diagram elements	Property: <code>pageSettings.boundaryConstraints</code> <code>\$("#diagramcontent").ejDiagram({ pageSettings: { boundaryConstraints: ej.datavisualization.Diagram.BoundaryConstraints.Diagram }});</code>	Property: <code>pageSettings.boundaryConstraints</code> <pre>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', pageSettings: { width: 800, height: 600, boundaryConstraints: 'Diagram' }},});diagram.appendTo('#diagram');</pre>

SymbolPalette

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the size and preview	Property: <code>nodes.paletteItem</code> <code>\$("#symbolpalette").ejSymbolPalette({ palettes: [{ name: "Basic Shapes", expanded: true, items: [{ name: "Rectangle", height: 40, width: 80,</code>	Property: <code>palettes</code> <pre>export function getFlowShapes(): NodeModel[] { var flowShapes = [{ id: 'Terminator', shape: {</pre>

w size of the node to add that to symbol palette	<pre>paletteItem: { previewWidth: 100, previewHeight: 100 } }]]]]]);</pre>	<pre>type: 'Flow', shape: 'Terminator' }, style: { strokeWidth: 2 } }, { id: 'Process', shape: { type: 'Flow', shape: 'Process' }, style: { strokeWidth: 2 } }, { id: 'Decision', shape: { type: 'Flow', shape: 'Decision' }, style: { strokeWidth: 2 } },]];var palette = new ej.diagrams.SymbolPalette ({ expandMode: 'Multiple', palettes: [{ id: 'flow', expanded: true, symbols: getFlowShapes(), title: 'Flow Shapes' },], width: '100%', height: '100%', symbolHeight: 50, symbolWidth: 50, symbolPreview: { height: 100, width: 100 }, enableSearch: true, symbolMargin: { left: 12, right: 12, top: 12, bottom: 12 }, getSymbolInfo: (symbol: NodeModel): SymbolInfo => { return { fit: true }; }}]);palette.appendTo('#sy mbolpalette');</pre>
Defines whether the symbol should be drawn at its actual size regardless of precedence factors or not	<pre>Property:nodes.paletteItem.enableScale\$("#symbolpal ette").ejSymbolPalette({ palettes: [{ name: "Basic Shapes", expanded: true, items: [{ name: "Rectangle", height: 40, width: 80, paletteItem: { previewWidth: 100, previewHeight: 100, enableScale:false } }] }]]]);</pre>	<pre>Property:palettes.fitexport function getFlowShapes(): NodeModel[] { var flowShapes = [{ id: 'Terminator', shape: { type: 'Flow', shape: 'Terminator' }, style: { strokeWidth: 2 } }, { id: 'Process', shape: { type: 'Flow', shape: 'Process' }, style: { strokeWidth: 2 } }, { id: 'Decision', shape: { type: 'Flow', shape: 'Decision' }, style: { strokeWidth: 2 } },]];var palette = new ej.diagrams.SymbolPalette ({ expandMode: 'Multiple', palettes: [{ id: 'flow', expanded: true, symbols: getFlowShapes(), title: 'Flow Shapes' },],</pre>

		<pre>width: '100%', height: '100%', symbolHeight: 50, symbolWidth: 50, symbolPreview: { height: 100, width: 100 }, enableSearch: true, symbolMargin: { left: 12, right: 12, top: 12, bottom: 12 }, getSymbolInfo: (symbol: NodeModel): SymbolInfo => { return { fit: true }; });palette.appendTo('#sy mbolpalette');</pre>
To display a name for nodes in the symbol palette	<pre>Property:nodes.palettetitem.label\$("#symbolpalette") .ejSymbolPalette({ palettes: [{ name: "Basic Shapes", expanded: true, items: [{ name: "Rectangle", height: 40, width: 80, paletteItem: { previewWidth: 100, previewHeight: 100, label: "label", margin: { left: 4, right: 4, top: 4, bottom: 4 } } }]]});</pre>	<pre>Property:palettes.titleexport function getFlowShapes(): NodeModel[] { var flowShapes = [{ id: 'Terminator', shape: { type: 'Flow', shape: 'Terminator' }, style: { strokeWidth: 2 } }, { id: 'Process', shape: { type: 'Flow', shape: 'Process' }, style: { strokeWidth: 2 } }, { id: 'Decision', shape: { type: 'Flow', shape: 'Decision' }, style: { strokeWidth: 2 } },]};var palette = new ej.diagrams.SymbolPalette ({ expandMode: 'Multiple', palettes: [{ id: 'flow', expanded: true, symbols: getFlowShapes(), title: 'Flow Shapes' },], width: '100%', height: '100%', symbolHeight: 50, symbolWidth: 50, symbolPreview: { height: 100, width: 100 }, enableSearch: true, symbolMargin: { left: 12, right: 12, top: 12, bottom: 12 }, getSymbolInfo: (symbol: NodeModel): SymbolInfo => { return { fit: true }; });palette.appendTo('#sy mbolpalette');</pre>

[ScrollSettings](#)

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines the zoom value, zoom factor, scroll status and view port size of the diagram	Property:scrollSettings.horizontalOffset\$("#diagramcontent").ejDiagram({ scrollSettings: { horizontalOffset: 300, verticalOffset: 300 } });	Property:scrollSettings.horizontalOffsetvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { horizontalOffset: 300, verticalOffset: 300 }, });diagram.appendTo('#diagram');
Allows to read the view port width of the diagram	Property:scrollSettings.viewPortWidth\$("#diagramcontent").ejDiagram({ scrollSettings: { viewPortWidth: 300, viewPortHeight: 300 } });	Property:scrollSettings.viewPortWidthvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { viewPortWidth: 300, viewPortHeight: 300 }, });diagram.appendTo('#diagram');
Allows to extend the scrollable region that is based on the scroll limit	Property:scrollSettings.padding\$("#diagramcontent").ejDiagram({ scrollSettings: { padding: { left: 25, right: 25, top: 25, bottom: 25 } } });	Not applicable
Allows to read the zoom value of diagram	Property:scrollSettings.currentZoom\$("#diagramcontent").ejDiagram({ scrollSettings: { currentZoom: 0.2 } });	Property:scrollSettings.currentZoomvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { currentZoom: 0.2 }, });diagram.appendTo('#diagram');
Allows to read the maximum zoom value of scroller	Not applicable	Property:scrollSettings.maxZoomvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { maxZoom: 0.23 }, });diagram.appendTo('#diagram');
Allows to read the maximum	Not applicable	Property:scrollSettings.autoScrollBordervar diagram = new

zoom value of scroller		<code>ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { autoScrollBorder: { left: 25, right: 25, top: 25, bottom: 25 } }, }); diagram.appendTo('#diagram');</code>
Enables/Disables the autoscroll option	Not applicable	Property:scrollSettings.canAutoScroll <code>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { canAutoScroll: false, }, }); diagram.appendTo('#diagram');</code>
Defines the scrollable area of diagram	Not applicable	Property:scrollSettings.scrollableArea <code>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', scrollSettings: { scrollableArea: new Rect(0, 0, 300, 300), }, }); diagram.appendTo('#diagram');</code>

RulerSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables both the horizontal and vertical ruler	Property:rulerSettings.showRulers <code>\$("#diagramcontent").ejDiagram({ rulerSettings: { showRulers: true, } });</code>	Property:rulerSettings.showRulers <code>var arrange = (args: IArrangeTickOptions) => { if (args.tickInterval % 10 == 0) { args.tickLength = 25; } }var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', rulerSettings: { showRulers: true, }, }); diagram.appendTo('#diagram');</code>
Enables or disables both the	Property:rulerSettings.horizontalRuler <code>\$("#diagramcontent").ejDiagram({ rulerSettings: { showRulers: true, horizontalRuler: { segmentWidth: 50, interval: 10, thickness: 50, length: 1000, markerColor: "pink", tickAlignment: ej.datavisualization.Diagram.TickAlignment.Left, } }, });</code>	Property:rulerSettings.horizontalRuler <code>var arrange = (args: IArrangeTickOptions) => { if (args.tickInterval %</code>

horizontal and vertical ruler	<pre>tOrTop, arrangeTick: function alignTick(args){ } }, }));</pre>	<pre>10 == 0) { args.tickLength = 25; } }var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', rulerSettings: { showRulers: true, horizontalRuler: { segmentWidth: 50, interval: 10, thickness: 20, markerColor: 'red', tickAlignment: 'LeftOrTop', arrangeTick: arrange, orientation: 'Horizontal', }, },});diagram.appendTo('#d iagram');</pre>
Enables or disables both the horizontal and vertical ruler	<pre>Property:rulerSettings.verticalRuler\$("#diagramcontent ").ejDiagram({ rulerSettings: { showRulers: true, verticalRuler: { segmentWidth: 50, interval: 10, thickness: 50, length: 1000, markerColor: "pink", tickAlignment: ej.datavisualization.Diagram.TickAlignment.Lef tOrTop, arrangeTick: function alignTick(args){ } } }));</pre>	<pre>Property:rulerSettings.verticalR ulervar arrange = (args: IArrangeTickOptions) => { if (args.tickInterval % 10 == 0) { args.tickLength = 25; } }var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', rulerSettings: { showRulers: true, verticalRuler: { segmentWidth: 200, interval: 20, arrangeTick: arrange, thickness: 20, markerColor: 'red', tickAlignment: 'LeftOrTop', arrangeTick: arrange, orientation: 'Horizontal' } },});diagram.appendTo('#d iagram');</pre>
Updates the gridlines relative to the ruler ticks	Not applicable	<pre>Property:rulerSettings.dynamic Gridvar arrange = (args: IArrangeTickOptions) => { if (args.tickInterval % 10 == 0) { args.tickLength = 25; } }var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', rulerSettings: { dynamicGrid: true, showRulers: true,</pre>

		},));diagram.appendTo('#diagram');
--	--	------------------------------------

SnapSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables snapping nodes/connectors to objects	Property: <code>snapSettings.enableSnapToObject</code> \$("#diagramcontent").ejDiagram({ snapSettings: { enableSnapToObject: false }});	Not applicable
Defines the appearance of horizontal gridlines	Property: <code>snapSettings.horizontalGridLines</code> var gridline = { lineColor : "blue", lineDashArray: "2,2", linesInterval: [1, 14, 0.5, 14.5], snapInterval : [5] }; \$("#diagramcontent").ejDiagram({ snapSettings: { horizontalGridLines: gridline } });	Property: <code>snapSettings.horizontalGridlines</code> var snapSettings = { horizontalGridLines: { lineColor: 'black', lineDashArray: '1,1', lineIntervals: [0.95, 9.05, 0.2, 9.75], snapIntervals: [10] } }; var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', snapSettings: snapSettings }); diagram.appendTo('#diagram');
Defines the appearance of vertical gridlines	Property: <code>snapSettings.verticalGridLines</code> var gridline = { lineColor : "blue", lineDashArray: "2,2", linesInterval: [1, 14, 0.5, 14.5], snapInterval : [5] }; \$("#diagramcontent").ejDiagram({ snapSettings: { verticalGridLines: gridline } });	Property: <code>snapSettings.verticalGridlines</code> var snapSettings = { verticalGridLines: { lineColor: 'black', lineDashArray: '1,1', lineIntervals: [0.95, 9.05, 0.2, 9.75], snapIntervals: [10] } }; var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', snapSettings: snapSettings }); diagram.appendTo('#diagram');
Defines the angle by which the object needs to be snapped	Property: <code>snapSettings.snapAngle</code> \$("#diagramcontent").ejDiagram({ snapSettings: { snapAngle: 10 } });	Property: <code>snapSettings.snapAngle</code> var snapSettings = { snapAngle: 5 }; var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', snapSettings:

		<code>snapSettings});diagram.appendTo('#diagram');</code>
Sets the minimum distance between the selected object and the nearest object	Property: <code>snapSettings.snapObjectDistance</code> <code>\$("#diagramcontent").ejDiagram({ snapSettings: { snapObjectDistance: 5 }});</code>	Property: <code>snapSettings.snapObjectDistance</code> <code>var snapSettings = { snapObjectDistance: 5,};var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', snapSettings: snapSettings});diagram.appendTo('#diagram');</code>
Defines and sets the snapConstraints	Property: <code>snapSettings.snapConstraints</code> <code>\$("#diagramcontent").ejDiagram({ snapSettings: { snapConstraints : ej.datavisualization.Diagram.SnapConstraints.ShowLines }});</code>	Property: <code>snapSettings.constraint</code> <code>\$var snapSettings = { constraints: SnapConstraints.ShowLines,};var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', snapSettings: snapSettings});diagram.appendTo('#diagram');</code>

ZoomFactor

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Sets the factor by which we can zoom in or zoom out	Property: <code>zoomFactor</code> <code>\$("#diagramcontent").ejDiagram({zoomFactor: 1});</code>	Property: <code>zoomFactor</code> <code>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px',});diagram.appendTo('#diagram');</code> <code>var zoomin = { type: 'ZoomIn', zoomFactor: 0.5};diagram.zoomTo(zoomin);</code>

Tool

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Enables/Disables the interactive behaviors of diagram	Property:tool <pre>\$("#diagramcontent").ejDiagram({tool: ej.datavisualization.Diagram.Tools.ZoomPan});</pre>	Property:tool <pre>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', });diagram.appendTo('#diagram');diagram.tool = DiagramTools.ZoomPan;</pre>
---	---	---

ShowTooltip

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Enables or disables tooltip of diagram	Property:showTooltip <pre>\$("#diagramcontent").ejDiagram({showTooltip: true});</pre>	Property:constraints <pre>var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, });diagram.appendTo('#diagram');</pre>

SelectedItems

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
A read only collection of the selected items	Property:selectedItems.children <pre>var diagram = \$("#diagramcontent").ejDiagram("instance");for(var i =0; i< diagram.model.selectedItems.children; i++){ //Do your actions here}</pre>	Not applicable
Controls the visibility of selector	Property:selectedItems.constraints <pre>\$("#diagramcontent").ejDiagram({ selectedItems: { constraints: ej.datavisualization.Diagram.SelectorConstraints.UserHandles }});</pre>	Property:selectedItems.constraints <pre>var diagram = new ej.diagrams.Diagram({ selectedItems: { constraints: SelectorConstraints.UserHandles }, });diagram.appendTo('#diagram');</pre>

Defines a method that dynamically enables/disables the interaction with multiple selection	<pre>Property: selectedItems.getConstraints\$("#diagramcontent").ejDiagram({selectedItems: { getConstraints: function() { return ej.datavisualization.Diagram.NodeConstraints.Drag ej.datavisualization.Diagram.NodeConstraints.Resize } } });</pre>	Not applicable
Sets the height of the selected items	<pre>Property: selectedItems.height\$("#diagramcontent").ejDiagram({ selectedItems: { height:100, width: 100, offsetX:100, offsetY: 100, rotateAngle: 90, } });</pre>	<pre>Property: selectedItems.height tvar diagram = new ej.diagrams.Diagram({ selectedItems: { height:100, width: 100, offsetX:100, offsetY: 100, rotateAngle: 90 },});diagram.appendTo('#diagram');</pre>
Sets the angle to rotate the selected items	<pre>Property: selectedItems.tooltip\$("#diagramcontent").ejDiagram({ selectedItems: { tooltip : { alignment:{ vertical:"top" } } } });</pre>	Not applicable
A collection of frequently used com	<pre>Property: selectedItems.userHandlesvar userHandle= [];var cloneHandle = ej.datavisualization.Diagram.UserHandle();userHandle.push(cloneHandle);\$("#diagramcontent").ejDiagram({ selectedItems: { userHandles:userHandle } });</pre>	<pre>Property: selectedItems.userHandlesvar handle = [{ name: 'handle1', pathData: 'M 60.3,18 H 27.5 c -3,0-5.5,2.4-5.5,5.5 v 38.2 h 5.5 V 23.5 h 32.7 V 18 z M 68.5,28.9 h -30 c -3,0-5.5,2.4-5.5,5.5 v 38.2</pre>

mands that will be added around the selector		<pre>c 0,3,2.4,5.5,5.5,5.5 h 30 c 3,0,5.5-2.4,5.5- 5.5 V 34.4 C 73.9,31.4,71.5,28.9,68. 5,28.9 z M 68.5,72.5 h -30 V 34.4 h 30 V 72.5 z' , visible: true, backgroundColor: 'black', offset: 0, side: 'Bottom', margin: { top: 0, bottom: 0, left: 0, right: 0 }, pathColor: 'white'}}];var diagram = new ej.diagrams.Diagram({ selectedItems: { constraints: SelectorConstraints.Use rHandles, userHandles: handle },});diagram.appendTo('#diagram');</pre>
Sets the horizontal alignment of the user handle	<pre>Property:selectedItems.userHandles.horizontalAlignmentvar userHandle = [];var cloneHandle = ej.datavisualization.Diagram.UserHandle();cloneHa ndle = {name : "cloneHandle",pathData : "M 4.6350084,4.8909971 L 4.6350084,9.3649971 9.5480137,9.3649971 9.5480137,4.8909971 z M 3.0000062,2.8189973 L 11.184016,2.8189973 11.184016,10.999997 3.0000062,10.999997 z M 0,0 L 7.3649998,0 7.3649998,1.4020001 1.4029988,1.4020001 1.4029988,8.0660002 0,8.0660002 0,1.4020001 0,0.70300276 z",visible : "true",backgroundColor : "#4D4D4D",offset : ej.datavisualization.Diagram.point(0, 0),position :" middleleft"margin : { left: 5 },pathColor : "white",horizontalAlignment : ej.datavisualization.Diagram.HorizontalAlignment. Right,verticalAlignment : ej.datavisualization.Diagram.VerticalAlignment.To p,borderColor : "red",borderWidth : 3,size : 20};userHandle.push(cloneHandle);\$("#diagramconte nt").ejDiagram({ selectedItems: { userHandles:userHandle }}});</pre>	<pre>Property:selectedItems.user Handles.horizontalAlignmen tvar handle = [{ name: 'handle1', pathData: 'M 60.3,18 H 27.5 c -3,0- 5.5,2.4-5.5,5.5 v 38.2 h 5.5 V 23.5 h 32.7 V 18 z M 68.5,28.9 h -30 c -3,0-5.5,2.4-5.5,5.5 v 38.2 c 0,3,2.4,5.5,5.5,5.5 h 30 c 3,0,5.5-2.4,5.5- 5.5 V 34.4 C 73.9,31.4,71.5,28.9,68. 5,28.9 z M 68.5,72.5 h -30 V 34.4 h 30 V 72.5 z' , visible: true, backgroundColor: 'black', offset: 0, side: 'Bottom', margin: { top: 0, bottom: 0, left: 0, right: 0 }, pathColor: 'white', horizontalAlignment: 'Center', verticalAlignment: 'Center', borderColor: 'red', borderWidth: 3, size: 30}];var diagram = new ej.diagrams.Diagram({ selectedItems: {</pre>

		<pre>constraints: SelectorConstraints.Use rHandles, userHandles: handle },});diagram.appendTo('#diagram');</pre>
<p>Defines the interactive behavior of the user handle</p>	<p>Property: <code>selectedItems.userHandles.tool</code></p> <pre>var CloneTool = (function(base) { ej.datavisualization.Diagram.extend(CloneTool, base); function CloneTool(name) { base.call(this, name); this.singleAction = true; this.clonedNodes = []; this.cursor = "pointer"; } CloneTool.prototype.mouseup = function(evt) { this.diagram.copy(); this.diagram.paste(); }}return CloneTool;});(ej.datavisualization.Diagram.ToolBa se);var userHandle = [];var cloneHandle = ej.datavisualization.Diagram.UserHandle();cloneHa ndle.name = "cloneHandle";cloneHandle.pathData = "M 4.6350084,4.8909971 L 4.6350084,9.3649971 9.5480137,9.3649971 9.5480137,4.8909971 z M 3.0000062,2.8189973 L 11.184016,2.8189973 11.184016,10.999997 3.0000062,10.999997 z M 0,0 L 7.3649998,0 7.3649998,1.4020001 1.4029988,1.4020001 1.4029988,8.0660002 0,8.0660002 0,1.4020001 0,0.70300276 z";cloneHandle.tool = new CloneTool(cloneHandle.name);;userHandle.push(clon eHandle);\$("#diagramcontent").ejDiagram({ selectedItems: { userHandles: userHandle }});</pre>	Not applicable
<p>Defines whether the user handle should be added, when more than one element is selected</p>	<p>Property: <code>selectedItems.userHandles.enableMultiSelection</code></p> <pre>var userHandle = [];var cloneHandle = ej.datavisualization.Diagram.UserHandle();cloneHa ndle.name = "cloneHandle";cloneHandle.enableMultiSelection = true;userHandle.push(cloneHandle);\$("#diagramcont ent").ejDiagram({ selectedItems: { userHandles: userHandle }});</pre>	Not applicable

Sets the horizontal alignment of the user handle	Not applicable	<pre> Property:selectedItems.user Handles.displacementvar handle = [{ name: 'handle1', pathData: 'M 60.3,18 H 27.5 c -3,0- 5.5,2.4-5.5,5.5 v 38.2 h 5.5 V 23.5 h 32.7 V 18 z M 68.5,28.9 h -30 c -3,0-5.5,2.4-5.5,5.5 v 38.2 c 0,3,2.4,5.5,5.5,5.5 h 30 c 3,0,5.5-2.4,5.5- 5.5 V 34.4 C 73.9,31.4,71.5,28.9,68. 5,28.9 z M 68.5,72.5 h -30 V 34.4 h 30 V 72.5 z', displacement: 30}];var diagram = new ej.diagrams.Diagram({ selectedItems: { constraints: SelectorConstraints.Use rHandles, userHandles: handle },});diagram.appendTo('#diagram');</pre>
--	----------------	---

SerializationSettings

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
Defines whether the default diagram properties can be serialized or not	<pre> Property:serializationSettings.preventDefaultValues\$("#diagramcontent").ejDiagram({ serializationSettings: { preventDefaultValues: true }});</pre>	Not applicable

How to load EJ1 diagram in EJ2 diagram

To load EJ1 JSON data in an EJ2 diagram, follow these steps.

1. Import and inject the EJ1SerializationModule as shown in the following code example.

```
`javascript
```



```
import { Diagram } from '@syncfusion/ej2-diagrams';
import { EJ1SerializationModule } from '@syncfusion/ej2-diagrams';
Diagram.Inject(EJ1SerializationModule);
`
```

2. Load the EJ1 JSON data using the diagram loadDiagram method and set the second parameter to true.

```
`javascript
var ej1Data = {"JSONData"}; // Replace JSONData with your EJ1 JSON data
//Load the EJ1 JSON and pass boolean value as true
diagram.loadDiagram(ej1Data, true);
`
```

Tooltip

<!-- markdownlint-disable MD033 -->

behavior	API in Essential JS 1	API in Essential JS 2
An object that defines the description, appearance and alignments of tooltip	Property:tooltip \$("#diagramcontent").ejDiagram({ tooltip: { templateId: "mouseovertooltip", relativeMode: ej.datavisualization.Diagram.RelativeMode.Mouse, }, nodes: [{ name: "elizabeth", width: 70, height: 40, offsetX: 100, offsetY: 100, Designation: "Managing Director" }]});	Property:tooltip var diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { content: 'Diagram', position: 'TopLeft', relativeMode: 'Object', animation: { open: { effect: 'FadeZoomIn', delay: 0 }, close: { effect: 'FadeZoomOut', delay: 0 } } },});diagram.appendTo('#diagram');
Defines the alignment of tooltip	Property:tooltip.alignment \$("#diagramcontent").ejDiagram({ tooltip: { templateId: "mouseovertooltip", alignment: { horizontal: "center", vertical: "bottom" }, relativeMode: ej.datavisualization.Diagram.RelativeMode.Mouse, }, nodes: [{ name: "elizabeth", width: 70, height: 40, offsetX: 100, offsetY: 100, Designation: "Managing Director" }]});	Not applicable
Sets the margin of the tooltip	Property:tooltip.margin \$("#diagramcontent").ejDiagram({ tooltip: { templateId: "mouseovertooltip", alignment: { horizontal: "center", vertical: "bottom" }, relativeMode:	Not applicable

	ej.datavisualization.Diagram.RelativeMode.Mouse, margin : { left: 5, right: 5, top: 5, bottom: 5 } }, nodes: [{ name: "elizabeth", width: 70, height: 40, offsetX: 100, offsetY: 100, Designation: "Managing Director" }]]);	
Sets the svg/html template to be bound with tooltip	Property:tooltip.templateId\$("#diagramcontent").ejDiagram({ tooltip: { templateId: "mouseovertooltip", alignment: { horizontal: "center", vertical: "bottom" }, relativeMode: ej.datavisualization.Diagram.RelativeMode.Mouse, margin : { left: 5, right: 5, top: 5, bottom: 5 } }, nodes: [{ name: "elizabeth", width: 70, height: 40, offsetX: 100, offsetY: 100, Designation: "Managing Director" }]]);	Property:tooltip.contentvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { content: 'Diagram', },});diagram.appendTo('#diagram');
Defines if the Tooltip has tip pointer or not	Not applicable	Property:tooltip.showTipPointervar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { content: 'Diagram', showTipPointer: true, },});diagram.appendTo('#diagram');
Defines the position of the Tooltip	Not applicable	Property:tooltip.positionvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { content: 'Diagram', position: 'TopLeft', relativeMode: 'Object', },});diagram.appendTo('#diagram');
Allows to set the same or different animation option	Not applicable	Property:tooltip.animationvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default

for the Tooltip, when it is opened or closed		<pre>DiagramConstraints.Tooltip, tooltip: { content: 'Diagram', position: 'TopLeft', relativeMode: 'Object', animation: { open: { effect: 'FadeZoomIn', delay: 0 }, close: { effect: 'FadeZoomOut', delay: 0 } } },});diagram.appendTo('#diagram');</pre>
Sets the width of the tooltip	Not applicable	<pre>Property:tooltip.widthvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { width: 100, content: 'Diagram', position: 'TopLeft', relativeMode: 'Object' },});diagram.appendTo('#diagram');</pre>
Sets the height of the Tooltip	Not applicable	<pre>Property:tooltip.heightvar diagram = new ej.diagrams.Diagram({ width: '100%', height: '600px', constraints: DiagramConstraints.Default DiagramConstraints.Tooltip, tooltip: { height: 100, content: 'Diagram', position: 'TopLeft', relativeMode: 'Object', },});diagram.appendTo('#diagram');</pre>

How to load EJ1 diagram in EJ2 diagram

To load EJ1 JSON data in an EJ2 diagram, follow these steps.

1. Import and inject the EJ1SerializationModule as shown in the following code example.

```
`javascript
```

```
import { Diagram } from '@syncfusion/ej2-diagrams';
```

```
import { EJ1SerializationModule } from '@syncfusion/ej2-diagrams';
```

```
Diagram.Inject(EJ1SerializationModule);
```

```
,
```

2. Load the EJ1 JSON data using the diagram loadDiagram method and set the second parameter to true.

```
`javascript
var ej1Data = {"JSONData"}; // Replace JSONData with your EJ1 JSON data
//Load the EJ1 JSON and pass boolean value as true
var diagram = document.getElementById('diagram').ej2_instances[0];
diagram.loadDiagram(ej1Data, true);
`
```

Dialog

Template in EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { Button } from '@syncfusion/ej2-buttons';
/**
 * Dialog template sample
 */
let icontemp: string = '<button id="sendButton" class="e-control e-btn e-primary" data-ripple="true">' + 'Send</button>';
let headerimg: string = '';
let sendbutton: Button = new Button();
let proxy: any = this;
// Initialize Dialog component
let dialog = new 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 = (): void => {
    // Call the show method to open the Dialog
    dialog.show();
}
(document.getElementById('sendButton') as HTMLElement).onkeydown = (e: any)
=> {
    if (e.keyCode === 13) { updateTextValue(); }
};
(document.getElementById('inVal') as HTMLElement).onkeydown = (e: any) => {
    if (e.keyCode === 13) { updateTextValue(); }
};
document.getElementById('sendButton').onclick = (): void => {
    updateTextValue();
};
function onBeforeopen(): void {
    document.getElementById('dlgContent').style.visibility = 'visible';
}
function updateTextValue () : void {
    let enteredVal: HTMLInputElement = document.getElementById('inVal') as
HTMLInputElement;
    let dialogTextElement: HTMLElement =
document.getElementsByClassName('dialogText')[0] as HTMLElement;
    let dialogTextWrap : HTMLElement =
document.getElementsByClassName('dialogContent')[0] as HTMLElement;
    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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Dialog component
let dialog = new 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',
        cssClass: 'e-flat'
      }
    }
  ],
  // 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 = (): void => {
    // 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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let dialog = new 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',
                cssClass: 'e-flat'
            }
        }
    ],
    // 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 = (): void => {
    // 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 EJ2 JavaScript 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.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
document.getElementById('targetButton').onclick = (): void => {
    // Initialize and render alert dialog
    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.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
document.getElementById('targetButton').onclick = (): void => {
  // Initialize and render alert dialog with options
  DialogUtility.alert({
    title: 'Alert Dialog',
    content: "This is an Alert Dialog!",
    okButton: { text: 'OK', click: okClick.bind(this) },
    showCloseIcon: true,
    closeOnEscape: true,
    animationSettings: { effect: 'Zoom' }
  });
};
function okClick(): void {
  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.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
document.getElementById('targetButton').onclick = (): void => {
  // Initialize and render Confirm dialog
  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.TS

```

import { DialogUtility } from '@syncfusion/ej2-popups';
// Initialize and render Confirm dialog with options
document.getElementById('targetButton').onclick = (): void => {
    DialogUtility.confirm({
        title: ' Confirmation Dialog',
        content: "This is a Confirmation Dialog!",
        okButton: { text: 'OK', click: okClick.bind(this) },
        cancelButton: { text: 'Cancel', click: cancelClick.bind(this)},
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
    });
};
function okClick(): void {

```

```

    alert('you clicked OK button');
}
function cancelClick(): void {
    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.TS

```
import { DialogUtility } from '@syncfusion/ej2-popups';
var DialogObj;
// Initialize and render Confirm dialog with options
document.getElementById('targetButton').onclick = (): void => {
    DialogObj = DialogUtility.confirm({
        title: ' Confirmation Dialog',
        content: "This is a Confirmation Dialog!",
        okButton: { text: 'OK', click: okClick.bind(this) },
        cancelButton: { text: 'Cancel', click: cancelClick.bind(this) },
        showCloseIcon: true,
        closeOnEscape: true,
        animationSettings: { effect: 'Zoom' }
    });
};
function okClick(): void {
    alert('you clicked OK button');
}
function cancelClick(): void {
    //Hide the dialog
    DialogObj.hide();
}
```

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>
<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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Load French culture for Dialog close button tooltip text
L10n.load({
  'fr-BE': {
    'dialog': {
      'close': "Fermer"
    }
  }
});
// Initialization of Dialog component
let dialog = new 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',
});
// Sample level code to handle the button click action
document.getElementById('targetButton').onclick = (): void => {
  // 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 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 EJ2 JavaScript 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 ARIA 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 its 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Dialog component
let dialog: Dialog = new 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',
      cssClass: 'e-flat',
    },
  ],
  // 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 = (): void => {
    // Call the show method to open the Dialog
    dialog.show();
};
function onBeforeopen(): void {
    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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
```



```
enableRipple(true);
// Initialize the Outer Dialog component
let dialog = new 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 = (): void => {
  // Call the show method to open the Dialog
  dialog.show();
}
// initialize the Inner Dialog component
let innerDialog = new 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 = (): void => {
  // Call the show method to open the Dialog
  innerDialog.show();
}
// Render initialized inner Dialog
innerDialog.appendTo('#innerDialog');
function onBeforeopen(): void {
  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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Dialog component

```

```

let dialog = new 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 = (): void => {
  dialog.cssClass = 'e-fixed';
}
function onBeforeopen(): void {
  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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialization of Dialog component
let dialog = new Dialog({
    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',
                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 = (): void => {
    // 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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Dialog component
let dialog = new Dialog({
    // 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,
            },
        },
    ],
});

```

```

        cssClass: 'e-flat',
        content: 'OK'
    },
    {
        'click': () => {
            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 = (): void => {
    // 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>
<script>
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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
let firstDialog = new 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');
let secondDialog = new 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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
// To get the all input fields and its container.
let 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 (let i = 0; i < inputElement.length; i++) {
    inputElement[i].addEventListener("focus", function () {
        let 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 () {
        let 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 (let i = 0; i < inputIcon.length; i++) {
    inputIcon[i].addEventListener('mousedown', function () {
        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
enableRipple(true);
// Initialize Dialog component
let dialog: Dialog = new 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 = (): void => {
    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(): void {
    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>
</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 EJ2 JavaScript 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.TS

```
import { enableRipple } from '@syncfusion/ej2-base';
import { Button } from '@syncfusion/ej2-buttons';
import { Dialog, OpenEventArgs } from '@syncfusion/ej2-popups';
let dialogObj: Dialog = new 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
});
dialogObj.appendTo('#dialog');
document.getElementById('targetButton').onclick = (): void => {
    dialogObj.show();
};
function btnClick() {
    dialogObj.hide();
}
function onOpen(args: OpenEventArgs) {
    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>
<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 EJ2 JavaScript 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.TS

```

import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
// To get the all input fields and its container.
let 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 (let i = 0; i < inputElement.length; i++) {
  inputElement[i].addEventListener("focus", function () {
    let 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 () {
    let 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 (let i = 0; i < inputIcon.length; i++) {
  inputIcon[i].addEventListener('mousedown', function () {

```

```

        this.classList.add('e-input-btn-ripple');
    });
    inputIcon[i].addEventListener('mouseup', function () {
        let element = this;
        setTimeout(function () {
            element.classList.remove('e-input-btn-ripple');
        }, 500);
    });
}
enableRipple(true);
// Initialize Dialog component
let dialog = new Dialog({
    header: 'Success',
    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,
                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 = (): void => {
    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 EJ2 JavaScript 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.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: '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
});
dialogObj.appendTo('#dialog');
document.getElementById('openBtn').onclick = (): void => {
    dialogObj.show();
};
let modalObj: Dialog;
function createModalDialog() {
    dialogObj.hide()
    if (!document.getElementById('modalDialog').classList.contains('e-dialog')) {
        modalObj = new 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();
            dialogObj.show();
        }},
        target: document.querySelector('body'),
        animationSettings: { effect: 'Zoom' }
    });
    modalObj.appendTo('#modalDialog');
} else {
    modalObj.content = getDynamicContent();
    modalObj.show()
}
}
function getDynamicContent(): string {
    let input: HTMLInputElement =
document.getElementById('dialog').querySelector('#name');
    let email: HTMLInputElement =
document.getElementById('dialog').querySelector('#email');
    let contact: HTMLInputElement =
document.getElementById('dialog').querySelector('#contact');
    let address: HTMLTextAreaElement =
document.getElementById('dialog').querySelector('#address');
    let template: string = "<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(): void {
    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>
</body>
<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 EJ2 JavaScript 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.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: '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
});
dialogObj.appendTo('#dialog');
document.getElementById('openBtn').onclick = (): void => {
    dialogObj.show();
};
function onBeforeopen(): void {
    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>
</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 EJ2 JavaScript 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>
<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 EJ2 JavaScript 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.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', 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 = (): void => {
    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.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,
    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 = (): void => {
    dialogObj.show();
};
document.getElementById('Button1').onclick = (): void => {
    dialogObj.hide();
};
document.getElementById('Button2').onclick = (): void => {
    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 EJ2 JavaScript 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.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: `<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 = (): void => {
    dialogObj.show();
};
function btnClick() {
    dialogObj.hide();
}
let hide: any;
let isFullScreen: Boolean;
let dialogOldPositions: any;
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 EJ2 JavaScript 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.TS

```
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize Dialog component.
let dialog = new 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 = (): void => {
    // Call the show method to open the Dialog.
    dialog.show();
}
function onCreate() {
    document.getElementById('dlgContent').style.display = 'block';
    dialog.refreshPosition();
}
function onOpen(args: beforeOpenEventArgs) {
    // 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.
                    <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 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>

```

Ej1 api migration in EJ2 JavaScript Dialog control

This section API migration process of Dialog component from Essential JS1 to Essential JS2.

Accessibility and Localization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Keyboard Navigation	Property: allowKeyboardNavigation <pre>\$('#dialog').ejDialog({ allowKeyboardNavigation: true })</pre>	No separate API: for enable/disable keyboard navigation.Its enabled by default.
Localization	Property: locale <pre>\$('#dialog').ejDialog({ locale: 'es-ES' })</pre>	Property: locale <pre>var dialog = ej.popups.Dialog({ locale: 'es-ES' })</pre>
Right to left	Property: enableRTL <pre>\$('#dialog').ejDialog({ enableRTL: true })</pre>	Property: enableRTL <pre>var dialog = ej.popups.Dialog({ enableRtl: true })</pre>

Header

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Header Content	Property: title <pre>\$('#dialog').ejDialog({ title: 'EJ1 Dialog header' })</pre> Method: setTitle <pre>\$('#dialog').ejDialog('setTitle', 'EJ1 Dialog Header')</pre>	Property: header <pre>var dialog = ej.popups.Dialog({ header: 'EJ2 Dialog' })</pre>
close button	Property: actionButtons <pre>\$('#dialog').ejDialog({</pre>	Property: showCloseIcon <pre>var dialog = ej.popups.Dialog({</pre>

	<pre> actionButtons: ["close"] }) </pre>	<pre> showCloseIcon: true }) </pre>
Event triggers when click on action buttons	<pre> Event: actionButtonClick \$('#dialog').ejDialog({ actionButtonClick: function () {} }) </pre>	Not Applicable
Minimize	<pre> Property: actionButtons \$('#dialog').ejDialog({ actionButtons: ["minimize"] }) </pre>	Not Applicable
Maximize	<pre> Property: actionButtons \$('#dialog').ejDialog({ actionButtons: ["maximize"] }) </pre>	Not Applicable
Collapse /Expand	<pre> Property: actionButtons Method: collapse(), expand () \$('#dialog').ejDialog({ actionButtons: ["collapsible"] }) \$('#dialog').ejDialog('collapse') \$('#dialog').ejDialog('expand') </pre>	Not Applicable
Event triggers when expanding the collapsed dialog	<pre> Event: expand \$('#dialog').ejDialog({ expand: function () {} }) </pre>	Not Applicable
Event triggers when collapsing the expanded dialog	<pre> Event: collapse \$('#dialog').ejDialog({ collapse: function () {} }) </pre>	Not Applicable
Pin	<pre> Property: actionButtons \$('#dialog').ejDialog({ actionButtons: ["pin"] }) </pre>	Not Applicable
Header visibility	<pre> Property: showHeader \$('#dialog').ejDialog({ </pre>	Not Applicable

	showHeader: true })	
Close on escape key press	Property: closeOnEscape \$('#dialog').ejDialog({ closeOnEscape: true })	Property: closeOnEscape var dialog = ej.popups.Dialog({ closeOnEscape: true })

Footer

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Footer Content	Property: footerTemplateId \$('#dialog').ejDialog({ footerTemplateId: 'sample' })	Property: footerTemplate var dialog = ej.popups.Dialog({ footerTemplate: ' <button>submit< button>'<br=""></button>submit<> })
Footer action buttons	Not applicable	Property: buttons var dialog = ej.popups.Dialog({ buttons: [{ click: dialogBtnClick, buttonModel: { content: 'OK', isPrimary: true } }] })
Footer visibility	Property: showFooter \$('#dialog').ejDialog({ showFooter: true })	Not Applicable

Content

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Dialog content	Method: setContent	Property: content

	<code>\$('#dialog').ejDialog('setContent', 'Dialog Content')</code>	<code>var dialog = ej.popups.Dialog({ content: 'Dialog content' })</code>
Loading content using AJAX request	<code>Property: contentType, contentUrl</code> <code>\$('#dialog').ejDialog({ contentType: "ajax", contentUrl: ' ' })</code>	Not Applicable
Event triggers after the dialog content loaded in DOM	Event: contentLoad <code>\$('#dialog').ejDialog({ contentLoad: function () {} })</code>	Not Applicable
Event trigger when fails to load ajax content	Event: ajaxError <code>\$('#dialog').ejDialog({ ajaxError: function () {} })</code>	Not Applicable
Event trigger when load ajax content successfully	Event: ajaxSuccess <code>\$('#dialog').ejDialog({ ajaxSuccess: function () {} })</code>	Not Applicable

Animation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enabling Animation	Property: enableAnimation <code>\$('#dialog').ejDialog({ enableAnimation: true })</code>	Not Applicable
Animation effects	Property: animation.show.effect <code>\$('#dialog').ejDialog({ Animation: { show: { effect: 'slide' } } })</code>	Property: animationSettings.effect <code>var dialog = ej.popups.Dialog({ animationSettings: { effect: 'Zoom' } })</code>

	<pre> } }) </pre>	
Animation duration	Property: animation.show.duration <pre> \$('#dialog').ejDialog({ Animation: { show: { effect: 'slide', duration: 500 } } }) </pre>	Property: animationSettings.duration <pre> var dialog = ej.popups.Dialog({ animationSettings: { effect: 'Zoom', duration: 500 } }) </pre>
Animation delay	Not applicable	Property: animationSettings.delay <pre> var dialog = ej.popups.Dialog({ animationSettings: { effect: 'Zoom', duration: 500, delay: 300 } }) </pre>

Draggable and resizing

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Draggable dialog	Property: allowDraggable <pre> \$('#dialog').ejDialog({ allowDraggable: true }) </pre>	Property: allowDragging <pre> var dialog = ej.popups.Dialog({ allowDragging: true }) </pre>
Event triggers when the user drags the dialog	Event: drag <pre> \$('#dialog').ejDialog({ drag: function () {} }) </pre>	Event: drag <pre> var dialog = ej.popups.Dialog({ drag: function() {} }) dialog.appendTo('#ej2Dialog') </pre>
Event triggers when the start to drag the dialog	Event: dragStart <pre> \$('#dialog').ejDialog({ </pre>	Event: dragStart <pre> var dialog = ej.popups.Dialog({ dragStart: function() {} </pre>

	dragStart: function () {} })	dialog.appendTo('#ej2Dialog')
Event triggers when the stops to drag the dialog	Event: dragStop \$('#dialog').ejDialog({ dragStop: function () {} })	Event: dragStop var dialog = ej.popups.Dialog({ dragStop: function() {} })dialog.appendTo('#ej2Dialog')
Resizing dialog	Property: enableResize \$('#dialog').ejDialog({ enableResize: true })	Not applicable
Event triggers when resizing the dialog	Event: resize \$('#dialog').ejDialog({ resize: function () {} })	Not Applicable
Event triggers when starts to resizing the dialog	Event: resizeStart \$('#dialog').ejDialog({ resizeStart: function () { } })	Not Applicable
Event triggers when the stops to resizing the dialog	Event: resizeStop \$('#dialog').ejDialog({ resizeStop: function () {} })	Not Applicable

Target

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Target element to append dialog in document	Property: target \$('#dialog').ejDialog({ target: '#dialogTarget' })	Property: target var dialog = ej.popups.Dialog({ target: '#dialogTarget' })
Element for draggable area	Property: containment \$('#dialog').ejDialog({	Not applicable

	containment: '#dragArea' }))	
--	---------------------------------	--

Position

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Customizing dialog position using X, Y coordinate values	Property: position \$('#dialog').ejDialog({ position: { X: 300, Y: 100 } })	Property: position var dialog = ej.popups.Dialog({ position: { X: 300, Y: 100} })
positioning dialog using position values	Not Applicable	Property: position var dialog = ej.popups.Dialog({position: { X: 'Center', Y: 'Center'} })

Visibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Render dialog in visible/hidden state	Property: showOnInit \$('#dialog').ejDialog({ showOnInit: true })	Property: visible var dialog = ej.popups.Dialog({ visible: true })

Dialog Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Render modal dialog	Property: enableModal \$('#dialog').ejDialog({ enableModal: true })	Property: isModal var dialog = ej.popups.Dialog({ isModal: true })

Tooltip

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Sets the tooltip for dialog buttons	Property: tooltip <code>\$('#dialog').ejDialog({ tooltip: { close: 'Exit' } })</code>	No Separate API for tooltip. It renders based on locale text.

Control State

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the control	Property: enabled <code>\$('#dialog').ejDialog({ enabled: false })</code>	Not Applicable
Enable/ Disable page scrolling	Property: backgroundScroll <code>\$('#dialog').ejDialog({ backgroundScroll: false })</code>	No separate API for disabling page scroll.By default, scrolling prevented for modal dialog

State Maintenance

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Save the model values in local storage or cookies	Property: enablePersistence <code>\$('dialog').ejDialog({ enablePersistence: true })</code>	Property: enablePersistence <code>var dialog = ej.popups.Dialog({ enablePersistence: true })</code>

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
----------	-----------------------	-----------------------

Adjusting Height	Property: height \$('#dialog').ejDialog({ height: 400 })	Property: height var dialog = ej.popups.Dialog({ height: '50%' })
Adjusting width	Property: width \$('#dialog').ejDialog({ width: 400 })	Property: width var dialog = ej.popups.Dialog({ width: '50%' })
Adding custom class	Property: cssClass \$('#dialog').ejDialog({ cssClass: 'custom-class' })	Property: cssClass var dialog = ej.popups.Dialog({ cssClass: 'custom-class' })
Adding zIndex	Property: zIndex \$('#dialog').ejDialog({ zIndex: 2000 })	Property: zIndex var dialog = ej.popups.Dialog({ zIndex: 2000 })
Maximum height	Property: maxHeight \$('#dialog').ejDialog({ maxHeight: 600 })	Not Applicable
Maximum width	Property: maxWidth \$('#dialog').ejDialog({ maxWidth: 600 })	Not Applicable
Minimum height	Property: minHeight \$('#dialog').ejDialog({ minHeight: 300 })	Not Applicable
Minimum width	Property: minWidth \$('#dialog').ejDialog({ minWidth: 300 })	Not Applicable
Adding html attributes	Property: htmlAttributes \$('#dialog').ejDialog({	Not Applicable

	<pre>htmlAttributes: { class: 'my-class' } })</pre>	
Custom icon in the header	<pre>Property: faviconCSS \$('#dialog').ejDialog({ faviconCSS: 'custom-icon' })</pre>	Not Applicable
Rounded corner appearance	<pre>Property: showRoundedCorner \$('#dialog').ejDialog({ showRoundedCorner: true })</pre>	Not Applicable
Make control flexible for mobile view	<pre>Property: isResponsive \$('#dialog').ejDialog({ isResponsive: true })</pre>	Not Applicable
Close the Dialog	<pre>Method: close() \$('#dialog').ejDialog('close')</pre>	<pre>Method: hide() var dialog = ej.popups.Dialog() dialog.appendTo("#ej2Dialog") dialog.hide()</pre>
Event triggers before the dialog closes	<pre>Event: beforeClose() \$('#dialog').ejDialog({ beforeClose: function () {} })</pre>	<pre>Event: beforeClose() var dialog = ej.popups.Dialog({ beforeClose: beforeCloseDialog }) dialog.appendTo('#ej2Dialog') function beforeCloseDialog() {}</pre>
Event triggers when the dialog closes	<pre>Event: close() \$('#dialog').ejDialog({ close: function () {} })</pre>	<pre>Event: close() var dialog = ej.popups.Dialog({ close: CloseDialog }) dialog.appendTo('#ej2Dialog') function CloseDialog() {}</pre>
Destroy the control	<pre>Method: destroy() \$('#dialog').ejDialog('destroy')</pre>	<pre>Method: destroy() var dialog = ej.popups.Dialog()</pre>

		<pre>dialog.appendTo("#ej2Dialog") dialog.destroy()</pre>
Focus the dialog element	Method: focus() <pre>\$('#dialog').ejDialog('focus')</pre>	Not Applicable
Check whether the dialog is open	Method: isOpen() <pre>\$('#dialog').ejDialog('isOpen')</pre>	Not Applicable
Maximize the dialog	Method: maximize() <pre>\$('#dialog').ejDialog('maximize')</pre>	Not Applicable
Minimize the dialog	Method: minimize() <pre>\$('#dialog').ejDialog('minimize')</pre>	Not Applicable
Open the dialog	Method: open() <pre>\$('#dialog').ejDialog('open')</pre>	<pre>Method: show() var dialog = ej.popups.Dialog() dialog.appendTo("#ej2Dialog") dialog.show()</pre>
Event trigger before the dialog opens	Event: beforeOpen() <pre>\$('#dialog').ejDialog({ beforeOpen: function () {} })</pre>	<pre>Event: beforeOpen() var dialog = ej.popups.Dialog({ beforeOpen: beforeOpenDialog }) dialog.appendTo("#ej2Dialog") function beforeOpenDialog() {}</pre>
Event triggers when the opens the dialog	Event: open() <pre>\$('#dialog').ejDialog({ open: function () {} })</pre>	<pre>Event: open() var dialog = ej.popups.Dialog({ open: function() {} }) dialog.appendTo("#ej2Dialog")</pre>
Refresh the dialog	Method: refresh() <pre>\$('#dialog').ejDialog('refresh')</pre>	<pre>Method: refreshPosition() var dialog = ej.popups.Dialog() dialog.appendTo("#ej2Dialog") dialog.refreshPosition()</pre>
Pin/ unpin the dialog	Method: pin <pre>\$('#dialog').ejDialog('pin') \$('#dialog').ejDialog('unpin')</pre>	Not Applicable

Event triggers after the dialog created successfully	Event: create() <pre>\$('#dialog').ejDialog({ create: function () {} })</pre>	Event: created() <pre>var dialog = ej.popups.Dialog created: function() {} }) dialog.appendTo('#ej2Dialog')</pre>
Event triggers when the control destroyed successfully	Event: destroy <pre>\$('#dialog').ejDialog({ destroy: function () {} })</pre>	Not Applicable
Event triggers on clicking on modal dialog overlay	Not Applicable	Event: overlayClick() <pre>var dialog = ej.popups.Dialog({ overlayClick: function() {} })</pre>

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\(ES5\)](#)
- [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.

Getting started in EJ2 JavaScript Document editor control

The Essential JS 2 for JavaScript (global script) is an ES5 formatted pure JavaScript framework which can be directly used in latest web browsers.

Component Initialization

The Essential JS 2 JavaScript components can be initialized by using either of the following ways.

- Using local script and style references in a HTML page.
- Using CDN link for script and style reference.

Using local script and style references in a HTML page

Step 1: Create an app folder `app` for Essential JS 2 JavaScript components.

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

Syntax:

Script: **(installed location)/JavaScript - EJ2/{RELEASEVERSION}/Web (Essential JS 2)/JavaScript/{PACKAGENAME}/dist/global/{PACKAGE_NAME}.min.js**

Styles: **(installed location)/JavaScript - EJ2/{RELEASEVERSION}/Web (Essential JS 2)/JavaScript/{PACKAGENAME}/styles/material.css**

Example:

Script: `C:/Program Files (x86)/Syncfusion/Essential Studio/JavaScript - EJ2/23.1.36/Web (Essential JS 2)/JavaScript/ej2-documenteditor/dist/global/ej2-documenteditor.min.js`

Styles: `C:/Program Files (x86)/Syncfusion/Essential Studio/JavaScript - EJ2/23.1.36/Web (Essential JS 2)/JavaScript/ej2-documenteditor/styles/material.css`

Step 3: Create a folder `app/resources` and copy/paste the global scripts and styles from the above installed location to `app/resources` location.

Step 4: Create a HTML page (`index.html`) in `app` location and add the Essentials JS 2 script and style references.

```
`html
<!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 -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-office-chart.min.js" type="text/javascript"></script>
<!-- EJ2 Document Editor script -->
<script src="resources/scripts/ej2-documenteditor.min.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

Step 5: Now, add the `Div` element and initiate the `Essential JS 2 DocumentEditor` component in the `index.html` by using following code

```
`html
<!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 -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-office-chart.min.js" type="text/javascript"></script>
<!-- EJ2 Document Editor script -->
<script src="resources/scripts/ej2-documenteditor.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
```

```
<div id='DocumentEditor'></div>

<script>

// Initialize DocumentEditor component.

var documenteditor = new ej.documenteditor.DocumentEditor({ isReadOnly: false, serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/' });

documenteditor.acceptTab = true;

//Enable all the built in modules.

documenteditor.enableAllModules();

documenteditor.pageOutline = '#EOEOEO';

//Documenteditor control rendering starts

documenteditor.appendTo('#DocumentEditor');

</script>

</body>

</html>
`
```

Step 6: Now, run the `index.html` in web browser, it will render the **Essential JS 2 DocumentEditor** component.

Step 7: To render DocumentEditorContainer component, add the `Div` element and initiate the **Essential JS 2 DocumentEditorContainer** component in the `index.html` by using following code

```
`html
<!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 -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-office-chart.min.js" type="text/javascript"></script>
<!-- EJ2 Document Editor script -->
<script src="resources/scripts/ej2-documenteditor.min.js" type="text/javascript"></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 =
'https://services.syncfusion.com/js/production/api/documenteditor/import';
```



```
//DocumentEditorContainer control rendering starts
documenteditorContainer.appendTo('#DocumentEditor');
</script>
</body>
</html>
`
```

Now, run the `index.html` in web browser, it will render the **Essential JS 2 DocumentEditorContainer** component.

Using CDN link for script and style reference

Step 1: Create an app folder `app` for the Essential JS 2 JavaScript components.

Step 2: The Essential JS 2 component's global scripts and styles are already hosted in the below CDN link formats.

Syntax:

Script:

<https://cdn.syncfusion.com/ej2/23.1.36/{PACKAGENAME}/dist/global/{PACKAGENAME}.min.js>

Styles: https://cdn.syncfusion.com/ej2/23.1.36/{PACKAGE_NAME}/styles/material.css

Example:

Script: <https://cdn.syncfusion.com/ej2/23.1.36/ej2-documenteditor/dist/global/ej2-documenteditor.min.js>

Styles: <https://cdn.syncfusion.com/ej2/23.1.36/ej2-documenteditor/styles/material.css>

Step 3: Create a HTML page (`index.html`) in `app` location and add the CDN link references. Now, add the `Div` element and initiate the `Essential JS 2 DocumentEditor` component in the `index.html` by using following code.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- EJ2 Document Editor dependent theme -->
<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-base/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
```

```
<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-navigations/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-splitbuttons/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-dropdowns/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />

<!-- EJ2 Document Editor theme -->

<link href="https://cdn.syncfusion.com/ej2/21.2.3/ej2-documenteditor/styles/material.css"
rel="stylesheet" type="text/css" rel='nofollow' />

<!-- EJ2 Document Editor dependent scripts -->

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-svg-base/dist/global/ej2-svg-
base.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-compression/dist/global/ej2-
compression.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-splitbuttons/dist/global/ej2-splitbuttons.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>
```

```

<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-office-chart/dist/global/ej2-office-chart.min.js"
type="text/javascript"></script>
<!-- EJ2 Document Editor script -->
<script src="https://cdn.syncfusion.com/ej2/21.2.3/ej2-documenteditor/dist/global/ej2-
documenteditor.min.js" type="text/javascript"></script>
</head>
<body>
<!--element which is going to render-->
<div id='DocumentEditor' style='height:350px'></div>
<script>
// Initialize DocumentEditor component.
var documenteditor = new ej.documenteditor.DocumentEditor({ height: '370px', isReadOnly: false,
serviceUrl: 'https://services.syncfusion.com/js/production/api/documenteditor/' });
documenteditor.acceptTab = true;
//Enable all the build in modules.
documenteditor.enableAllModules();
//Set page border color.
documenteditor.pageOutline = '#E0E0E0';
//Documenteditor control rendering starts
documenteditor.appendTo('#DocumentEditor');
</script>
</body>
</html>
`

```

Step 4: Now, run the `index.html` in web browser, it will render the `Essential JS 2 DocumentEditor` component.

Step 5: To render `DocumentEditorContainer` component, add the `Div` element and initiate the `Essential JS 2 DocumentEditorContainer` component in the `index.html` by using following code.

```
`html
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Essential JS 2</title>
<!-- EJ2 Document Editor dependent theme -->
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-base/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-buttons/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-inputs/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-popups/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-lists/styles/material.css" rel="stylesheet"
type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-navigations/styles/material.css"
rel="stylesheet" type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-splitbuttons/styles/material.css"
rel="stylesheet" type="text/css" rel='nofollow' />
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-dropdowns/styles/material.css"
rel="stylesheet" type="text/css" rel='nofollow' />
<!-- EJ2 Document Editor theme -->
<link href="https://cdn.syncfusion.com/ej2/23.1.36/ej2-documenteditor/styles/material.css"
rel="stylesheet" type="text/css" rel='nofollow' />
<!-- EJ2 Document Editor dependent scripts -->
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-base/dist/global/ej2-base.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-data/dist/global/ej2-data.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-svg-base/dist/global/ej2-svg-base.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-file-utils/dist/global/ej2-file-utils.min.js"
type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-compression/dist/global/ej2-
compression.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-pdf-export/dist/global/ej2-pdf-export.min.js"
type="text/javascript"></script>
```

```
<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-buttons/dist/global/ej2-buttons.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-popups/dist/global/ej2-popups.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-splitbuttons/dist/global/ej2-
splitbuttons.min.js" type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-inputs/dist/global/ej2-inputs.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-lists/dist/global/ej2-lists.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-navigations/dist/global/ej2-navigations.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-dropdowns/dist/global/ej2-dropdowns.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-calendars/dist/global/ej2-calendars.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-charts/dist/global/ej2-charts.min.js"
type="text/javascript"></script>

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-office-chart/dist/global/ej2-office-chart.min.js"
type="text/javascript"></script>

<!-- EJ2 Document Editor script -->

<script src="https://cdn.syncfusion.com/ej2/23.1.36/ej2-documenteditor/dist/global/ej2-
documenteditor.min.js" type="text/javascript"></script>

</head>

<body>

<!--Element which is going to render as Document Editor-->

<div id='DocumentEditor' style='height:620px'>

</div>

<script>

// Initialize DocumentEditorContainer component.

var documenteditorContainer = new ej.documenteditor.DocumentEditorContainer({ enableToolbar:
true, height: '590px' });

ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);

documenteditorContainer.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';

//DocumentEditorContainer control rendering starts

documenteditorContainer.appendTo('#DocumentEditor');
```

```
</script>
</body>
</html>
`
```

Now, run the `index.html` in web browser, it will render the `Essential JS 2 DocumentEditorContainer` component.

Server side dependencies

The Document Editor component requires server-side interactions for the following operations:

- [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.

Please refer the [example from GitHub](#) to configure the web service and set the [serviceUrl](#).

Syncfusion provides a predefined [Word Processor server docker image](#) targeting ASP.NET Core 2.1 framework. You can directly pull this docker image and deploy it in server on the go. You can also create own docker image by customizing the existing [docker project from GitHub](#).

Note: Staring from `v19.3.0.x`, we have optimized the accuracy of text size measurements such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API [to disable it and retain the document pagination behavior of older versions](#)..

Frequently Asked Questions

- [How to localize the Documenteditor container.](#)
- [How to load the document by default.](#)
- [How to customize tool bar.](#)
- [How to resize Document editor component?](#)

Feature module in EJ2 JavaScript 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
--------	---	---

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 });|
```



```
|StyleDialog|DocumentEditor.Inject(Selection, Editor, StyleDialog,StyleDialog)|let
documenteditor: DocumentEditor = new DocumentEditor({ isReadOnly: false, enableEditor:
true, enableStyleDialog: true ,enableStyleDialog: 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 EJ2 JavaScript 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.TS

```
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
// Initialize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
let sfdt: string = `{
  "sections": [
    {
      "blocks": [
        {
          "inlines": [
            {
              "characterFormat": {
                "bold": true,
                "italic": true
              },
              "text": "Hello World"
            }
          ]
        }
      ]
    },
    "headersFooters": {
    }
  ]
}`;
//Open the sfdt document in the Document Editor.
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">

<script src="https://cdn.syncfusion.com/ej2/24.1.41/dist/ej2.min.js"
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.TS

```

import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
// Initialize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
document.getElementById('file_upload').setAttribute('accept', '.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') {
            let fileReader: FileReader = new FileReader();
            fileReader.onload = (e: any) => {
                let contents: string = e.target.result;
                //Open the sfdt document in the Document Editor.
                documenteditor.open(contents);
            };

```

```

        //Read the file content.
        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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, FormatType, Selection, Editor, SfdtExport } from
 '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(SfdtExport, Selection, Editor);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableSfdtExport: true, enableSelection: true, enableEditor: true,
isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', () => {
    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>
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.TS

```

import { DocumentEditor, FormatType, Selection, Editor, SfdtExport,
WordExport } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(SfdtExport, WordExport, Selection, Editor);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableWordExport: true, enableSelection: true, enableEditor: true,
isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', () => {
    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.TS

```

import { DocumentEditor, FormatType, Selection, Editor, SfdtExport,
TextExport} from '@syncfusion/ej2-documenteditor';
//Inject require modules for Export.
DocumentEditor.Inject(SfdtExport, TextExport, Selection, Editor);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableTextExport: true, enableSelection: true, enableEditor: true,
isReadOnly: false });
documenteditor.appendTo('#DocumentEditor');
document.getElementById('export').addEventListener('click', () => {
    //Export the document as text file.
    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, SfddExport } from '@syncfusion/ej2-
documenteditor';
```

```
//Inject require modules for Export.
```

```
DocumentEditor.Inject(WordExport, SfddExport);
```

```
let documenteditor: DocumentEditor = new DocumentEditor({ enableSfddExport: true,
enableWordExport: true, enableTextExport: true });
```

```
documenteditor.appendTo('#DocumentEditor');
```

```
documenteditor.open(sfdd);
```

```
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](#)

Web services in EJ2 JavaScript Document editor control

You can deploy web APIs for server-side dependencies of Document Editor component in the following platforms.

- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Java](#)

Which operations require server-side interaction

|Operations|When client-server communication will be triggered?|What type of data will be transferred between client and server?|

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

|[Open file formats other than SFDT](#)|When opening the document other than SFDT (Syncfusion Document Editor's native file format), the server-side web API is invoked from client-side script. | **Client:** Sends the input file.
Server: Receives the input file and sends the converted SFDT back to the client.

The server-side web API internally extracts the content from the document (DOCX, DOC, WordML, RTF, HTML) using Syncfusion Word library (DocIO) and converts it into SFDT for opening the document in Document Editor. |

|[Paste with formatting](#)|When pasting the formatted content (HTML/RTF) received from system clipboard. For converting HTML/RTF to SFDT format.

 Note: Whereas plain text received from system clipboard will be pasted directly in the client-side. | **Client:** Sends the input Html or Rtf string.
Server: Receives the input Html or Rtf string and sends the converted SFDT back to the client. |

|[Restrict editing](#)|When protecting the document, for generating hash. | **Client:** Sends the input data for hashing algorithm.
 Server: Receives the input data for hashing algorithm and sends the result hash information back to the client. |

|[Spellcheck](#)(default)|When the spellchecker is enabled on client-side Document Editor, and it performs the spell check validation for words in the document. | **Client:** Sends the words (string) with their language for spelling validation.
 Server: Receives the words (string) with their language for spelling validation and sends the validation result as JSON back to the client. |

|[SpellCheckByPage](#)|Document editor provides options to spellcheck page by page when loading the documents. By [enabling optimized spell check](#) in client-side, you can perform spellcheck page by page when loading the documents. | **Client:** Sends the words (string) with their language for spelling validation.
 Server: Receives the words (string) with their language for spelling validation and sends the validation result as JSON back to the client. |

|[Save as file formats other than SFDT and DOCX](#) (optional API)|You can configure this API, if you want to save the document in file format other than DOCX and SFDT.

 For saving the files as WordML,

DOC, RTF, HTML, ODT, Text using Syncfusion Word library (DocIO) and PDF using Syncfusion Word (DocIO) and PDF libraries. | You can transfer document from client to server either as SFDT or DOCX format.

First option (SFDT):
Client: Sends the SFDT.
Server: Receives the SFDT and saves the converted document as any file format supported by [Syncfusion Word library \(DocIO\)](#) in server or sends the saved file to the client browser.

Second option (DOCX):
Client: Sends the DOCX file.
Server: Receives the DOCX file and saves the converted document as any file format supported by [Syncfusion Word library \(DocIO\)](#) in server or sends the saved file to the client browser. |

Note: If you don't require the above functionalities then you can deploy as pure client-side component without any server-side interactions.

Please refer the [example from GitHub](#) to configure the web service and set the [serviceUrl](#).

If your running web service Url is `http://localhost:62869/`, set the serviceUrl like below:

```
`ts
```

```
container.serviceUrl = "http://localhost:62869/api/documenteditor/";
```

```
,
```

Required Web API structure

Please check below table for expected web API structure.

Expected method name	Parameters	Return type
----------------------	------------	-------------

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

Import	Files(IFormCollection)	json(sfdd format)
--------	------------------------	-------------------

SystemClipboard	CustomerParameter: content(type string either rtf or html) and type(either .rtf or .html)	json(sfdd format)
-----------------	---	-------------------

RestrictEditing	Parameter of type CustomRestrictParameter public class CustomRestrictParameter { public string passwordBase64 { get; set; } public string saltBase64 { get; set; } public int spinCount { get; set; } } result hash information
-----------------	--

SpellCheck(default)	Parameter: SpellCheckJsonData public class SpellCheckJsonData { public int LanguageID { get; set; } public string TexttoCheck { get; set; } public bool CheckSpelling { get; set; } public bool CheckSuggestion { get; set; } public bool AddWord { get; set; } } Json type of Spellcheck containing details of spell checked word
---------------------	---

SpellCheckByPage	Parameter: SpellCheckJsonData public class SpellCheckJsonData { public int LanguageID { get; set; } public string TexttoCheck { get; set; } public bool CheckSpelling { get; set; } public bool CheckSuggestion { get; set; } public bool AddWord { get; set; } } Json type of Spellcheck containing details of spell checked word
------------------	---

 Note: Document editor provides options to spellcheck page by page when loading the documents. By [enabling optimized spell check](#) in client-side, you can perform spellcheck page by page when loading the documents. |

Save(optional API)	parameter: SaveParameter public class SaveParameter { public string Content { get; set; } public string FileName { get; set; } } void(Save the file as file stream)
--------------------	--

|ExportSFDT(optional API) |parameter: SaveParameter
public class SaveParameter
{
public string Content { get; set; }
 public string FileName { get; set; }
 } |FileStreamResult (to save the document in client-side) |

|Export(optional API) |Files(IFormCollection) |FileStreamResult (to save the document in client-side) |

Customize the expected method name

Document editor component provides an option to customize the expected method name for Import, SystemClipboard, RestrictEditing and SpellCheck using [serverActionSettings](#).

The following example code illustrates how to customize the method name using serverActionSettings.

```
`javascript
var container = new ej.documenteditor.DocumentEditorContainer({ enableToolbar: true, height: '590px'
,enableSpellCheck:true});
ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
container.serviceUrl = hostUrl + 'api/documenteditor/';
// Customize the API name
var settings = { import: 'Import1', systemClipboard: 'SystemClipboard1', spellCheck: 'SpellCheck1',
restrictEditing: 'RestrictEditing1' }
container.serverActionSettings = settings;
container.appendTo('#container');
`
```

Modify the XMLHttpRequest before request send

Document editor component provides an option to modify the XMLHttpRequest object (setting additional headers, if needed) using [beforeXmlHttpRequestSend](#) event and it gets triggered before a server request.

You can customize the required [XMLHttpRequest](#) properties.

The following example code illustrates how to modify the XMLHttpRequest using beforeXmlHttpRequestSend.

```
`javascript
var container = new ej.documenteditor.DocumentEditorContainer({
enableToolbar: true,
height: '590px',
});
//Here, modifying the request headers
container.headers = [{ syncfusion: 'true' }];
// Below action, cancel all server-side interactions expect spell check
container.beforeXmlHttpRequestSend = function(args) {
args.headers = container.headers;
`
```



```
args.withCredentials = true;
switch (args.serverActionType) {
case 'Import':
case 'RestrictEditing':
case 'SystemClipboard':
args.cancel = true;
break;
}
};
container.appendTo('#container');
```

Note: Find the customizable serverActionType values are `'Import'` | `'RestrictEditing'` | `'SpellCheck'` | `'SystemClipboard'`.

Server Deployment

[Word processor server docker image overview in EJ2 JavaScript 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.

Step 4: Append the Docker instance running the URL (`http://localhost:6002/api/documenteditor`) to the service URL in the client-side Word Processor control. For more information about how to get started with the Word Processor control, refer to this [getting started page](#).

```
<!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 -->
<script src="resources/scripts/ej2-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-data.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-svg-base.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-file-utils.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-compression.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-pdf-export.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-buttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-popups.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-splitbuttons.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-inputs.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-lists.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-navigations.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-dropdowns.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-calendars.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-charts.min.js" type="text/javascript"></script>
<script src="resources/scripts/ej2-office-chart.min.js" type="text/javascript"></script>
<!-- EJ2 Document Editor script -->
<script src="resources/scripts/ej2-documenteditor.min.js" type="text/javascript"></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-processor-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 EJ2 JavaScript 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.

Append the app service running the URL `http://documenteditor-server.azurewebsites.net/api/documenteditor/` to the service URL in the client-side Document Editor control. For more information about the Document Editor control, refer to this [getting started page](#).

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 EJ2
JavaScript 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 [kubectli](#) into the workspace using the following command.

`

```
az aks install-cli
```

`

To configure `kubectli` 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.

Step 6: Append the Kubernetes service running the URL `http://<external-ip>/api/documenteditor/` to the service URL in the client-side Document Editor control. For more information about the Document Editor control, refer to this [getting started page](#).

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 EJ2
JavaScript Document editor control

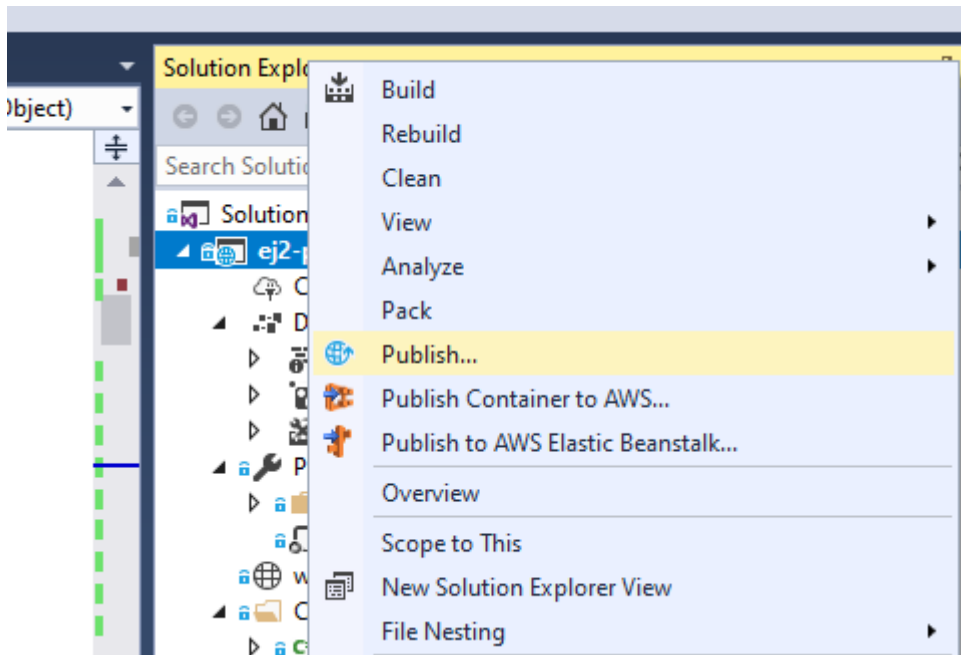
Prerequisites

- 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.

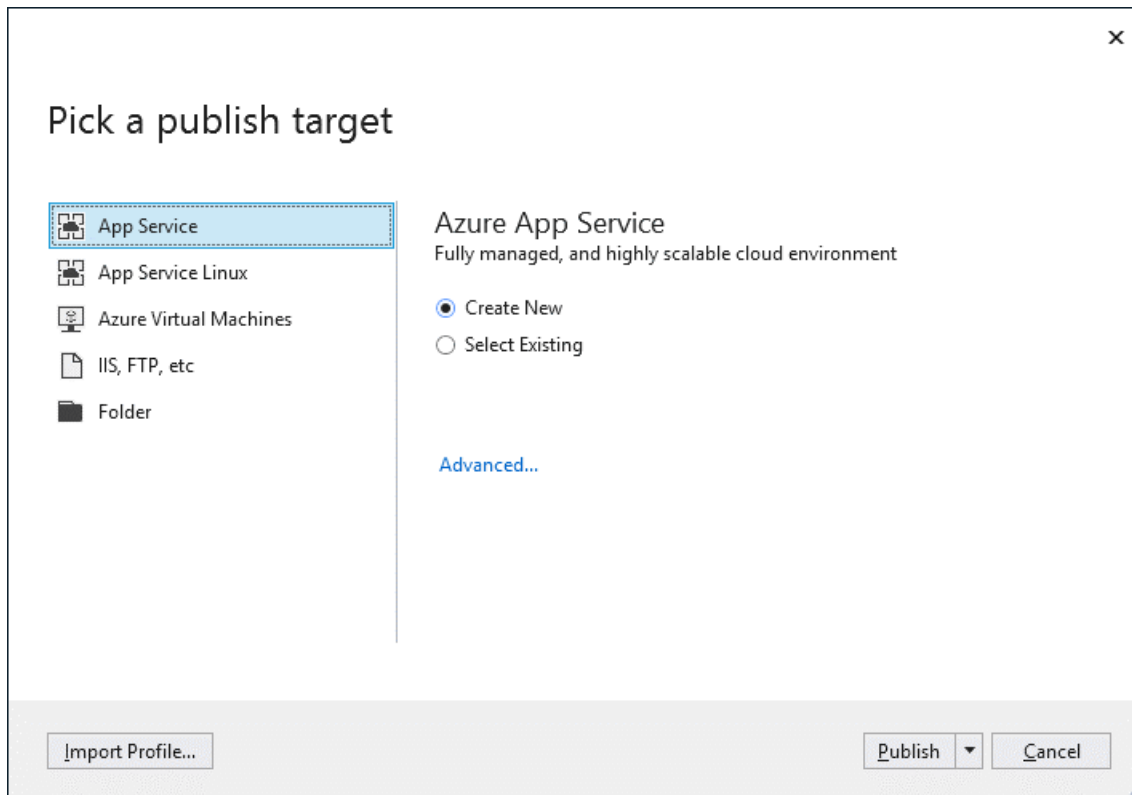
Publish to Azure App Service

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.

Append the app service running the URL <http://ej2-documenteditor-server20200514102909.azurewebsites.net/api/documenteditor> to the service URL in the client-side Document Editor control. For more information about how to get started with the Document Editor control, refer to this [getting started page](#).

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 EJ2 JavaScript 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.

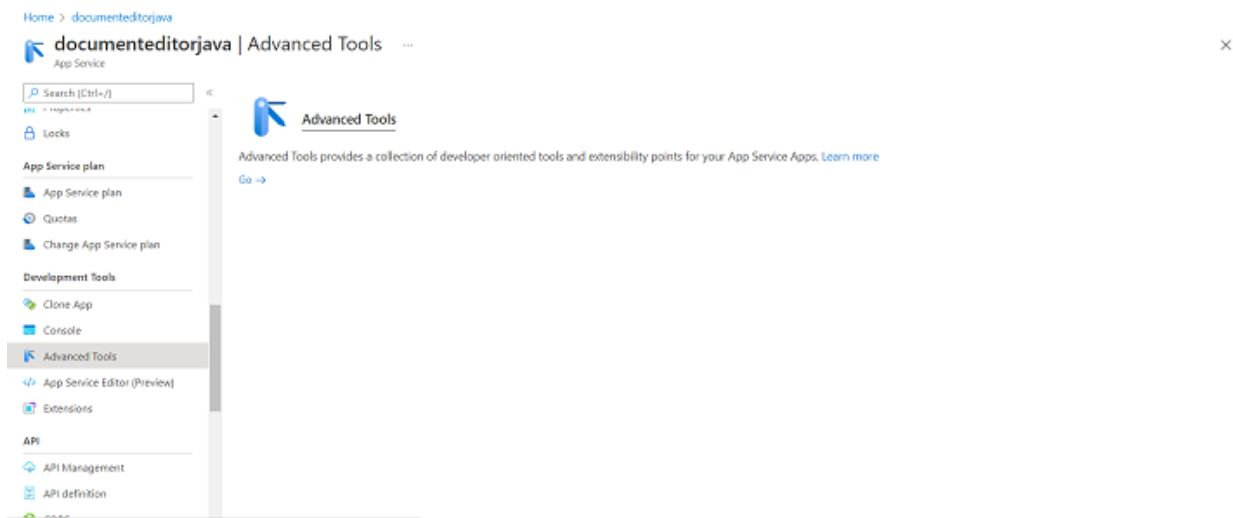
```
`console  
mvn package  
`
```

Above package generation command creates the **tomcat-0.0.1-SNAPSHOT.war** in the below location in the sample folder.

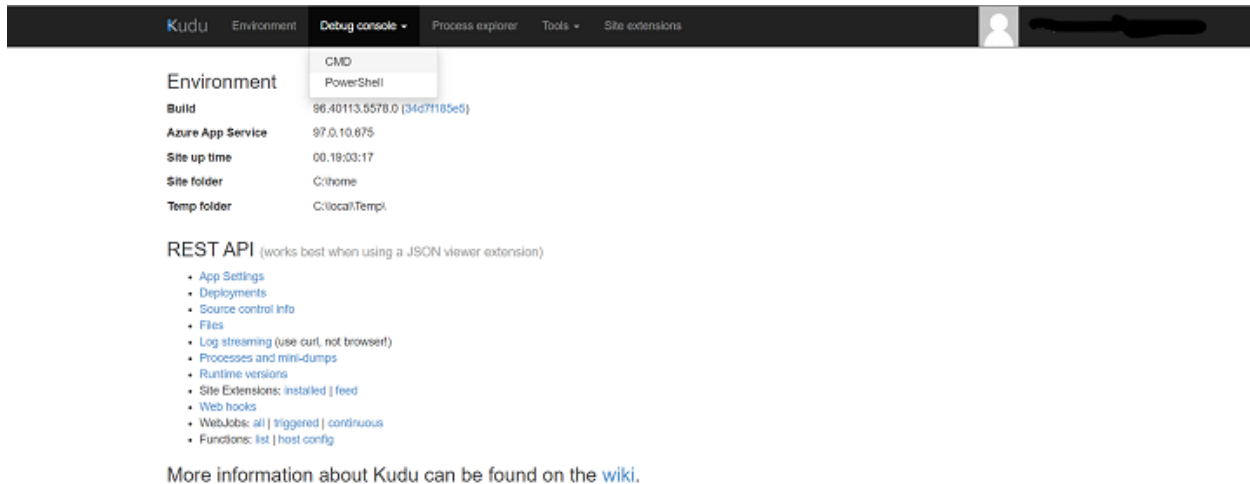
target/tomcat-0.0.1-SNAPSHOT.war

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**.



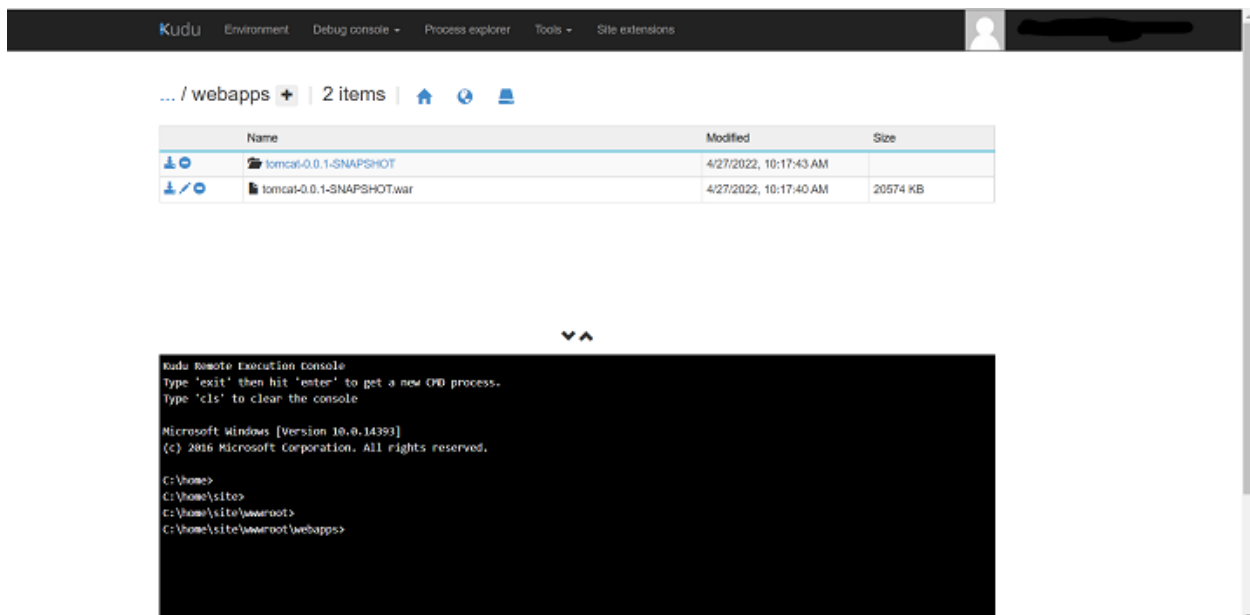
Then, click **Go** and select the **CMD** options under **Debug console**.



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.

Append the app service running the URL `http://documenteditorjava.azurewebsites.net/tomcat-0.0.1-SNAPSHOT` to the service URL in the client-side Document Editor control. For more information about the Document Editor control, refer to this [getting started page](#).

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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor, Selection, ImageResizer, EditorHistory }
from '@syncfusion/ej2-documenteditor';
import { Button } from '@syncfusion/ej2-buttons';
//Inject require modules.
DocumentEditor.Inject(Editor, Selection, ImageResizer, EditorHistory);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableEditor: true,
  enableSection: true,
  enableImageResizer: true,
  enableEditorHistory: true,
  height: '370px'
});
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');
let insertPictureButton: Button = new Button();
insertPictureButton.appendTo('#insert-picture');
//Open file picker.
document.getElementById('insert-picture').addEventListener('click', () => {
  let pictureUpload: HTMLInputElement =
document.getElementById("insertImageButton") as HTMLInputElement;
  pictureUpload.value = '';
  pictureUpload.click();
});
// File picker change event.
document.getElementById('insertImageButton').addEventListener('change',
onInsertImage);
//Get select image from file picker and insert it in Document Editor.
function onInsertImage(args: any): void {
  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]) {
      let path = args.target.files[0];
```

```
let reader = new FileReader();
reader.onload = function (frEvent: any) {
    let base64String = frEvent.target.result;
    let image = document.createElement('img');
    image.addEventListener('load', function () {
        //Insert image.
        documenteditor.editor.insertImage(base64String,
this.width, this.height);
    })
    image.src = base64String;
};
reader.readAsDataURL(path);
}
//Safari does not Support FileReader Class
} else {
    let 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>
<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

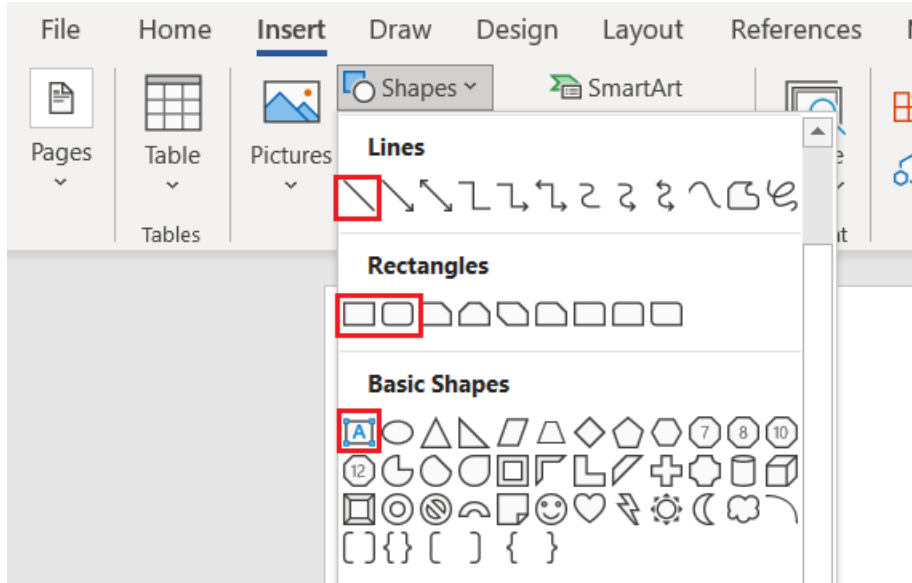
- [Feature modules](#)

Shapes in EJ2 JavaScript 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 American, European and Asian operation is located in Bothell, several regional sales teams are located throughout their market base.

Adventure Works
Cycles

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 EJ2 JavaScript 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 centers are 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 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.

Adventure works cycles
company.

Bookmark in EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor, Selection, SfddtExport, EditorHistory,
BookmarkDialog } from '@syncfusion/ej2-documenteditor';
// Enable requir modules
DocumentEditor.Inject(Editor, Selection, SfddtExport, EditorHistory,
BookmarkDialog);
// Initialize Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableEditor: true,
  enableSelection: true,
  enableEditorHistory: true,
  enableBookmarkDialog: true,
  height: '590px'
});
documenteditor.appendTo('#DocumentEditor');
document.getElementById('dialog').addEventListener('click', () => {
  //Open bookmark dialog.
  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>
```

See Also

- [Feature modules](#)
- [Bookmark dialog](#)

Link in EJ2 JavaScript 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.TS

```
import { DocumentEditor, SfdtExport, Selection, RequestNavigateEventArgs }
from '@syncfusion/ej2-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Selection, SfdtExport);
//Initilize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ enableSelection:
true, height: '370px' });
documenteditor.appendTo('#DocumentEditor');
// Add event listener for requestNavigate event to customize hyperlink
navigation functionality
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;
    }
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
    <title>EJ2 Animation</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=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>
<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;
```

```
}
};
`ts
```

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.TS

```
import { DocumentEditor, SfdtExport, Selection, Editor,
RequestNavigateEventArgs } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Selection, SfdtExport, Editor);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
isReadOnly: false, enableSelection: true, enableEditor: true });
documenteditor.appendTo('#DocumentEditor');
// Add event listener for requestNavigate event to customize hyperlink
navigation functionality.
documenteditor.requestNavigate = (args: RequestNavigateEventArgs) => {
    if (args.linkType !== 'Bookmark') {
        let link: string = 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.

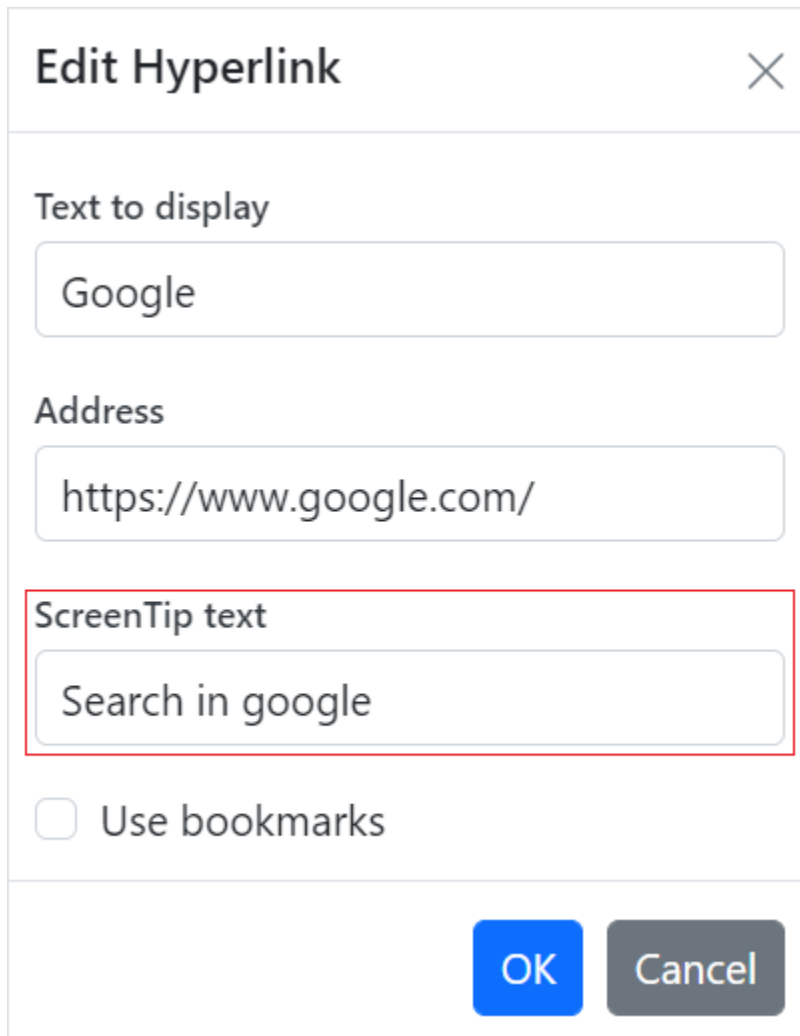
```
`ts
documenteditor.insertHyperlink('https://www.google.com', 'Google');
`
```

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.TS

```
import { DocumentEditor, Editor, Selection, EditorHistory, HyperlinkDialog,
SfdtExport } from '@syncfusion/ej2-documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, EditorHistory, HyperlinkDialog,
SfdtExport);
```

```
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditorHistory: true,
  enableEditor: true,
  enableHyperlinkDialog: true,
  enableSfdtExport: true,
  height: '370px'
});
//Click the hyperlink button, the hyperlink dialog will open
function showHyperlinkDialog() {
  //Open the hyperlink dialog.
  documenteditor.showDialog('Hyperlink');
}
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', showHyperlinkDialog)
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">
```



```

        <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>

```

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 EJ2 JavaScript 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
```

```
documenteditor.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.TS

```
import { DocumentEditor, Editor, Selection, SfdtExport, EditorHistory,
TableDialog, ContextMenu } from '@syncfusion/ej2-documenteditor';
import { Toolbar } from '@syncfusion/ej2-navigations';
DocumentEditor.Inject(Editor, Selection, EditorHistory, TableDialog,
ContextMenu, SfdtExport);
let documenteditor: DocumentEditor = new 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;
    }
}
```

```
let toolBar: Toolbar = new 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-de-delete-table e-icons',
      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');
//Insert table.
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 EJ2 JavaScript 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

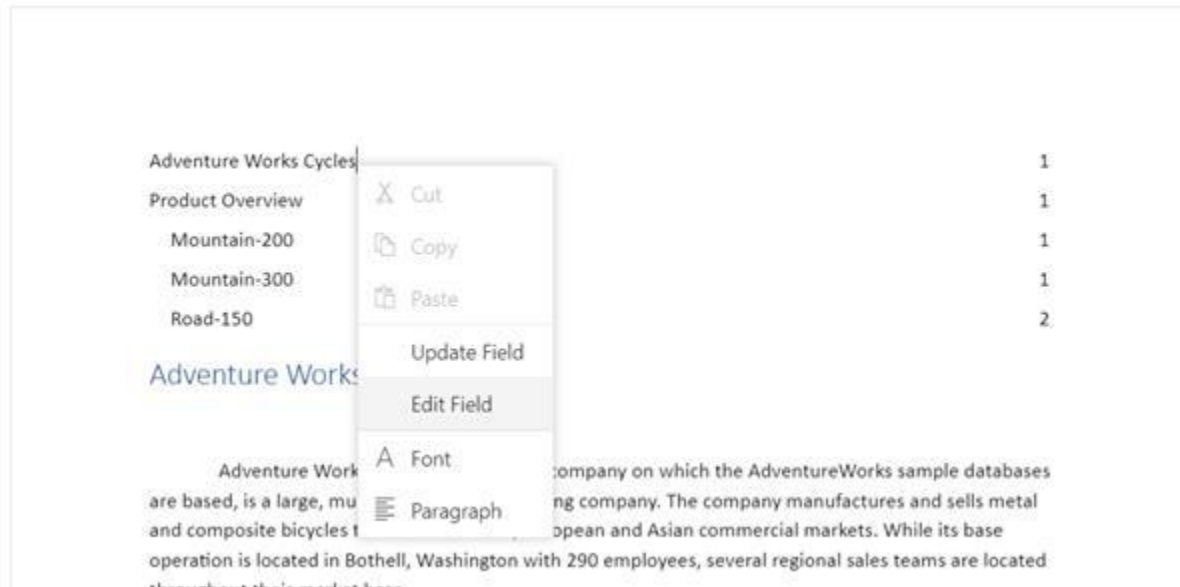
```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css"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

```
`typescript
container.documentEditor.selection.sectionFormat.oddPageHeader.linkToPrevious = false;
container.documentEditor.selection.sectionFormat.oddPageFooter.linkToPrevious = false;
```

In case the document has different header and footer types, such as different first page, odd, and even pages.

```
`typescript
// 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;
```

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 EJ2 JavaScript 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;
`
```

You can toggle the bold formatting based on existing value at selection. Refer to the following sample code.

```
`ts
documenteditor.editor.toggleBold();
`
```

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;
`
```

You can toggle the Italic formatting based on existing value at selection. Refer to the following sample code.

```
`ts
documenteditor.editor.toggleItalic();
`
```

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';
```

```
,
```

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');
```

```
,
```

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';
```

```
,
```

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();
```

```
,
```

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;
```

`

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';
```

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';
```

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';
```

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.TS

```
import { DocumentEditor, Editor, Selection, EditorHistory, SfdtExport } from
 '@syncfusion/ej2-documenteditor';
import { Toolbar } from '@syncfusion/ej2-navigations';
import { ComboBox } from '@syncfusion/ej2-dropdowns';
import { ColorPicker } from '@syncfusion/ej2-inputs';
//Inject required modules.
DocumentEditor.Inject(Editor, Selection, EditorHistory, SfdtExport);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
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) {
    let toggleBtn: HTMLElement = 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");
        }
    }
}
let fontStyle: string[] = ['Algerian', 'Arial', 'Calibri', 'Cambria',
'Cambria Math', 'Candara', 'Courier New', 'Georgia', 'Impact', 'Segoe
Print', 'Segoe Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana',
'Windings'
];
let fontSize: string[] = ['8', '9', '10', '11', '12', '14', '16', '18',
'20', '22', '24', '26', '28', '36', '48', '72', '96'];
let toolBar: Toolbar = new 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 ColorPicker({
                value: '#000000',
                showButtons: true,
                change: changeFontColor
            })
        },
    ],
});

```

```

        { type: 'Separator' },
        {
            type: 'Input',
            template: new ComboBox({
                dataSource: fontStyle,
                width: 120,
                index: 2,
                allowCustom: true,
                change: changeFontFamily,
                showClearButton: false,
            }),
        },
        {
            type: 'Input',
            template: new 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>
</body>
<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 EJ2 JavaScript 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";
`
```

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;
`
```

Toolbar with paragraph formatting options

The following sample demonstrates the paragraph formatting options using a toolbar.

INDEX.TS

```
import { DocumentEditor, Editor, Selection, EditorHistory, SfdtExport,
ContextMenu } from '@syncfusion/ej2-documenteditor';
import { Toolbar } from '@syncfusion/ej2-navigations';
import { ItemModel, DropDownButton } from '@syncfusion/ej2-splitbuttons';
//Inject the require module
DocumentEditor.Inject(Editor, Selection, EditorHistory, SfdtExport,
ContextMenu);
let documenteditor: DocumentEditor = new DocumentEditor({
    height: '370px', isReadOnly: false, enableSelection: true,
    enableEditorHistory: true, enableEditor: true, enableContextMenu: true,
    enableSfdtExport: true
});
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':
        //Clear all formatting of the selected paragraph or content.
        documenteditor.editor.clearFormatting();
        break;
    case 'ShowParagraphMark':
        //Show or hide the hidden characters like spaces, tab, paragraph
marks, and breaks.
        documenteditor.documentEditorSettings.showHiddenMarks =
!documenteditor.documentEditorSettings.showHiddenMarks;
        break;
    }
}
//Change the line spacing of selected or current paragraph
function lineSpacingAction(args: any) {
    let text: string = 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(() : void => {
        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', 'ShowParagraphMark'];
        for (var i = 0; i < toggleBtnId.length; i++) {
            let toggleBtn: HTMLElement =
document.getElementById(toggleBtnId[i]);
            //Remove toggle state.

```

```

        toggleBtn.classList.remove('e-btn-toggle');
    }
    //Add toggle state based on selection paragraph format.
    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');
    }
    if (documenteditor.documentEditorSettings.showHiddenMarks) {
        document.getElementById('ShowParagraphMark').classList.add('e-
btn-toggle');
    }
    // #endregion
}
}
//Toolbar configuration to add paragraph formatting options
let toolBar: Toolbar = new 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',
        },
        {
            type: 'Separator'
        },
        {
            prefixIcon: 'e-de-e-paragraph-mark e-icons',
            tooltipText: 'Show the hidden characters like spaces, tab,
paragraph marks, and breaks.(Ctrl + *)',
            id: 'ShowParagraphMark',
        }
    ],
});
toolBar.appendTo('#toolbar');
let items: ItemModel[] = [
    {
        text: 'Single',
    },
    {
        text: '1.15',
    },
    {
        text: '1.5',
    },
    {
        text: 'Double',
    },
];
let dropdown = new 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>
<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](#)
- [Paragraph dialog](#)
- [Keyboard shortcuts](#)

Styles in EJ2 JavaScript Document editor control

Styles are useful for applying a set of formatting consistently throughout the document. In Document Editor, styles are created and added to a document programmatically or via the built-in Styles dialog.

Styles definition overview

A Style in Document Editor should have the following properties:

- **name:** Name of the style. All styles in a document have a unique name, which is used as an identifier when applying the style.
- **type:** Specifies the document elements that the style will target. For example, paragraph or character.
- **next:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined.
- **link:** Provides a relation between the paragraph and character style.
- **characterFormat:** Specifies the properties of paragraph and character style.
- **paragraphFormat:** Specifies the properties of paragraph style.
- **basedOn:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined. It can be optional.

The style type should match the inherited style type. For example, it is not possible to have a character style inherit a paragraph style.

Default style

The default style for span and paragraph properties is normal. It internally inherits the default style of the document loaded or Document Editor component.

Style hierarchy

Each style initially checks its local value for the property that is being evaluated and turns to the style it is based on. If no local value is found, it turns to its default style.

Style inheritance of different styles are listed as follows:

Character style

Character styles are based only on other character styles.

The inheritance is: Character properties are inherited from the base character style.

Paragraph style

Paragraph styles are based on other paragraph styles or on linked styles. When a paragraph style is based on another paragraph style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the base paragraph style.
- Span properties are inherited from the base paragraph style.

When a paragraph style is based on a linked style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Linked style

Linked styles are composite styles and their components are paragraph and character styles with link between them. To apply paragraph properties, take the properties from the linked paragraph style. Similarly, to apply character properties, take the properties from linked character style. Linked styles are based on other linked styles or on paragraph styles.

When a linked style is based on a paragraph style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the 'basedOn' paragraph style.

- Character properties are inherited from the 'basedOn' paragraph style.

When a linked style is based on another linked style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Defining new styles

New Styles are defined and added to the style collection of the document. In this way, they will be discovered by the default UI and applied to the parts of a document.

Defining a character style

The following example shows how to programmatically create a character style.

```
`javascript
//Initialize Document Editor component.

var documentEditor = new ej.documenteditor.DocumentEditor({ enableEditor: true, isReadOnly: false,
enableSelection: true });

// Create custom style object.
var styleJson = {
  "type": "Character",
  "name": "New CharacterStyle",
  "basedOn": "Default Paragraph Font",
  "characterFormat": {
    "fontSize": 16.0,
    "fontFamily": "Calibri Light",
    "fontColor": "#2F5496",
    "bold": true,
    "italic": true,
    "underline": "Single"
  }
};

//Created new style using createStyle method.
documentEditor.editor.createStyle(JSON.stringify(styleJson));
`
```

Defining a paragraph style

The following example shows how to programmatically create a paragraph style.

```
`javascript
//Initialize Document Editor component.
```

```
var documentEditor = new ej.documenteditor.DocumentEditor({ enableEditor: true, isReadOnly: false,
enableSelection: true });
// Create custom style object.
var styleJson = {
  "type": "Paragraph",
  "name": "New ParagraphStyle",
  "basedOn": "Normal",
  "characterFormat": {
    "fontSize": 16.0,
    "fontFamily": "Calibri Light",
    "fontColor": "#2F5496",
    "bold": true,
    "italic": true,
    "underline": "Single"
  },
  "paragraphFormat": {
    "leftIndent": 0.0,
    "rightIndent": 0.0,
    "firstLineIndent": 0.0,
    "beforeSpacing": 12.0,
    "afterSpacing": 0.0,
    "lineSpacing": 1.0791666507720947,
    "lineSpacingType": "Multiple",
    "textAlignment": "Left",
    "outlineLevel": "Level1"
  }
};
//Created new style using createStyle method.
documentEditor.editor.createStyle(JSON.stringify(styleJson));
`
```

Defining a linked style

The following example shows how to programmatically create linked style.

```
`javascript
//Initialize Document Editor component.
```

```

var documentEditor = new ej.documenteditor.DocumentEditor({ enableEditor: true,isReadOnly: false,
enableSelection: true });

// Create custom style object.
var styleJson = {
  "type": "Paragraph",
  "name": "New Linked",
  "basedOn": "Normal",
  "next": "Normal",
  "link": "New Linked Char",
  "characterFormat": {
    "fontSize": 16.0,
    "fontFamily": "Calibri Light",
    "fontColor": "#2F5496"
  },
  "paragraphFormat": {
    "leftIndent": 0.0,
    "rightIndent": 0.0,
    "firstLineIndent": 0.0,
    "beforeSpacing": 12.0,
    "afterSpacing": 0.0,
    "lineSpacing": 1.0791666507720947,
    "lineSpacingType": "Multiple",
    "textAlignment": "Left",
    "outlineLevel": "Level1"
  }
};

//Created new style using createStyle method.
documentEditor.editor.createStyle(JSON.stringify(styleJson));

```

Applying a style

The styles are applied using the **applyStyle** method of **editorModule**, the parameter should be passed is the **Name** of the Style.

The styles of the **Character** type is applied to the currently selected part of the document. If there is no selection, the values that will be applied to the word at caret position. The styles of **Paragraph** type follow the same logic and are applied to all paragraphs in the selection or the current paragraph.

When there is no selection, styles of **Linked** type will change the values of the paragraph, and apply both the Paragraph and Character properties. When there is selection, Linked Style changes only the character properties of the selected text.

For example, the following line will apply the "New Linked" to the current paragraph.

```
`javascript
//Apply specified style for selected paragraph.
editor.editorModule.applyStyle('New Linked');
//Clear direct formatting and apply the specified style
editor.editorModule.applyStyle('New Linked', true);
`
```

Get Styles

You can get the styles in the document using the below code snippet.

```
`ts
//Get paragraph styles
let paragraphStyles = documentEditor.getStyles('Paragraph');
//Get character styles
let paragraphStyles = documentEditor.getStyles('Character');
`
```

Modify an existing style

You can modify a existing style with the specified style properties using [createStyle](#) method. If modifyExistingStyle parameter is set to `true` the style properties is updated to the existing style.

The following illustrate to modify an existing style.

```
`ts
let styleJson: any = {
  "type": "Paragraph",
  "name": "Heading 1",
  "characterFormat": {
    "fontSize": 32,
    "fontFamily": "Calibri"
  }
};
documentEditor.editor.createStyle(styleName, true);
`
```

If modifyExistingStyle parameter is set to true and a style already exists with same name, it modifies the specified properties in the existing style.

If modifyExistingStyle parameter is set to false and a style already exists with same name, it creates a new style with unique name by appending '_1'. Hence, the newly style will not have the specified name.

If no style exists with same name, it creates a new style.

List format in EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor, Selection, EditorHistory } from
 '@syncfusion/ej2-documenteditor';
import { Toolbar } from '@syncfusion/ej2-navigations';
//Inject the require module
DocumentEditor.Inject(Editor, Selection, EditorHistory);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditorHistory: true,
  enableEditor: 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;
  }
};
let toolbar: Toolbar = new 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

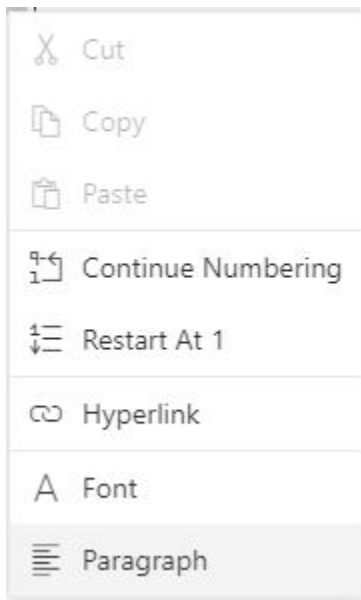
```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Typescript UI Controls">
  <meta name="author" content="Syncfusion">
  <link href="index.css" 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>
<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 EJ2 JavaScript 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, SfddExport } from '@syncfusion/ej2-documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, SfddExport);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditor: true,
  enableSfddExport: 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, SfddExport } from '@syncfusion/ej2-documenteditor';
//Inject the required module
DocumentEditor.Inject(Editor, Selection, SfddExport);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditor: true,

```

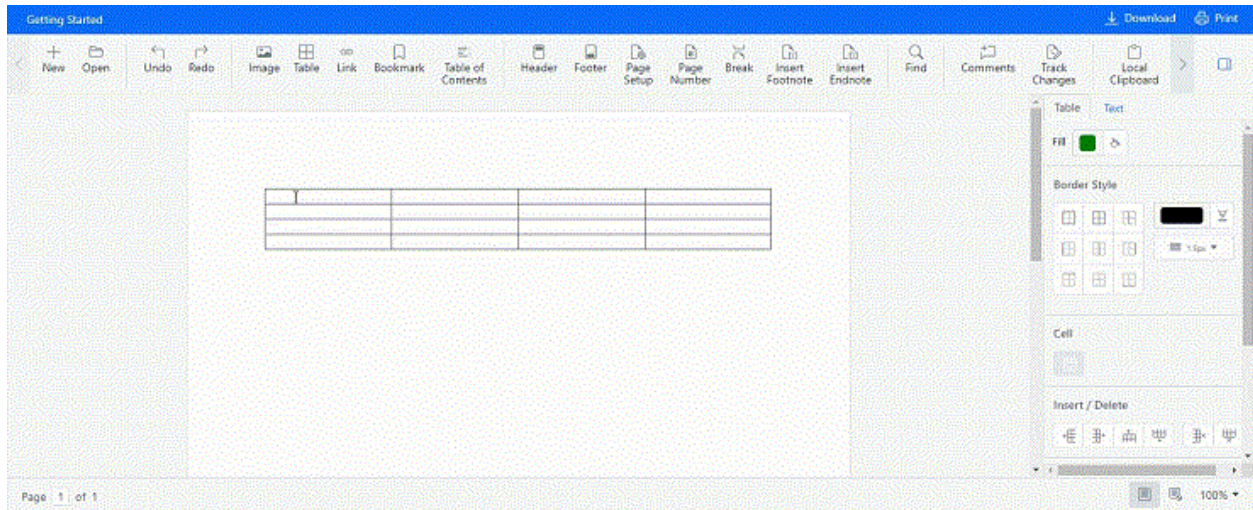
```
enableSfdtExport: true
});
documenteditor.appendTo('#DocumentEditor');
documenteditor.editor.insertTable(2, 2);
//To change the width of a table
documenteditor.selection.tableFormat.preferredWidthType = 'Point';
documenteditor.selection.tableFormat.preferredWidth = 300;
`
```

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 EJ2 JavaScript 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 EJ2 JavaScript 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

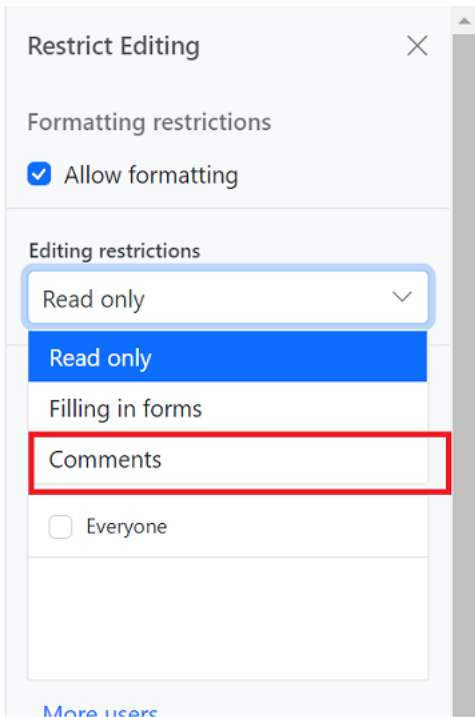
Document Editor provides support for protecting the document with **CommentsOnly** protection. In this protection, user allowed to add or edit comments alone in the document.

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.

Track changes in EJ2 JavaScript Document editor control

Track Changes allows you to keep a record of changes or edits made to a document. You can then choose to accept or reject the modifications. It is a useful tool for managing changes made by several reviewers to the same document. If track changes option is enabled, all editing operations are preserved as revisions in Document Editor.

Enable track changes in Document Editor

The following example demonstrates how to enable track changes.

```
`ts
var container = new ej.documenteditor.DocumentEditorContainer({ enableTrackChanges: true });
container.appendTo('#container');
`
```

Get all tracked revisions

The following example demonstrate how to get all tracked revision from current document.

```
`ts
var container = new ej.documenteditor.DocumentEditorContainer({ enableTrackChanges: true });
container.appendTo('#container');
/
```

- Get revisions from the current document

```
*/
var revisions = container.documentEditor.revisions;
`
```

Accept or Reject all changes programmatically

The following example demonstrates how to accept/reject all changes.

```
`ts
var container = new ej.documenteditor.DocumentEditorContainer({ enableTrackChanges: true });
container.appendTo('#container');
/
```

- Get revisions from the current document

```
*/
var revisions = container.documentEditor.revisions;
/
```

- Accept all tracked changes

```
*/
revisions.acceptAll();
/
```

- Reject all tracked changes

```
*/
revisions.rejectAll();
`
```

Accept or reject a specific revision

The following example demonstrates how to accept/reject specific revision in the Document Editor.

```
`ts
/

• Get revisions from the current document

*/
var revisions = container.documentEditor.revisions;
/
```

- Accept specific changes

```
*/
```

```
revisions.get(0).accept();
```

```
/
```

- Reject specific changes

```
*/
```

```
revisions.get(1).reject();
```

```
`
```

Navigate between the tracked changes

The following example demonstrates how to navigate tracked revision programmatically.

```
`ts
```

```
var container = new ej.documenteditor.DocumentEditorContainer({ enableTrackChanges: true });
```

```
container.appendTo('#container');
```

```
/
```

- Navigate to next tracked change from the current selection.

```
*/
```

```
container.documentEditor.selection.navigateNextRevision();
```

```
/
```

- Navigate to previous tracked change from the current selection.

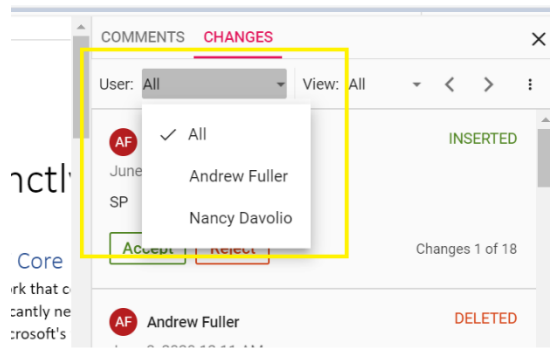
```
*/
```

```
container.documentEditor.selection.navigatePreviousRevision();
```

```
`
```

Filtering changes based on user

In DocumentEditor, we have built-in review panel in which we have provided support for filtering changes based on the user.



Protect the document in track changes only mode

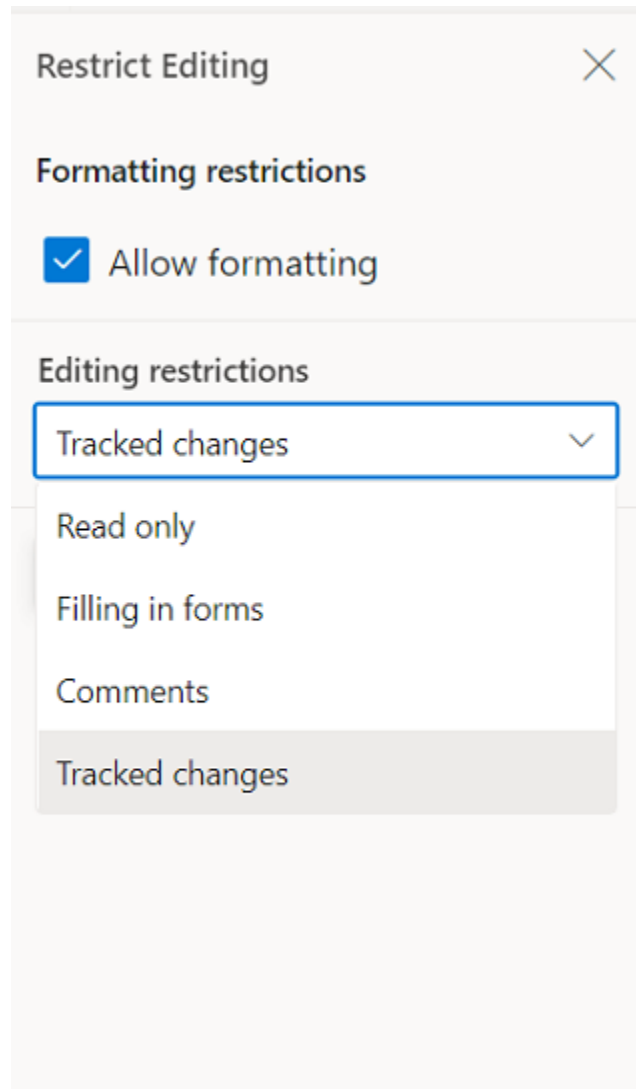
Document Editor provides support for protecting the document with **RevisionsOnly** protection. In this protection, all the users are allowed to view the document and do their corrections, but they cannot accept or reject any tracked changes in the document. Later, the author can view their corrections and accept or reject the changes.

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
var container = new ej.documenteditor.DocumentEditorContainer({
  enableToolbar: true,
  height: '590px',
});
ej.documenteditor.DocumentEditorContainer.Inject(Toolbar);
container.serviceUrl =
'http://localhost:5000/api/documenteditor/';
container.appendTo('#container');
//enforce protection
container.documentEditor.editor.enforceProtection('123', 'RevisionsOnly');
//stop the document protection
container.documentEditor.editor.stopProtection('123');
```

Tracked changes 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` | `RevisionsOnly`. In stop protection method, parameter denotes the password.

Events

DocumentEditor provides [beforeAcceptRejectChanges](#) event, which is triggered before a tracked content is accepted or rejected. This event provides an opportunity to perform custom logic before accepting or rejecting changes. The event handler receives the [RevisionActionEventArgs](#) object as an argument, which allows access to information about the tracked content. .

To demonstrate a specific use case, let's consider an example where we want to restrict the accept and reject changes functionality based on the author name. The following code snippet illustrates how to allow only the author of the tracked content to accept or reject changes:

```
`typescript
var container = new ej.documenteditor.DocumentEditorContainer({
  beforeAcceptRejectChanges:{beforeAcceptRejectChanges},
```

```
enableToolbar: true,
height: '590px',
currentUser: 'Hary'
});
ej.documenteditor.DocumentEditorContainer.Inject(ej.documenteditor.Toolbar);
container.appendTo('#container');
// Event get triggered before accepting/rejecting changes
function beforeAcceptRejectChanges(args) {
// Check the author of the revision and current user are different
if (args.author !== container.currentUser) {
// Cancel the accept/reject action
args.cancel = true;
}
}
,
```

Fields in EJ2 JavaScript 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);
,
```

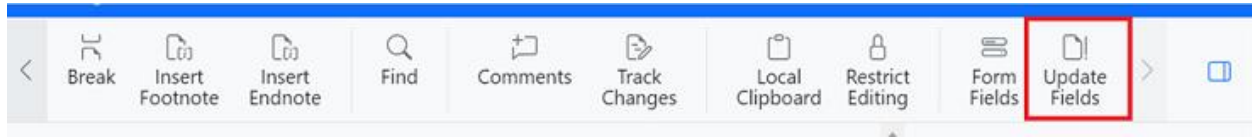
Note: Document editor does not validate/process the field code/field result. it simply inserts the field with specified field information.

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.

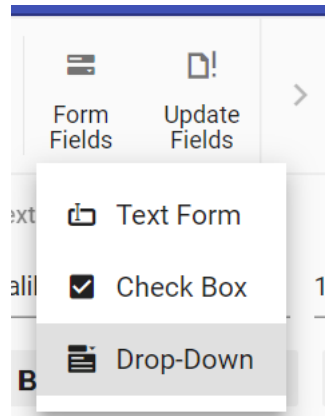
See Also

- [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 EJ2 JavaScript 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 EJ2 JavaScript 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 [Synconfusion.EJ2.WordEditor.AspNet.Core](#) by the web API service implementation. This library helps you to paste the system clipboard data with formatting.

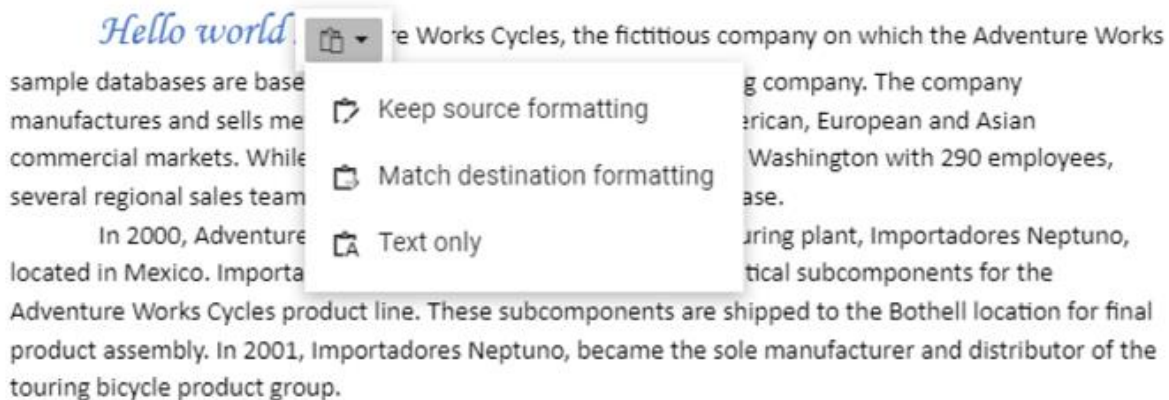
Refer this [page](#) for more details.

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.TS

```
import { DocumentEditorContainer, DocumentEditor,
CollaborativeEditingHandler } from '@syncfusion/ej2-documenteditor';
//Inject collaborative editing module.
DocumentEditor.Inject(CollaborativeEditingHandler);
DocumentEditorContainer.Inject(Toolbar);
let container: DocumentEditorContainer = new 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.TS

```
import { HubConnectionBuilder, HttpTransportType, HubConnectionState } from
 '@microsoft/signalr';
let connectionId: string = "";
var connection = new HubConnectionBuilder().withUrl(serviceUrl +
 '/documenteditorhub', {
    skipNegotiation: true,
    transport: HttpTransportType.WebSockets
}).withAutomaticReconnect().build();
connection.onclose(async () => {
    if (connection.state === HubConnectionState.Disconnected) {
        alert('Connection lost. Please reload the browser to continue.');
```

```

        setTimeout(start, 5000);
    }
};
//Event handler to handle data received
connection.on('dataReceived', onDataRecived.bind(this));
function onDataRecived(action: string, data: any) {
    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.TS

```

function openDocument(responseText: string, roomName: string): void {
    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.TS

```

container.contentChange = function (args: ContainerContentChangeEventArgs) {
    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 = clientVersion;
}
```

```
//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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, Selection, Editor, Search, OptionsPane } from
 '@syncfusion/ej2-documenteditor';
```

```

DocumentEditor.Inject(Selection, Search, Editor, OptionsPane);
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableSelection: true, enableSearch: true, enableEditor: true, isReadOnly:
false, enableOptionsPane: true });
documenteditor.appendTo('#DocumentEditor');
let sfdt: string = `{
  "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.TS

```
import { DocumentEditor, Selection, Editor, Search } from '@syncfusion/ej2-
documenteditor';
//Inject require modules.
DocumentEditor.Inject(Selection, Search, Editor);
```



```
//Initialize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
enableSelection: true, enableSearch: true, enableEditor: true, isReadOnly:
false });
documenteditor.appendTo('#DocumentEditor');
let sfdt: string = `{
  "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."
            }
          ]
        }
      ]
    }
  ]
};`
//Open the SFDT document in Document Editor.
documenteditor.open(sfdt);
document.getElementById('replace_all').addEventListener('click', () => {
  let textToFind: string = (document.getElementById('find_text') as
HTMLInputElement).value;
  let textToReplace: string = (document.getElementById('replace_text') as
HTMLInputElement).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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false, height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
//Open default document in DocumentEditor
onLoadDefault();
documenteditor.scrollToPage(2);
function onLoadDefault(): void {
    let defaultDocument: object = {
        "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.TS

```
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
//Initialize the Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false, height: '370px', serviceUrl:
    'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
onLoadDefaultDocument();
documenteditor.viewer.selection.goToPage(3);
function onLoadDefaultDocument(): void {
    let defaultDocument: object = {
        "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.TS

```
import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
import { createElement } from '@syncfusion/ej2-base';
import { DropDownButton, ItemModel, MenuEventArgs } from '@syncfusion/ej2-splitbuttons';
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false, height: '370px', serviceUrl:
  'https://services.syncfusion.com/js/production/api/documenteditor/'
});
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
let statusBarDiv = document.getElementById('page-fit-type-div');
let startPage: number = 1;
let label: HTMLElement = createElement('label', { styles: 'margin-top:
6px;margin-right: 2px' });
label.textContent = 'Page ';
statusBarDiv.appendChild(label);
let pageNumberLabel = createElement('label', { id: 'documenteditor_page_no',
styles: 'text-transform:capitalize;white-space:pre;overflow:hidden;user-
select:none;cursor:text;height:17px;max-width:150px' });
```

```

let editablePageNumber = createElement('div', { id: 'editablePageNumber',
styles: 'border: 1px solid #F1F1F1;display: inline-flex;height:
17px;padding: 0px 4px;', className: 'single-line e-de-pagenunder-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. ');
let labell: HTMLElement = createElement('label', { id:
'documenteditor_pagecount', styles: 'margin-left:2px;letter-spacing:
1.05px;' });
labell.textContent = 'of';
statusBarDiv.appendChild(labell);
let pageCount = createElement('label', { id: 'documenteditor_pagecount',
styles: 'margin-left:6px;letter-spacing: 1.05px;' });
updatePageCount();
statusBarDiv.appendChild(pageCount);
let editorPageCount = undefined;
let zoom: DropDownButton;
let zoomBtn: HTMLButtonElement = createElement('button', {
id: 'documenteditor-zoom',
// tslint:disable-next-line:max-line-length
className: 'e-de-statusbar-zoom'
}) as HTMLButtonElement;
statusBarDiv.appendChild(zoomBtn);
zoomBtn.setAttribute('title', 'Zoom level. Click or tap to open the Zoom
options. ');
let items: ItemModel[] = [
{
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 DropDownButton({ content: '100%', items: items, select: onZoom },
zoomBtn);
editablePageNumber.addEventListener('click',
updateDocumentEditorPageNumber);
editablePageNumber.addEventListener('keydown', onKeyDown);
editablePageNumber.addEventListener('blur', onBlur);
//Update page number on `viewChange` event
documenteditor.viewChange = (e): void => {
    updatePageNumberOnViewChange(e);
};
//Update page count on `contentChange` event.
documenteditor.contentChange = (): void => {
    //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();
        let pageNumber: number = 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();
  }
}
//Update zoom factor.
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() {
  let 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 EJ2 JavaScript 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.TS

```

import { DocumentEditor, Print } from '@syncfusion/ej2-documenteditor';
import { Button } from '@syncfusion/ej2-buttons';
DocumentEditor.Inject(Print);
let documenteditor: DocumentEditor = new DocumentEditor({
    enablePrint: true, height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
let sfdt: string = `{
    "sections": [
        {
            "blocks": [
                {
                    "inlines": [
                        {
                            "characterFormat": {
                                "bold": true,
                                "italic": true
                            },

```

```

        "text": "Hello World"
      }
    ]
  },
  "headersFooters": {
  }
}
]
} `;
documenteditor.open(sfdd);
let printButton: Button = new Button();
printButton.appendTo('#print');
document.getElementById('print').addEventListener('click', () => {
  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.TS


```
import { DocumentEditor, Print, Editor, Selection, EditorHistory } from
 '@syncfusion/ej2-documenteditor';
import { Button } from '@syncfusion/ej2-buttons';
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enablePrint: true,
    enableEditor: true,
    enableSelection: true,
    enableEditorHistory: true,
    height: '370px'
});
documenteditor.appendTo('#DocumentEditor');
let printButton: Button = new Button();
printButton.appendTo('#print');
document.getElementById('print').addEventListener('click', () => {
    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({
```

```
  readOnly: 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.TS

```
import { DocumentEditor, Print, Editor, Selection, EditorHistory, SfdtExport
} from '@syncfusion/ej2-documenteditor';
import { Button } from '@syncfusion/ej2-buttons';
//Inject require modules.
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory, SfdtExport);
let documenteditor1: DocumentEditor = new DocumentEditor({
  readOnly: false,
  enablePrint: true,
  enableEditor: true,
  enableSelection: true,
  enableEditorHistory: true,
  enableSfdtExport: true,
  height: '370px'
});
documenteditor1.appendTo('#DocumentEditor');
let printButton: Button = new Button();
printButton.appendTo('#print');
let documenteditor2: DocumentEditor = new DocumentEditor({
```

```

    enablePrint: true, enableSelection: true, isReadOnly: false,
    enableEditor: true, height: '370px'
  });
  documenteditor2.appendTo('#DocumentEditor2');
  //Print the document
  document.getElementById('print').addEventListener('click', () => {
    let sfddData = documenteditor1.serialize();
    documenteditor2.open(sfddData);
    //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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor, Selection, FontDialog, SfddtExport, } from
'@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfddtExport, FontDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableFontDialog: true,
    enableSfddtExport: true,
    height: '370px'
});
let containerPanel: HTMLElement = document.getElementById('container');
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    // To open Font Dialog
    documenteditor.showDialog('Font');
});
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">
            <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.TS

```
import { DocumentEditor, Editor, Selection, ParagraphDialog, SfdtExport, }
from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, ParagraphDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableParagraphDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open paragraph dialog
    documenteditor.showDialog('Paragraph');
});
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">
      <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 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.TS

```
import { DocumentEditor, Editor, Selection, TableDialog, SfdtExport, } from
 '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, TableDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableTableDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLInputElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open table dialog
    documenteditor.showDialog('Table');
});
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">
            <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.TS

```
import { DocumentEditor, Editor, Selection, BookmarkDialog, SfddtExport, }
from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfddtExport, BookmarkDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableBookmarkDialog: true,
    enableSfddtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open bookmark dialog
    documenteditor.showDialog('Bookmark');
});
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">
      <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>

```

Hyperlink dialog

This dialog allows editing or inserting a hyperlink at cursor position.

To open this dialog, refer to the following example.

INDEX.TS

```
import { DocumentEditor, Editor, Selection, HyperlinkDialog, SfdtExport, }
from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, HyperlinkDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableHyperlinkDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open hyperlink dialog
    documenteditor.showDialog('Hyperlink');
});
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">
    <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.TS

```

import { DocumentEditor, Editor, Selection, TableOfContentsDialog,
SfdtExport } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, TableOfContentsDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditor: true,
  enableTableOfContentsDialog: true,
  enableSfdtExport: true,
  height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
  //To open table of contents dialog
  documenteditor.showDialog('TableOfContents');
});
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">
            <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>

```

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.TS

```

import { DocumentEditor, Editor, Selection, StyleDialog, StylesDialog,
SfdtExport, } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, StyleDialog,
StylesDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,

```

```

        enableSelection: true,
        enableEditor: true,
        enableStyleDialog: true,
        enableSfdtExport: true,
        enableStylesDialog: true,
        height: '370px'
    });
    let button: HTMLElement = document.getElementById('dialog');
    button.addEventListener('click', function () {
        //To open styles dialog
        documenteditor.showDialog('Styles');
    });
    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">
      <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.TS

```
import { DocumentEditor, Editor, Selection, StyleDialog, SfdtExport } from
 '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, StyleDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableStyleDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open style dialog
    documenteditor.showDialog('Style');
});
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">
            <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>

```

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.TS

```

import { DocumentEditor, Editor, Selection, ListDialog, SfddExport, } from
'@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfddExport, ListDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableListDialog: true,
    enableSfddExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');

```



```
button.addEventListener('click', function () {  
    //To open list dialog  
    documenteditor.showDialog('List');  
});  
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">  
            <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.TS

```

import { DocumentEditor, Editor, Selection, BordersAndShadingDialog,
SfdtExport, } from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport,
BordersAndShadingDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
  isReadOnly: false,
  enableSelection: true,
  enableEditor: true,
  enableBordersAndShadingDialog: true,
  enableSfdtExport: true,
  height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
  //To open hyperlink dialog
  documenteditor.showDialog('BordersAndShading');
});
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>

```

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.TS

```

import { DocumentEditor, Editor, Selection, TableOptionsDialog, SfddtExport,
} from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfddtExport, TableOptionsDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableTableOptionsDialog: true,
    enableSfddtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open table options dialog
    documenteditor.showDialog('TableOptions');
});
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>
```

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.TS

```
import { DocumentEditor, Editor, Selection, TableOptionsDialog,
TablePropertiesDialog, BordersAndShadingDialog, SfdtExport, } from
'@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, TablePropertiesDialog,
BordersAndShadingDialog, TableOptionsDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableTableOptionsDialog: true,
    enableTablePropertiesDialog: true,
    enableBordersAndShadingDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open table properties dialog
    documenteditor.showDialog('TableProperties');
});
documenteditor.appendTo('#DocumentEditor');
// Insert a new table.
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.TS

```
import { DocumentEditor, Editor, Selection, PageSetupDialog, SfdtExport }
from '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfdtExport, PageSetupDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enablePageSetupDialog: true,
    enableSfdtExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open page setup dialog
    documenteditor.showDialog('PageSetup');
});
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">
      <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>

```

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.TS

```
import { DocumentEditor, Editor, Selection, ColumnsDialog, SfddExport } from
 '@syncfusion/ej2-documenteditor';
DocumentEditor.Inject(Editor, Selection, SfddExport, ColumnsDialog);
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableSelection: true,
    enableEditor: true,
    enableColumnsDialog: true,
    enableSfddExport: true,
    height: '370px'
});
let button: HTMLElement = document.getElementById('dialog');
button.addEventListener('click', function () {
    //To open page setup dialog
    documenteditor.showDialog('Columns');
});
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">
    <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 EJ2 JavaScript Document editor control

Document Editor provides RTL (right-to-left) support. This can be enabled using the “enableRtl” property.

INDEX.TS

```

import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
import { L10n } from '@syncfusion/ej2-base';
//Set locale object
L10n.load({
  'ar-AE': {
    'documenteditor': {
      'Table': 'الجدول',
      'Row': 'الصف',
      'Cell': 'الخلية',
      'Ok': 'موافق',
      'Cancel': 'إلغاء الأمر',
      'Size': 'حجم',
      'Preferred Width': 'العرض المفضل',
      'Points': 'نقاط',
      'Percent': 'المائة',
      'Measure in': 'القياس في',
      'Alignment': 'محاذاة',
      'Left': 'يسار',
      'Center': 'مركز',
      'Right': 'الحق',
      'Justify': 'تبرير',
      'Indent from left': 'مسافة بادئة من اليسار',
      'Borders and Shading': 'الحدود والتظليل',
      'Options': 'خيارات',
      'Specify height': 'تحديد الارتفاع',
      'At least': 'الاقبل',

```

```

'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 رأس',
'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': '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 الطبيعية',
'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': 'فقرة القائمة',
'Normal': 'العادية',
'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
    + 'علي سبيل المثال ، "الفصل 1". سيتم عرض الترقيم مثل ' +
'<br>' + 'الفصل الثاني- البند' + '<br>' + 'الفصل الأول- البند'
    + '<br>' + 'الفصل نون-البند' + '<br>'
    // tslint:disable-next-line:max-line-length
    + '<br>' + 'تنسيق رقم متعدد الإعدادات' + '<br>' +
' [مستوي المستوي] % [بأدئه] + '<br>' + ' [بأدئه] + ' [لاحقه] +
' [المستوي] للاحقه ]'
    + '<br>' + 'سيتم عرض الترقيم "1.2". سيتم عرض الترقيم "1.1" + '<br>' + 'البند 1.1' + '<br>' + 'البند 1.2' + '<br>' + 'نون-البند 1.'

```

```

        'Format': 'تنسيق',
        'Create New Style': 'إنشاء نمط جديد',
        'Modify Style': 'تعديل النمط',
        'New': 'الجديد',
        'Bullets': 'الرصاص',
        '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': 'إعادة تعيين',
        'Match case': 'حاله المباراة',
        'Whole words': 'كلمات كامل',
        'Add': 'إضافه',
        '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'
    }
});
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false,
    enableRtl: true, locale: 'ar-AE', height: '370px', serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/'
});
//Enable all the built in modules.
documenteditor.enableAllModules();
documenteditor.appendTo('#DocumentEditor');
let sfdt: string = `{
    "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
                        }
                    }
                ]
            }
        ]
    }
];
}
};
//Open the sfdt content in Document Editor.
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 EJ2 JavaScript 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.TS

```

import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
//Initialize DocumentEditor Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px' });
documenteditor.appendTo('#DocumentEditor');
let sfdt: string =
`{"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":"Verda

```



```
na"}, {"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", "style  
Name": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}}, {"text": "  
"}, {"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": 1  
0, "fontFamilyBidi": "Times New Roman"}}, {"text": "The best-selling products of  
Northwind Traders  
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 1  
0, "fontFamilyBidi": "Times New Roman"}}, {"text": "Company as follows:  
"}]}, {"paragraphFormat": {"afterSpacing": 0, "styleName": "Normal", "listFormat":  
{}}, {"characterFormat": {}, "inlines": [], {"rows": [{"cells": [{"blocks": [{"parag  
raphFormat": {"rightIndent": 26.850000381469727, "styleName": "Normal", "listForm  
at": {}}, {"characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "f  
ontFamily": "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": "Singl  
e", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADDBFF", "  
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spa  
ce": 0}, "bottom": {"color": "#4472C4FF", "hasNoneStyle": false, "lineStyle": "Singl  
e", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#00000  
0", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "spa  
ce": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "Non  
e", "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", "li  
neWidth": 0, "shadow": false, "space": 0}}, {"shading": {"backgroundColor": "#4472C4F  
F", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth":  
13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422  
465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 0}, {  
"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, {"charact  
erFormat": {}, "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": "#8EADDBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth":  
0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADDBFF", "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": {"  
color": "#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", "fontSizeBidi": 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
dow": 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": "#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": "#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.
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":"Côte de
Blaye"}]]], "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":f
alse,"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},"diagona
lUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":
0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":f
alse,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{
"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0}}, "shading":{"backgroundColor":"#D9E2F3FF","foregroundC
olor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.860000610351
56,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan
":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"par
agraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"i
nlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeB
idi":10,"fontFamilyBidi":"Times New
Roman"},"text":"141.396"}]]], "cellFormat":{"borders":{"top":{"color":"#8EAAD
BFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":fals
e,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"S
ingle","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":"Si
ngle","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#00
0000","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,"spac
e":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None",
"lineWidth":0,"shadow":false,"space":0}}, "shading":{"backgroundColor":"#D9E2
F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidt
h":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.9584189
9993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":
2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"h
eightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF",
"hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"sp
ace":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single
","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","ha
sNoneStyle":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,"spa
ce":0},"vertical":{"color":"#8EAADBFF","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":"#8EAADBFF",""

```

```

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":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":"Thüringer Rostbratwurst"}]
},
"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":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,
"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},
{"columnIndex":1},
{"blocks":
[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},
"text":"80.368"}]
},
"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":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"3"}]}],"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"},"columnIndex":0},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}}, "characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Raclette Courdavault"}]}],"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"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}}, "characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"71.155"}]}],"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":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":"4"}]}]}],"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":"#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":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Tarte au sucre"}]}]}],"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":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0}]]]

```



```

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":"#FFFFFF","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"}], "characterFormat":{},"bookmarkType":1,"name": "_GoB
ack"}]}], "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":fal
se,"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},"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":"#FFFFFF","foregroundColor"
:"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":"#8EAADBFF","hasNoneStyle":false
,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"l
ineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"colo
r":"#8EAADBFF","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":"#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,"gridBeforeWi
dthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Poin
t"}}, {"cells": [{"blocks": [{"paragraphFormat":{"styleName":"Normal","listForm
at":{},"characterFormat":{},"inlines": [{"characterFormat":{},"bookmarkType"
:0,"name": "_GoBack"}], "characterFormat":{"fontSize":10,"fontFamily":"Verdana"
,"fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"5"}]}], "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":"#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": "#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.
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": "Camembert
Pierrot
"}]}], "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": "#D9E2F3FF", "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": {"inlines
": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi":
10, "fontFamilyBidi": "Times New
Roman"}, "text": "46.825"}]}], "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": 14.3999999618530273, "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", "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","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":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"6"}]}],"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":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":"#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":48.860000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"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":48.860000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1}]}]}]}]

```

```

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": "#FFFFFF
FFF", "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", "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, "spa
ce": 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": "7"}]}], "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.
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": "Manjimup
Dried
Apples"}]}], "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", "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,"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":"41.819"}]}]},"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":"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.

```

```

202000076293945,"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":"#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},"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":"#FFFFFFF","foregroundColor":"empty",
"texturesStyle":"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",
"fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"32.698"}]}],
"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":"#FFFFF
FF","foregroundColor":"empty",
"texturesStyle":"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","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":"Singl
e","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":"#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"},
{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},
"text":"Carnarvon Tigers"}]]],
"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":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},
{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},
"text":"29.171"}]]],
"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,"he

```

```

ightType": "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": "10" } ] } ] }, "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": 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": "Rössle Sauerkraut." } ] } ] }, "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": 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": "#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": "#FFFFFF
FFF", "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", "h
asNoneStyle": 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"}], "grid": [64.71214527422465, 292.87942351880633, 117.95
841899993776], "tableFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneS
tyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0},
"left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWi
dth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyl
e": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bot
tom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWid
th": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNone
Style": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "dia
gonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWid
th": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#8EADBFF", "hasNoneSt
yle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "v
ertical": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lin
eWidth": 0.5, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFF
FF", "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,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]},"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]},"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"chartLegend":{"position":"Right","chartTitleArea":{"fontName":"+mn-1t","fontSize":9,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"808080","rgb":"#808080"}}}},"chartTitleArea":{"fontName":"+mn-1t","fontSize":14,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"000000","rgb":"#000000"}}},"chartArea":{"foreColor":"#FFFFFFF","plotArea":{"foreColor":"#000000F","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","rgb":"#4472c4"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"ed7d31","rgb":"#ed7d31"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"a5a5a5","rgb":"#a5a5a5"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"ffc000","rgb":"#ffc000"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"5b9bd5","rgb":"#5b9bd5"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"70ad47","rgb":"#70ad47"},"line":{"color":"ffffff","rgb":"#ffffff"}},{fill":{"foreColor":"264379","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":{"color":"ffffff","rgb":"#ffffff"}}],"seriesName":"Sales"}],"chartPrimaryCategoryAxis":{"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":"TickLabelPosition_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","minorTickMark":"TickMark_None","tickLabelPosition":"TickLabelPosition_NextToAxis"},"chartTitle":"Best Selling Products","chartType":"Pie","gapWidth":0,"overlap":0,"height":225,"width":432]]],{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[]},"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]}}

```



```

idi:"Verdana"}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "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."}], "headersFooters": {}, "characterFormat": {"bold": false, "italic": false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignmment": "Normal", "highlightColor": "NoColor", "fontColor": "#000000", "fontSizeBidi": 11, "fontFamilyBidi": "Calibri"}, "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing": 8, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listFormat": {}, "bidi": false, "defaultTabWidth": 36, "enforcement": false, "hashValue": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection", "styles": [{"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", "listFormat": {}}, "characterFormat": {"bold": true, "fontSize": 16, "fontFamily": "Arial", "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial"}, "basedOn": "Normal", "link": "Heading 1 Char", "next": "Normal"}, {"name": "Heading 1 Char", "type": "Character", "characterFormat": {"bold": true, "fontSize": 16, "fontFamily": "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": "Default 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": []}`;
//Open the document in Document Editor.
documenteditor.open(sfddt);

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Animation</title>
  <meta charset="utf-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>
<!-- TypeScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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.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('#DocumentEditor');
container.documentEditor.currentUser = 'engineer@mycompany.com';
```

```

let sfdt: string =
`{"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":{"allowAutoFit":
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,"boldBid
i":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, "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": [{"editRangeId": "808933422", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.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, "preferredWidthType": "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, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 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, "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}}}]}, {"title": null, "description": null, "tableFormat": {"allowAutoFit": 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": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "Single", "lineWidth": 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, "space": 0.0, "hasNoneStyle": false}}, {"characterFormat": {"bold": true, "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, "fontSizeBidi": 14.0}}, {"name": "_GoBack", "bookmarkType": 0}, {"name": "_GoBack", "bookmarkType": 1}]}]}, {"rows": [{"rowFormat": {"allowBreakAcrossPages": true, "isHeader": false, "height": 20.0, "heightType": "AtLeast", "borders": {"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", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 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, "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": [{"editRangeId": "810441411", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com"}], {"text": "Enter email address"}, {"editRangeId": "810441411", "editableRangeStart": {"editRangeId": "810441411", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com"}}]}]}]]]

```



```
"cellFormat": {"columnSpan": 1, "rowSpan": 1, "preferredWidth": 467.5, "preferredWidthType": "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, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 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, "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}}}, {"title": null, "description": null, "tableFormat": {"allowAutoFit": 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": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "Single", "lineWidth": 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, "space": 0.0, "hasNoneStyle": false}}, {"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": {"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", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 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, "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"}}, {"editRangeId": "1016946268", "columnFirst": 0, "columnLast": 0, "user": "manager@mycompany.com"}, {"text": "Enter the feedbacks"}], {"editRangeId": "1016946268", "editableRangeStart": {"editRangeId": "1016946268", "columnFirst": 0, "columnLast": 0, "user": "manager@mycompany.com"}}]}]}, {"cellFormat": {"columnSpan": 1, "rowSpan": 1, "preferredWidth": 467.5, "preferredWidthType": "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, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 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, "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}}
```



```

style":false}}}}}}], "title":null, "description":null, "tableFormat":{"allowAut
toFit":true, "leftIndent":0.0, "tableAlignment":"Left", "preferredWidthType":"A
uto", "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":"Sin
gle", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "botto
m":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNone
Style":false}, "vertical":{"lineStyle":"Single", "lineWidth":0.5, "shadow":fals
e, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"Single", "line
Width":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, "spa
ce":0.0, "hasNoneStyle":false}}, "bidi":false}}, {"characterFormat":{"bold":tru
e, "fontSize":14.0, "boldBidi":true, "fontSizeBidi":14.0}, "paragraphFormat":{"l
ineSpacing":32.0, "lineSpacingType":"Exactly", "styleName":"Normal"}, "inlines
":[{"text":"Review
comments", "characterFormat":{"bold":true, "fontSize":14.0, "boldBidi":true, "f
ontSizeBidi":14.0}}}], {"rows":[{"rowFormat":{"allowBreakAcrossPages":true, "i
sHeader":false, "height":20.0, "heightType":"AtLeast", "borders":{"left":{"line
Style":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":fals
e}, "right":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "h
asNoneStyle":false}, "top":{"lineStyle":"None", "lineWidth":0.0, "shadow":false
, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineStyle":"None", "lineWidth":
0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical":{"lineStyle"
:"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "ho
rizontal":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "ha
sNoneStyle":false}, "diagonalDown":{"lineStyle":"None", "lineWidth":0.0, "shado
w":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":[{"editRangeId
":"1373703080", "columnFirst":0, "columnLast":0, "user":"manager@mycompany.com
"}, {"text":"Enter the
comments"}, {"editRangeId":"1373703080", "editableRangeStart":{"editRangeId":
"1373703080", "columnFirst":0, "columnLast":0, "user":"manager@mycompany.com"}
}]}], "cellFormat":{"columnSpan":1, "rowSpan":1, "preferredWidth":467.5, "preferre
dWidthType":"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, "sha
dow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"None", "line
Width":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineS
tyle":"None", "lineWidth":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, "shad
ow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"lineStyle":"Non
e", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagona
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, "space": 0.0, "hasNoneStyle": false}, "bidi": false}, {"paragraphFormat": {"styleName": "Normal"}, "inlines": []}, {"headersFooters": {"header": {"blocks": [{"paragraphFormat": {"styleName": "Header"}, "inlines": [{"text": "Employee's Details"}]}]}, "sectionFormat": {"headerDistance": 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, "bidi": false}, {"characterFormat": {"fontSize": 11.0, "fontFamily": "Calibri", "fontSizeBidi": 11.0, "fontFamilyBidi": "Calibri"}, "paragraphFormat": {"afterSpacing": 8.0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple"}, "background": {"color": "#FFFFFF"}, "styles": [{"type": "Paragraph", "name": "Normal", "next": "Normal"}, {"type": "Character", "name": "Default Paragraph Font"}, {"type": "Paragraph", "name": "List Paragraph", "basedOn": "Normal", "paragraphFormat": {"leftIndent": 36.0, "contextualSpacing": true}}, {"type": "Paragraph", "name": "Header", "basedOn": "Normal", "next": "Normal", "link": "Header Char", "paragraphFormat": {"afterSpacing": 0.0, "lineSpacing": 1.0, "lineSpacingType": "Multiple", "tabs": [{"tabJustification": "Center", "position": 234.0, "tabLeader": "None", "deletePosition": 0.0}, {"tabJustification": "Right", "position": 468.0, "tabLeader": "None", "deletePosition": 0.0}]}, {"type": "Character", "name": "Header Char", "basedOn": "Default Paragraph Font"}, {"type": "Paragraph", "name": "Footer", "basedOn": "Normal", "link": "Footer Char", "paragraphFormat": {"afterSpacing": 0.0, "lineSpacing": 1.0, "lineSpacingType": "Multiple", "tabs": [{"tabJustification": "Center", "position": 234.0, "tabLeader": "None", "deletePosition": 0.0}, {"tabJustification": "Right", "position": 468.0, "tabLeader": "None", "deletePosition": 0.0}]}, {"type": "Character", "name": "Footer Char", "basedOn": "Default Paragraph Font"}], "defaultTabWidth": 36.0, "formatting": false, "protectionType": "ReadOnly", "enforcement": true, "hashValue": "TQGuJuLceQCe234Ygx4q6NFgHpRMfilhjFTojyKzbQOkwk+ckEM9CjNidkiUhSR/e/7sfMxO4sbPcg/DBzztMg==", "saltValue": "FXbkrlRtDIIIZfwlM7ldMg=="}`;
//Open the document in Document Editor.
container.documentEditor.open(sfddt);

```

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>
<!-- TypeScript 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 EJ2 JavaScript 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;
```

```
`
```

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;
```

```
,
```

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;
```

```
,
```

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:

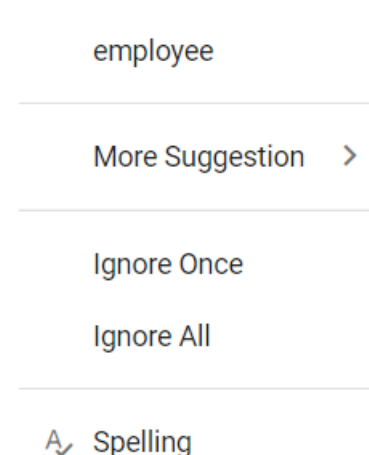
1. The rules are framed automatically using the root word, the possible words and affix file.
2. If you pass null for the parameters `affPath` and `possibleWords`, then it will add a single root word to dictionary.
3. This API is included starting from v20.2.0.xx.

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

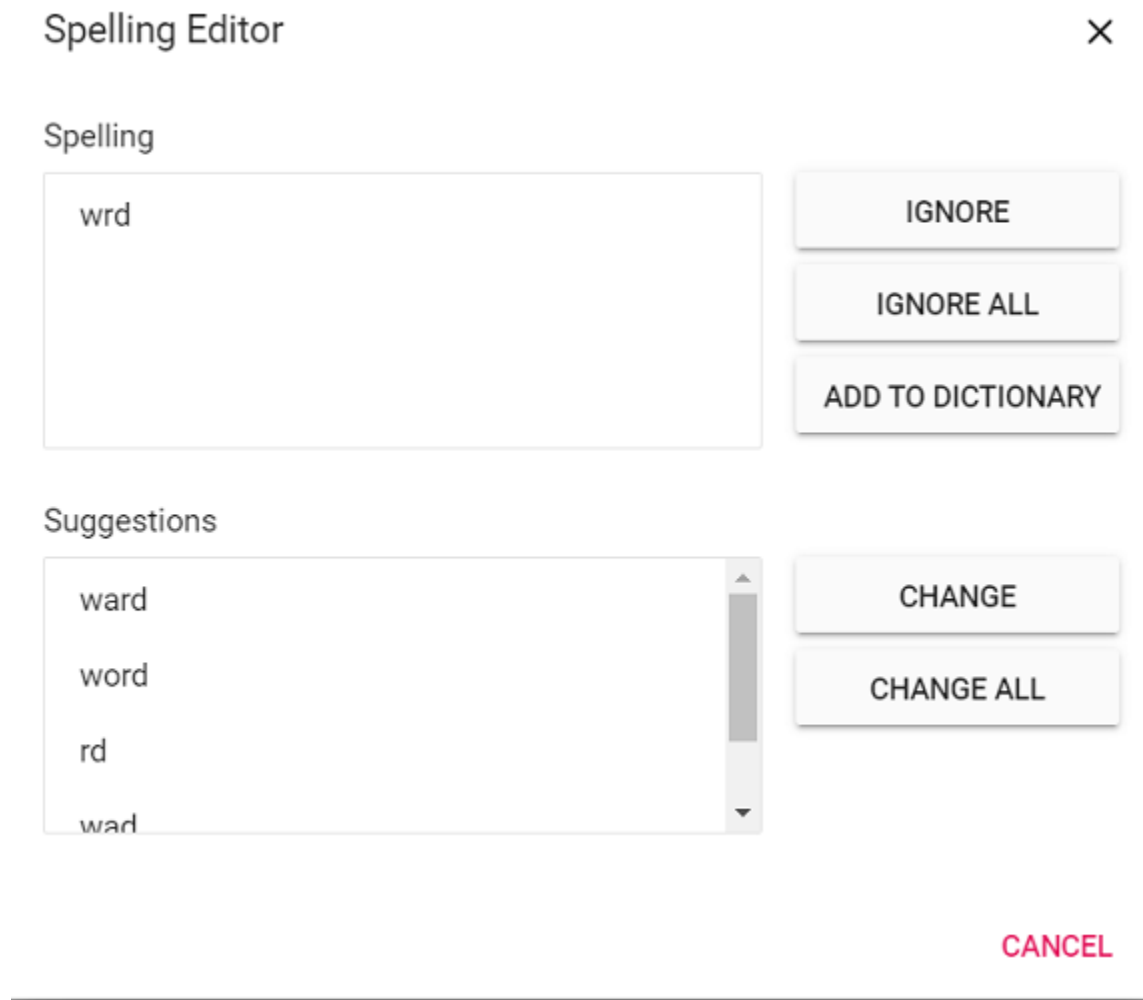
If you do not wish to add the word to dictionary and do not want to show error, use Ignore Once or Ignore All options.

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 EJ2 JavaScript 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 EJ2 JavaScript 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.
```

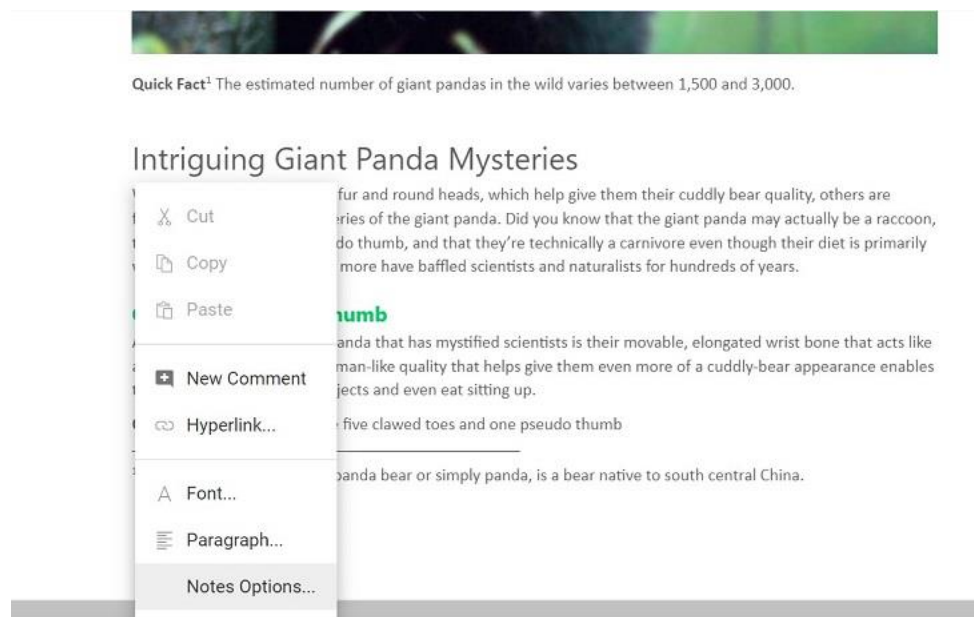
```
DocumentEditorContainer.Inject(Toolbar);
```

```
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 EJ2 JavaScript 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>
    </body>
</html>
```



```

    </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 EJ2 JavaScript 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: 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;
}
};
`

```

[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: 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;
}
};
`
```

The following is the output of custom context menu with customization.

INDEX.TS

```
import { DocumentEditor, Editor, Selection, OptionsPane, Search,
ContextMenu, EditorHistory, ImageResizer, ListDialog, TableDialog,
HyperlinkDialog, ParagraphDialog, FontDialog, PageSetupDialog,
BookmarkDialog, StyleDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, TableOfContentsDialog } from
'@syncfusion/ej2-documenteditor';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
let defaultCheckBoxObj: CheckBox = new CheckBox({ label: 'Hide Default
Context Menu', change: contextmenuHelper });
defaultCheckBoxObj.appendTo('#default-context-menu');
let positionCheckBoxObj: CheckBox = new CheckBox({ label: 'Add Custom option
at bottom', change: contextmenuHelper });
positionCheckBoxObj.appendTo('#position-context-menu');
//Initialize Document Editor component.
let documentEditor: DocumentEditor = new DocumentEditor({
isReadOnly: false, height: '370px', serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/'
```

```

});
//Enable all built in modules.
documentEditor.enableAllModules();
//Render Document Editor component.
documentEditor.appendTo('#DocumentEditor');
//Creating custom menu items
let menuItems: MenuItemModel[] = [
    {
        text: 'Search In Google',
        id: 'search_in_google',
        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 + 'search_in_google':
            let searchContent: string = documentEditor.selection.text;
            if (!documentEditor.selection.isEmpty &&
/\S/.test(searchContent)) {
                window.open('http://google.com/search?q=' + searchContent);
            }
            break;
    }
};
//function to handle the CheckBox change event
function contextmenuHelper(args: ChangeEventArgs): void {
    documentEditor.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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor } from '@syncfusion/ej2-documenteditor';
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false, height: '370px', serviceUrl:
    'https://services.syncfusion.com/js/production/api/documenteditor/'
});
//Enable all built in modules.
documenteditor.enableAllModules();
//Open the default document in `created` event.
documenteditor.created = function () {
    //load your default document here
    let data: string =
    '{"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"}, {"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 Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3 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 4 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 5 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 6 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "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 7 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml1": []}';

```

```
//Open the default document
documenteditor.open(data);
};
//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">

    <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.TS

```
import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';
//Inject require modules.
DocumentEditorContainer.Inject(Toolbar);
//Initialize Document Editor component.
let container: DocumentEditorContainer = new DocumentEditorContainer({
    enableToolbar: true, height: '590px'
});
//Open the default document in `created` event.
container.created = function () {
    //load your default document here
    let data: string =
'{"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, "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": [], "customXml
1": [] }';
    //Open the default document
    container.documentEditor.open(data);
};
//Render Document Editor Container component.
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">

        <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>

```

Read only default in EJ2 JavaScript 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.TS

```

import { DocumentEditor } from '@syncfusion/ej2-documenteditor';
//Initialize Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({ height: '370px',
serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/' });
//Enable all the built in modules.

```

```

documenteditor.enableAllModules();
//Open the default document inside `created` event.
documenteditor.created = function () {
    //Load your default document here
    let data: string =
'{"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":"Cal

```

```

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": [], "customXml
1": []}';

    //Open the default document
    documenteditor.open(data);
};

//Enable read only mode inside `documentChange` event.
documenteditor.documentChange = (): void => {
    documenteditor.isReadOnly = true;
};

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="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.TS

```

import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';
DocumentEditorContainer.Inject(Toolbar);
//Initiliaze Document Editor Container component.
let documenteditorContainer: DocumentEditorContainer = new
DocumentEditorContainer({ enableToolbar: true, height: '500px' });
documenteditorContainer.serviceUrl =
'https://services.syncfusion.com/js/production/api/documenteditor/';
//Enable read only mode inside `documentChange` event.
documenteditorContainer.documentChange = (): void => {
  documenteditorContainer.restrictEditing = true;
};
//Render Document Editor container component.
documenteditorContainer.appendTo('#DocumentEditor');
let data: string =
'{"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":{"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": []}';
//Open the default document
documenteditorContainer.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>

```

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 EJ2 JavaScript 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]FileInfo 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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: CustomContextMenuEventArgs): 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);

`
```

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 EJ2 JavaScript 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');
```

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 EJ2 JavaScript 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.TS

```

import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
popups';
DocumentEditorContainer.Inject(Toolbar);
let container: DocumentEditorContainer = new DocumentEditorContainer({
    enableToolbar: true, height: "400"
});
createSpinner({
    // Specify the target for the spinner to show
    target: document.getElementById('container')
});
document.getElementById('import').addEventListener('click', function () {
    // load your default document here
    let data: string =
'{"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":{"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": []}';

// showSpinner() will make the spinner visible
showSpinner(document.getElementById('DocumentEditor'));
// Open the default document
container.documentEditor.open(data);
setInterval(function () {
    // hideSpinner() method used hide spinner
    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>
<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 EJ2 JavaScript 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 EJ2 JavaScript 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.DocIORenderer.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 DocIORenderer for Word to PDF conversion
```

```
DocIORenderer render = new DocIORenderer();
```

```
//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 EJ2 JavaScript 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'],
```

```
}
```

```
});
```

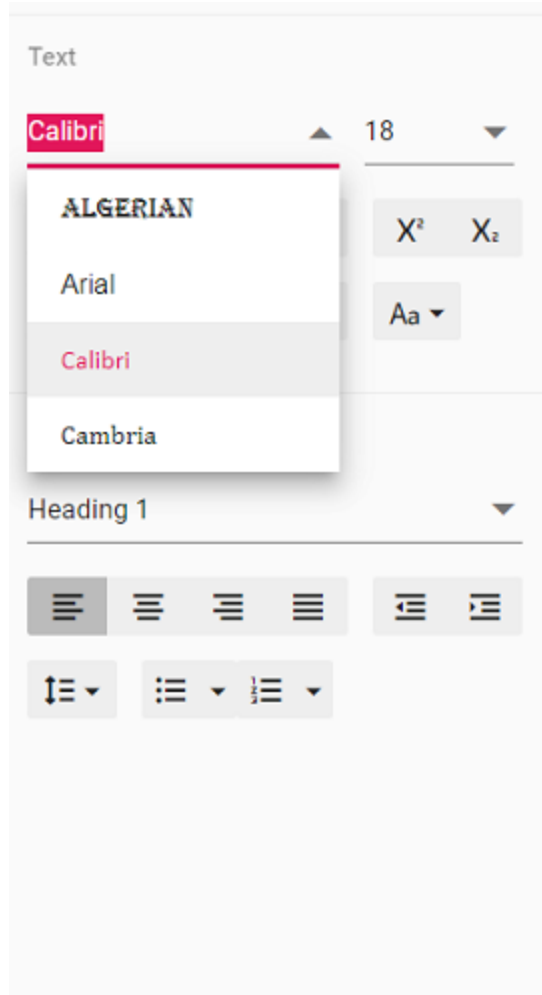
```
DocumentEditorContainer.Inject(Toolbar);
```

```
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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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 EJ2 JavaScript 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;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;
};
`ts

```

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 });
`ts

```

[Disable header and footer edit in document editor in EJ2 JavaScript 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');
`ts

```



```
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 EJ2 JavaScript 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.

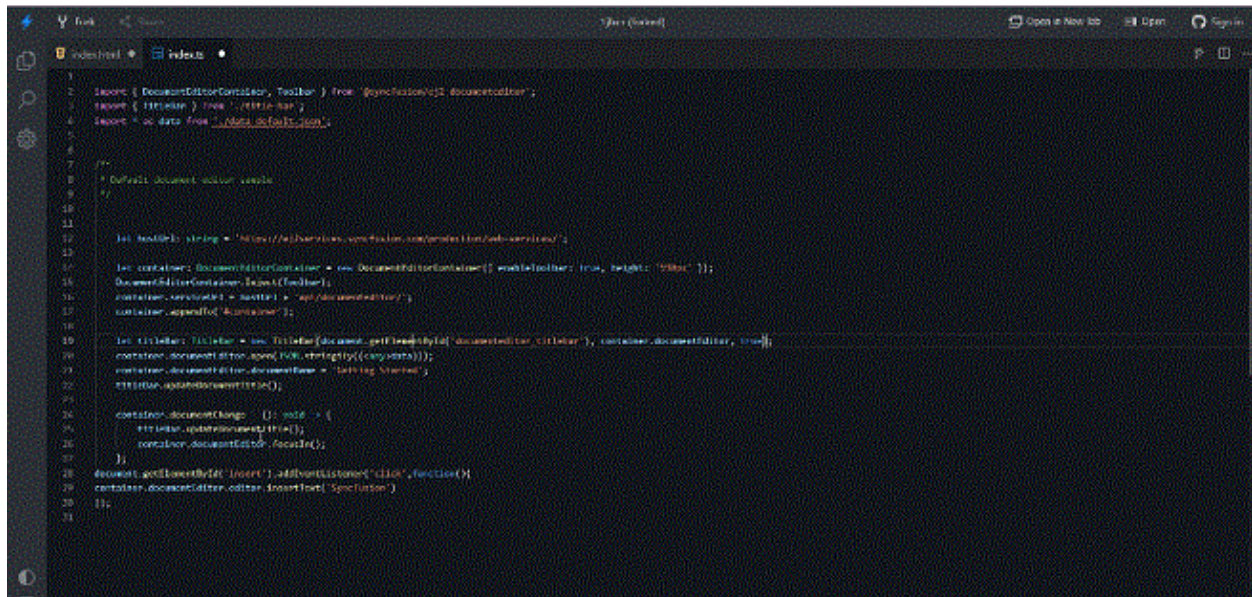
```
`ts
let hostUrl: string = 'https://ej2services.syncfusion.com/production/web-services/';
let container: DocumentEditorContainer = new DocumentEditorContainer({ enableToolbar: true, height: '590px' });
DocumentEditorContainer.Inject(Toolbar);`
```

```

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 EJ2 JavaScript 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`.

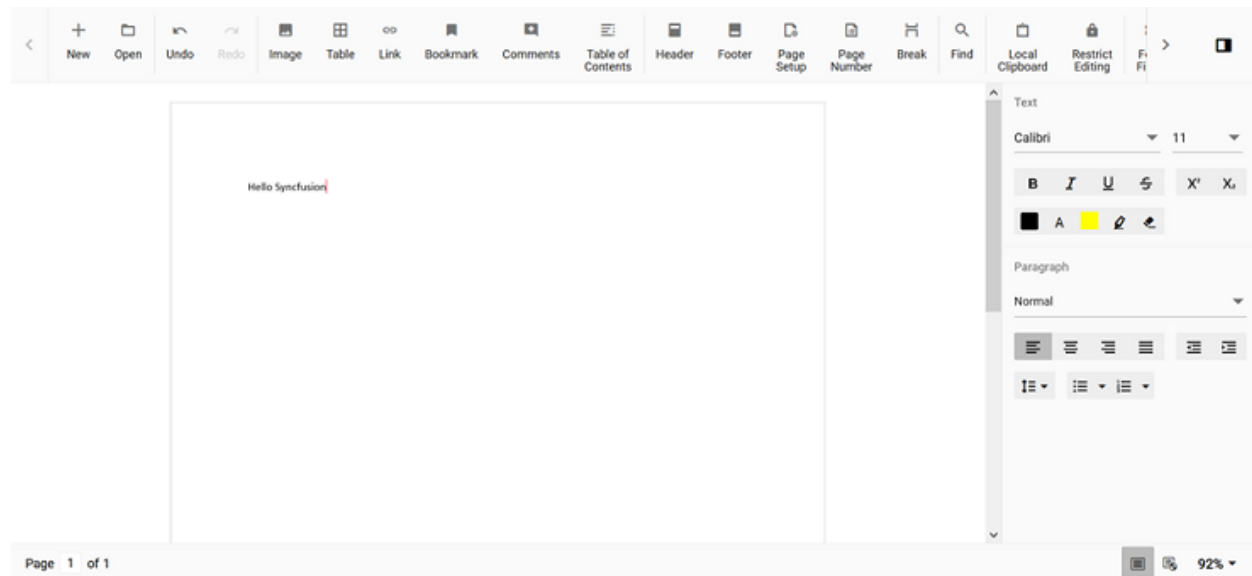
Please refer the below code snippet to change the cursor color to red.

```

\
.e-de-blink-cursor {
border-left: 1px solid red!important;
}
\

```

Output will be like below:



[Hide tool bar and properties pane in EJ2 JavaScript 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 exiting 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 EJ2 JavaScript 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
AAUAAAAFCAYAAACNbybIAAAAHIEQVQI12P4
//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
AAUAAAAFCAYAAACNbyblAAAAHElEQVQI12P4
//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
```

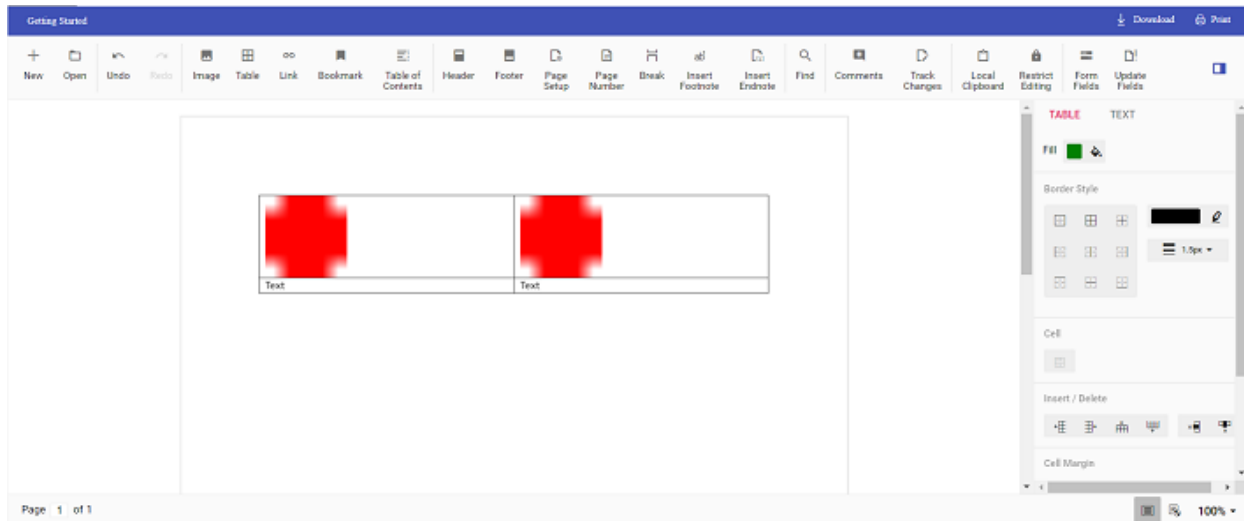


```

container.documentEditor.selection.select(startOffset, startOffset);
}
`

```

The output will be like below.



Change the default search highlight color in EJ2 JavaScript 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.

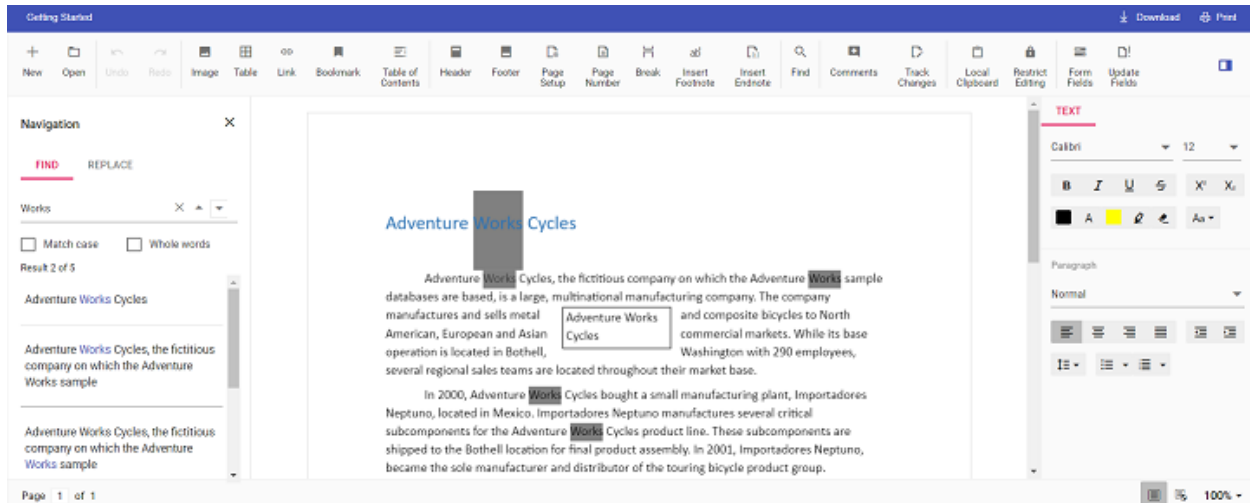
The following example code illustrates how to change the default search highlight color.

```

`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:



Optimize sfdt in EJ2 JavaScript Document editor control

Starting from version v21.1.x, the SFDT file generated in Word Processor component is optimized by default to reduce the file size. All static keys are minified, and the final JSON string is compressed. This helps reduce the SFDT file size relative to a DOCX file and provides the following benefits,

- File transfer between client and server through the internet gets faster.
- The new optimized SFDT files require less storage space than the old SFDT files.

Hence, the optimized SFDT file can't be directly manipulated as JSON string.

This feature comes with a public API to switch between the old and new optimized SFDT format, allowing backward compatibility.

As a backward compatibility to create older format SFDT files, refer the following code changes,

Client/Server	Old Code	New Code from v21.1.x
Client-side	<pre>var documenteditorContainer = new ej.documenteditor.DocumentEditorContainer();</pre>	<pre>var documenteditorContainer = new ej.documenteditor.DocumentEditorContainer({ documentEditorSettings: { optimizeSfdt: false } });</pre>
Server-side C#	<pre>WordDocument sfdtDocument = WordDocument.Load(stream, formatType);string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);</pre>	<pre>WordDocument sfdtDocument = WordDocument.Load(stream, formatType);sfdtDocument.OptimizeSfdt = false;string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);</pre>
Server-side Java	<pre>String sfdtDocument = WordProcessorHelper.load(stream, formatType);</pre>	<pre>String sfdtDocument = WordProcessorHelper.load(stream, formatType, false);</pre>

To convert from older format SFDT from a new optimized SFDT file, refer the following code example,

Client/Server	Code example
Client-side	<pre>var documenteditorContainer = new ej.documenteditor.DocumentEditorContainer({ documentEditorSettings: { optimizeSfdt: false } });</pre>
Server-side C#	<pre>using(Syncfusion.DocIO.DLS.WordDocument docIODocument = WordDocument.Save(optimizedSfdt)) { sfdtDocument = WordDocument.Load(docIODocument); sfdtDocument.OptimizeSfdt = false; string oldSfdt = JsonSerializer.Serialize(sfdtDocument);}</pre>
Server-side Java	<pre>WordDocument docIODocument = WordProcessorHelper.save(optimizedSfdt);String oldSfdt = WordProcessorHelper.load(docIODocument, false);</pre>

Disable auto focus in EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```
import { DocumentEditor, Editor } from '@syncfusion/ej2-documenteditor';
//Initialize Document Editor component.
let documenteditor: DocumentEditor = new DocumentEditor({
    isReadOnly: false, height: '370px', documentEditorSettings: {showRuler:
true}, serviceUrl:
'https://services.syncfusion.com/js/production/api/documenteditor/'
});
//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.TS

```

import { DocumentEditorContainer, Toolbar } from '@syncfusion/ej2-
documenteditor';
//Inject require modules.
DocumentEditorContainer.Inject(Toolbar);
//Initialize Document Editor Container component.
let container: DocumentEditorContainer = new DocumentEditorContainer({
    height: '590px', documentEditorSettings: {showRuler: true}
});
document.getElementById('container_ruler_button').addEventListener('click',
function () {
    container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
});
//Render Document Editor Container component.
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="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>

```

DropDown Menu

Icons in EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
// To initialize the DropDownButton with icon.
let drpDownBtn: DropDownButton = new DropDownButton({ iconCss: 'ddb-icons e-
message', items: items }, '#iconbutton');
// To position the icon to the top of the text on a DropDownButton.
let dropDownBtnObj: DropDownButton = new 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="iconpstn">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,AAEAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNfudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQb4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJcG9zdIwSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAPuAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAAABADnBf//AADnA///AAAAQA
EAAAAAQACAAAAAIAAAAAHAA6AFwAAAACAAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zsMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAQAKAAEAAQAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAAABAAKABwAAQAAAAAABQALACY
AAQAAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAAcWASAGcAAwABBAKAAAACAHkAAwABBAKAAQA
UAHsAAwABBAKAAgAOAI8AAwABBAKAAwAUAJ0AAwABBAKABAAUALEAAwABBAKABQAWAMUAwABBAK
ABgAUANsAAwABBAKACgBYAO8AAwABBAKACwAKAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWwRiLWljb25
zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZG1vd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkABwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABLAGQAIAB1AHMAAQBuAGcAIABTAHkAbgBjAGYAdQBzAGkABwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkABwBuAC4AYwBvAG0AAAAAAgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbWFpbAtyZWwF
kLXVucmVhZAZkZWxldGUAAAAA==) 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.TS

```
import { DropDownButton, ItemModel, DropDownButtonModel } from
'@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
  {
    text: 'New tab'
  },
  {
    text: 'New window'
  },
  {
    text: 'New incognito window'
  },
  {
    separator: true
  },
  {
    text: 'Print'
  },
  {
    text: 'Cast'
  },
  {
    text: 'Find'
  }
];
```

```
// Initialize DropDownButton options.
let options: DropDownButtonModel = {
  items: items,
  iconCss: 'e-icons e-menu',
  cssClass: 'e-caret-hide'
};
// To initialize the icon only DropDownButton.
let drpDownBtn: DropDownButton = new 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.TS

```

import { DropDownButton, ItemModel, DropDownButtonModel } from
'@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
  {
    text: 'Display Settings'
  },
  {
    text: 'System Settings'
  },
  {
    text: 'Additional Settings'
  }
];
// Initialize DropDownButton options.
let options: DropDownButtonModel = {
  items: items,
  cssClass: 'e-caret-hide',
  iconCss: 'e-image'
};
// To initialize the DropDownButton with sprite image.
let drpDownBtn: DropDownButton = new 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="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
<link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
<link href="https://cdn.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.TS

```
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
// To initialize the vertical DropDownButton.
let drpDownBtn: DropDownButton = new DropDownButton({ iconCss: 'ddb-icons e-
message', cssClass: 'e-vertical', 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">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,AAEAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNfudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACmlheHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJcG9zdIwkSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAIAAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABADnBf//AADnA///AAAAQA
EAAAAAQACAMAAAAAAAAAAHAA6AFwAAAACAAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAAAgAHAAsAAQAAAAAAAAAwAKABIAAQAAAAAAAABAABAAQAAAAAABQALACY
AAQAAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGcAAwABBAkAAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLW1jb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0Zvb2Npb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUACgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIAB1AHMAaQBuaGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtZW50bWpAtYzWf
kLXVucmVhZAZkZWxldGUAAAAA==) 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 EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
  {
    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 initialize the DropDownButton.
let drpDownBtn: DropDownButton = new 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,AAEAAAAKAIAAAwAgTlMvMjltSfYAAAEoAAAAVmNtYXdnG0dnAAABmAAAAD5nbHlm/RE
9ZwAAAaegAAAJ8aGVhZBOPuxsAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YbG9jYOHIAO4AAAHYAAAAADmlheHABFACZAAABCAAAACBuYW11lLBM9QAABGQAAAI9cG9zdOdmKCA
```



```
AAAakAAAAZgABAAAEAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAafI9ISF8
PPPUACwQAAAAAANG+uxUAAAAA2D67FQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAGAI0ABAAAAAA
AAgAAAAoACgAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBAQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAMAAAAUAMAAQAAABQABAAqAAAAABAAEAAEAAOce//8AAOc
A//8AAAAABAAQAAAAABAAIAAwAEAAUAAAAAAAAALgBaAHYAIAE+AAAAAwAAAAAD9AP0AAIABgAZAAA
3JSc3FwEnNwcXPwM1LwcPAgwBJOo76QHT6qlu6XIFBAICBAWmCAkJCgkJCQw66jrpAdLpqW7pcgg
JCgkKCQimBwQDAQEDBAAAAAAEAAAAANNA/QAAwAHABAAGAAAAAREjESMRixEnETMVITUzESE3IxU
hNSM1IQLih4SFhUICGED9ZoWFAPqF/nACp/3qAhb96gIWQv1mQ0MC3YVCQkMAAAAAAGAAAAAD8wN
uAAYACgAANyERIwKBiWUXASEMA+gC/g3+DgEBGNwBufyOkqLC/hcB6Z+5AXIAAAACAAAAAAPQA/Q
ABQAKAAA3IREjCQELBwkCMAOGA/4x/jIBA8sBlgGX/moMAl7+PgHC2LL+nAFkAWQAAAAACAAAAAP
0A8UAAwCMAAA3MxEjAQ8DFRcPDBeZNx8ENxc/Cj0BLwU/Cy8INzU/CDUvBTU/DTUvCQclPwQ1Lws
jDwEMra0B+QIKBAEBAQEYIREREhMiCQkoEAYhBzUHHjmT2w4FCAsNCwkFAwQCAgQJBgIBAQEDDgQ
JCAYHawMBAQEBAwMDCQIBAQMWcWUEBAMDAGICBAQKAQEBAoHBWYFBQQDAwEBAQEEBQcJBQUFBhH
+rQ8JBAMCAQEDAwOMFQMHBGwLDQCwHGwGHAd4BBQMDdh8KBCw6HRscGi8JCBsM/ooBAR8DAQEBAGe
BAwYKCGwGCAgIBQgJCAsFBAQEBOGMAwICAwIBwGHBgYGBQUJBAIGAgQMCQYFBgcJCQoJCAgHCwQ
CBQMCBAQEBOUGBwcIBWYGBgYKCGgGAgIBAQEBRjEZGhsNDQwNCyIeMQQEAgQBAQIAAAASAN4AAQA
AAAAAAAAABAAAAAQAAAAAAQAJAAEAAQAAAAAAAGAHAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABO
AAQAAAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACGAsADCAAQAAAAAACwASAGMAAwABBAKAAAA
CAHUAwABBAKAAQASAHCAAwABBAKAAgAOAIKAawABBAKAAwASAJcAAwABBAKABAASAKKAawABBAK
ABQAWALsAAwABBAKABgASANEAAwABBAKACgBYAOMAawABBAKACwAkATsgZGRiLWljB25zUmVndWx
hcmRkYi1pY29uc2RkYi1pY29uc1Zlc2NpY24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXN
pbmcgU3luY2Zlc2l2b2lBNXZRYbyBTdHVkaW93d3cuc3luY2Zlc2l2b2l5b20AIABkAGQAYgAtAGk
AYwBvAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AAQBJAG8AbgBzAGQAZABiAC0AAQBJAG8AbgB
zAFYAZQByAHMAAQBVAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABLAGQAIAAB1AHMAAQBuAGcAIABTAKhAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAGA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGAQIBAwEEAQUBBgEHAAdlZG10XzAzCWRlbGV
0ZV8wMgxtZXNzYWdlLW1haWwLcmVhZC11bnJlYWQJbGlrZS0tLTAAxAAAAA==)
format('truetype');
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.TS

```
import { DropDownButton, ItemModel, DropDownButtonModel, MenuEventArgs } from
 '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
  {
    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'
  }
];
let menuOptions: DropDownButtonModel = {
  items: items,
  iconCss: 'e-cart-icon e-shopping',
  beforeItemRender: (args: MenuEventArgs) => {
    args.element.getElementsByTagName('a')[0].setAttribute('target',
'_blank');
  }
};
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new 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,AAEAAAAKAIAAAwAgTlMvMj0gSQ4AAAEoAAAAVmNtYXdDnEOdVAAABiAAAADZnbHlmGat
ngwAAAcgAAADYAGVhZBKtP4wAAADQAAAAANmhoZWEHmQNpAAAArAAAACRobXR4B+j//gAAAYAAAAA
IbG9jYQBsAAAAAHAAAAABmlheHABDwBQAAABCAAAACBuYW1lfiv2lQAAAqAAAAIBcG9zdIZzcJA
AAASkAAAAOgABAAADUv9qAFoEAP/+//wD7AABAAAAAAAAAAAAAAAAAAAAAgABAAAAAQAA2UwSaF8
PPPUACwPoAAAAANfSfWUAAAAA19J9Zf/+AAAD7APdAAAAACAACAAAAAAAAAAAAEAAAACAEQAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPdAJYAAAABAAAAAAAAABAAAAAPo//4
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA

```

```

AAAAAbAAAAAP//gAAA+wD3QAJABIAQwAAJR4BMjY0JicOAQeATI2NCYiBgEOAwclIgYXEx4BMwU
yFgcOAQclIgYXBhYXBT4BPwE2PwI2NxM+AycmIyIGAeABJzonJx0gJ/6jASc6Jyc6JwMXIDgZPyX
9giQkBkIIoyQBACQgCQo/JP7RIxcBARcjAVgkQg0QDgkICgkNkw4xMBwCBScJE1UdJyc6JwEBJx0
dJyc6JycDZQg0PigBAy0k/uAkMAQnHRsmAQMTDA8QAQMBLCi1IhYWFxYhAYciNg4YDBMCAAAAEgD
eAAEAAAAAAAAAAAAEAAAAAAAAEABAAABAEAAAAAAAAIABwAFAAEAAAAAAAAAMABAAMAEAAAAAAAAQ
ABAAQAAEAAAAAAAAUACwAUAAEAAAAAAAAAYABAAfAAEAAAAAAAAoALAAjAAEAAAAAAAAsAEgBPAAQAQQ
JAAAAAgBhAAMAAQQJAAEACABjAAMAAQQJAAIADgBrAAMAAQQJAAMACAB5AAMAAQQJAAQACACBAAM
AAQQJAAUAFgCJAAMAAQQJAAAYACACfAAMAAQQJAAoAWACnAAMAAQQJAAAsAJAD/IGNhcnRSZWdlbGF
yY2FydGNhcnRWZXJzaW9uIDEuMGNhcnRGb250IGdlbmVyYXRlZCB1c2luZyBTew5jZnVzaW9uIE1
ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGMAYQByAHQAUgBlAGcAdQBsAGEAcgBjAGE
AcgB0AGMAYQByAHQAVgBlAHIAcWBPAG8AbgAgADEALgAwAGMAYQByAHQARgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcWBPAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcWBPAG8AbgAuAGMAbwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMAEHNob3BwaW5nLWNhcnQtMDUAAAA
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.TS

```

import { DropDownButton, ItemModel, DropDownButtonModel, MenuEventArgs }
from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';

```

```

enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Edit'
    },
    {
        text: 'Cut'
    }
];
let ddbOption: DropDownButtonModel = {
    iconCss: 'e-ddb-icons e-paste',
    cssClass: 'e-vertical',
    items: items,
    iconPosition: 'Top',
    // To handle li templating
    beforeItemRender: (args: MenuEventArgs) => {
        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';
            let span: Element = args.element.children[0];
            span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
            let div: Element = 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';
            let span: Element = args.element.children[0];
            span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
            let div: Element = args.element.children[1];
            div.setAttribute('class', 'e-div-align');
        }
    }
};
//To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton(ddbOption,
'#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="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
base/styles/material.css" rel="stylesheet">
    <link href="https://cdn.syncfusion.com/ej2/24.1.41/ej2-
buttons/styles/material.css" rel="stylesheet">
    <link href="https://cdn.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,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAAVmNtYXNDicOK6AAABjAAAADhnbHlmGcE
PFQAAAcwAAAMwaGVhZA69CA8AAADQAAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQDYAZgAAAEHAAAAACG1heHABEGDAAAABCAAAACBuYw1lY1dlQAAAPwAAAKFcG9zdPjWcMo
AAAEeAAAAASAABAAAEAAAAAFwEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAgmhm8l8
PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAACAACAAAAAAAAAAAAEAAAADALQABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQQAAXAQAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAAAAAAAAgAAAAAUAAMAAQAAABQABAakAAAABAAEAAEAJOJh//8AAOJg//8AAAABAAQAAAA
BAAIAAAAAANgBmAFAAAAAAAXA/QABAALAC0ATgCzAAABISchJzcVHwc/By8HDwYBFSE1MxEhESU
HFQ8GLwc/Bx8GJysBDw4RHw4zITM/DhEvDisBLw4rAQ8NAUQBdlxAfr0BAwQGBwgICgkJCacGBAM
BAQMEBgcICQkKCAgHbGqD/qYB1l79jQFoAQMEBgcHCQkJCQgGBgQDAQEEDBAYGCAkJCQkHbWYEA6y
9CGkICQgHBwcGBQQAawMBAQEBAwMDBQUGBwcHCAkICQoCeAoJCAkIBwcHBgUEBAMDAQEBAQMDAwU
FBgcHBwgJCAkKvQqEBgUHBwcICQkJCgoKCwsLCwoKCgkJCQgHBwcFBgQBBYVRuh0FBQkIBwUFAgE
BAgUFBwgJCGkJCAcGBAMBAQMEBgcICQEifX39LwLRMwQFCAGHBQUCAQECEBQUHCAgJCQkIBwUEAwE
BAwQFBwgJIGICAwQFBQYGBwgICAKJcf0pCQkJCAgIBwYGBQUEAwICAgIDBAUFBgYHCAgICQkJAtc
JCQkICAgHBgYFBQQDAgIKCQkICAgHBgYFBAQDAgICAgMEBAUGBgcICAgJCQAFAAAAAAXA/QABwA
PABcAOACdAAABHwIjPwIDMzcZfZMDIYcVITUzESERJQCVDwYvBz8HHwYnKwEPDhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWCO90B1l79jQFoAQMEBgcHCQkJCQgGBgQDAQEEDBAYGCAk
JCQkHbWYEA6y9CGkICQgHBwcGBQQAawMBAQEBAwMDBQUGBwcHCAkICQoCeAoJCAkIBwcHBgUEBAM
DAQEBAQMDAwUFBgcHBwgJCAkKvQqEBgUHBwcICQkJCgoKCwsLCwoKCgkJCQgHBwcFBgQCFREigG4
SM/6wd3cBe/t9ff0vAtEzBAUICAcFBQIBAQIFBQcICAKJCQgHBQQDAQEEDBAUHCAkiAgIDBAUFBgY
HCAgICQkJ/SkJCQkICAgHBgYFBQQDAgICAgMEBQUGBgcICAgJCQkC1wkJCQgICAcGBgUFBAMCAgo
JCQgICAcGBgUEBAMCAgICAwQEBQYGBwgICAKJAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAA8
AAQABAAAAAAACAACAEAAABAAAAAADAA8AFwABAAAAAAAEAA8AJgABAAAAAAAFAA8ANQABAAAAAA
GAA8AQABAAAAAAAKACwATwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4AjwADAAE

```

```
ECQACAA4ArQADAAEECQADAB4AuWADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQAGAB4BDQA
DAAEECQAKAFgBKwADAAEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWd1bGFyQ29udGV4dE11bnU
gKDIpQ29udGV4dE11bnUgKDIpVmVyc2lubiAxLjBDb250ZXh0TWVudSAoMilGb250IGd1bmVyYXR
lZCB1c2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgaEMAbwB
uAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAFIAZQBnAHUAbABhAHIAQwBvAG4AdABlAHgAdABNAGU
AbgB1ACAAKAAyACkAQwBvAG4AdABlAHgAdABNAGUAbgB1ACAAKAAyACkAVgB1AHIAcWBPAG8AbgA
gADEALgAwAEMAbwBuAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAEYAbwBuAHQAIAbnAGUAbgB1AHIA
YQYB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAD01UX1Bhc3RlU3B1Y2lhbAxNVF9QYXN
0ZVRleHQAAA==) format('trueype');
    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,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgaEMAbwB
3NQAAAdwAAAjMAgVhZBKO9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
Ubg9jYQGOAegAAAHQAAADG1heHABEWBlAAABCAAAACBuYw1l1LBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAwAAUABAAADUv9qAFoEAAAAAAAAADygABAAAAAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAAACAACAAAAAAAAAAAAAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAwGPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAGADAAQAAAAAAI4AwgEAAASYAAwAA//oDNQPsAA4AHQBYAAALHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMHYBhgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEffXo6IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfikwVhcMBwYlFRYkBWcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOZsaKQ4ONZcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLoZGdBA/5vAZEDQQAEEAAAAA0qA+kABQANABCAHwAAARUzFSErAYERiZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4+++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41afbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAA
AAAABAAAAQAAAAAAQAJAAEAAQAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLW1jb25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLW1jb25zRm9udCBnZW51cmF0ZWQgdXNpbmc
gU3luY2Z1c2lubiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwBlAGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIAblAHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
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.TS

```
import { DropDownButton, DropDownButtonModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize DropDownButton options.
let menuOptions: DropDownButtonModel = {
    target: '#target',
    iconCss: 'e-icons e-table',
    iconPosition: 'Top',
    cssClass: 'e-vertical'
};
// To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new 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>
        </tbody></table>
    </div>
    <button id="action">Table</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-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.TS

```

import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
  {
    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-iconse-paragraph'
  }
];
// To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({ 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,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAAVmNtYXDicOK6AAABjAAAADhnbHlmGcE
PFQAAAcwAAAMWagVhZA69CA8AAADQAAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQDYAZgAAAHEAAAACG1heHABEGDAAAABCAAAACBuYW11xY1dlQAABPwAAAKFcG9zdPjWcMo
AAAEeAAAAASAABAAAEAAAAAFwEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAwABAAAAQAAGmh8l8
PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAAACAACAAAAAAAAAAAAEAAAADALQABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAMAAAUAAMAAQAAABQABAakAAAABAAEAAEAAOJh//8AAOJg//8AAAABAAQAAAA
BAAIAAAAAANgBmAAFAAAAAAAXA/QABAALAC0ATgCzAAABIScHJzcVHwc/By8HDwYBFSE1MxEhESU
HFQ8GLwc/Bx8GJysBDw4RHw4zITM/DhEvDisBLw4rAQ8NAUQBd1xAfr0BAwQGBwgICgkJCAcGBAM
BAQMEBgcICQkKCAgHBgQD/qYB1179jQFoAQMEBgcHCQkJCQgGBgQDAQEDBAYGCAkJCQkHBwYEA6y
9CgkICQgHBwCGBQQAawMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAMDAQEBAQMDAwU
FBgCHBwgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCFBgQBByVRuh0FBQkIBwUFAgE
BAgUFBwgJCAkJCAcGBAMBAQMEBgcICQEifX39LwLRMwQFCAGHBQUCAQECEBQUHCAgJCQkIBwUEAwE
BAwQFBwgJIgICAwQFBQYGBwgICAKJCf0pCQkJCAGIBwYGBQUEAwICAgIDBAUFBgYHCAgICQkJAtc
JCQkICAGHBgYFBQQDAgIKCQkICAGHBgYFBAQDAgICAgMEBAUGBgICAgJCQAFAAAAAAXA/QABwA
PABCAOACdAAABHwIjPwIDMzcZfZMDIYcVITUzESERJQCVDwYvBz8HHwYnKwEPDhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWC090B1179jQFoAQMEBgcHCQkJCQgGBgQDAQEDBAYGCAk
JCQkHBwYEA6y9CgkICQgHBwCGBQQAawMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAM
DAQEBAQMDAwUFBgCHBwgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCFBgQCFREigG4

```

```
SM/6wd3cBe/t9ff0vAtEzBAUICAcFBQIBAQIFBQcICAkJCQgHBQQDAQEDBAUHCakiAgIDBAUFBgY
HCAGICQkJ/SkJCQkICAGHBgYFBQQDAgICAgMEBQUGBgcICAgJCQkC1wkJCQgICAcGBgUFBAMCAgo
JCQgICAcGBgUEBAMCAgICAwQEBQYGBwgICAkJAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAA8
AAQABAAAAAACAAcAEAABAAAAAADAA8AFwABAAAAAAEAA8AJgABAAAAAFAAAsANQABAAAAAA
GAA8AQABAAAAAAKACwAtwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4AjwADAAE
ECQACAA4ArQADAAEECQADAB4AuWADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQAGAB4BDQA
DAAEECQAKAFgBKwADAAEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWdlbGFyQ29udGV4dE1lbnU
gKDIpQ29udGV4dE1lbnUgKDIpVmVyc2lvbiAxLjBDb250ZXh0TWVudSAoMilGb250IGdlbmVyYXR
lZCB1c2luZyBTew5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAEMabwB
uAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAFIAZQBnAHUAbABhAHIAQwBvAG4AdABlAHgAdABNAGU
AbgBlACAkAAyACkAQwBvAG4AdABlAHgAdABNAGUAbgBlACAkAAyACkAVgBlAHIAcWBPAG8AbgA
gADEALgAwAEMabwBuAHQAZQB4AHQATQBlAG4AdQAgACgAMgApAEYAbwBuAHQAIAbnAGUAbgBlAHIA
AYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAD01UX1Bhc3RlU3B1Y2lhbAxNVF9QYXN
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,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNdE+dkAAABlAAAAADxnbHlmlh3
3NQAAAdwAAAjMAgVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEWBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAACAACAAAAAEEEEAAFAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAAPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAaaaaAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAALHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMHYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXhGE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDrgkOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfIkWVhcMBwYlFRYkBWcMf1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsakQ4ONzcniiYXNBgYNBcmiiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEEEAAAOqA+kABQANABcAHwAAARUzFSErAYERIZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fhW+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAC
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBs
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41AfBMAZgBJwmYCHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAA
AAAAAABAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWlj25zUmVndWxhcmlR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWlj25zRm9udCBnZW51cmF0ZWQgdXNpbmc
gU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABkAGQAYgAtAGkAYwB
```

```

vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIAB1AHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaGAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxBGZvbnQ
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](#)

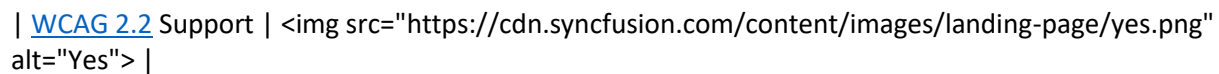
Accessibility in EJ2 JavaScript Drop down button control

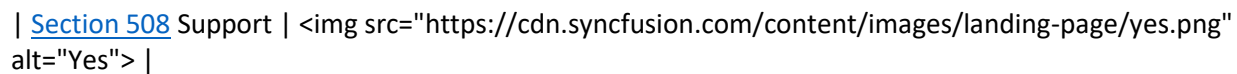
The Drop down button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

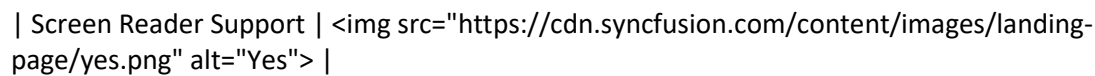
The accessibility compliance for the Drop down button component is outlined below.

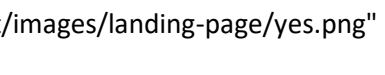
| Accessibility Criteria | Compatibility |

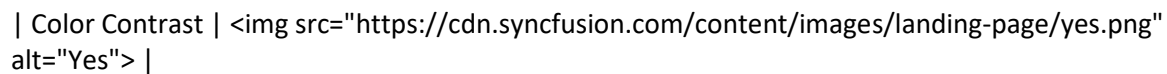
| -- | -- |

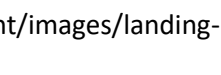
| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

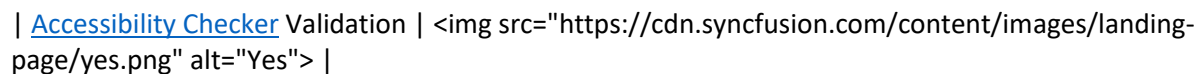
| Screen Reader Support |  |

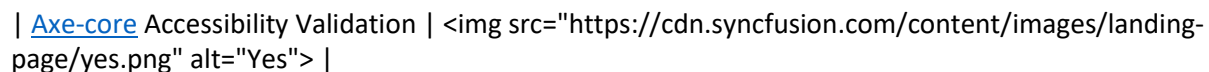
| Right-To-Left Support |  |

| 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 EJ2 JavaScript components](#)

How To

Change caret icon in EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel, BeforeOpenCloseMenuEventArgs,
DropDownButtonModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// Initialize DropDownButton options.
let options: DropDownButtonModel = {
    items: items,
    // Removing 'e-caret-up' class.
    beforeClose: (args: BeforeOpenCloseMenuEventArgs) => {
        drpDownBtn.cssClass = '';
    },
    // Adding 'e-caret-up' class.
    beforeOpen: (args: BeforeOpenCloseMenuEventArgs) => {
        drpDownBtn.cssClass = 'e-caret-up';
    }
};
// To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton(options);
// Render initialized DropDownButton.
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 EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new 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>
<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 EJ2 JavaScript 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.TS

```

import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
//To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new 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,AAEAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNlFudgAAABkAAAApnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYl1lH00mDAAAAPAAAAJJcG9zdIwKsr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAABAABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQPUAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQAjgAAAAQABAABADnBf//AADnA//AAAAQA
EAAAAQACAAMAAAAAAAAAHAA6AFwAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4

```

```

NAQEY3QG4/I+IASL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAAEEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAAQAKAAEAQAQAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAAABAAKABwAAQAAAAAABQALACY
AAQAAAAAABgAKADEAAQAAAAAACgAsADsAAQAAAAAACwASAGcAAwABBAKAAAACAHkAAwABBAKAAQA
UAHsAAwABBAKAAgAOAI8AAwABBAKAAwAUAJ0AAwABBAKABAAUALEAAwABBAKABQAWAMUAAwABBAK
ABgAUANsAAwABBAKACgBYAO8AAwABBAKACwAkAUcgZS1kYilpY29uc1JlZ3VsYXJlLWLiLWljb25
zZS1kYilpY29uc1ZlcnNpb24gMS4wZS1kYilpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAUADAQZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABLAGQAIAB1AHMAaQBwAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgB
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';
}
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
button {
  margin: 25px 5px 20px 20px;
}

```

Customize icon and width in EJ2 JavaScript 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.TS

```

import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);

```

```
// Initialize action items.
let items: ItemModel[] = [
  {
    text: 'Find'
  },
  {
    text: 'Replace'
  },
  {
    text: 'Go To'
  },
  {
    text: 'Go To Special'
  }
];
// To initialize DropDownButton with `e-custom` class.
let drpDownBtn: DropDownButton = new 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 EJ2 JavaScript 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.TS

```

import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Edit'
    },
    {
        text: 'Delete'
    },
    {
        text: 'Mark as Read'
    },
    {
        text: 'Like Message'
    }
];
//To initialize the DropDownButton component.

```



```
let drpDownBtn: DropDownButton = new 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,AAEAAAoAKIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNlFudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQb4ADoAAAHMMAAACm1heHABEAAAYAAABCAAAACBuYW1lH00mDAAAAPAAAAJJcG9zdIwkSr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAAAAAAABAAAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAAEAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAA
```

```
D6AAAA+gAAAAAAAAIAAADAAAFAADAAEAAAAUAAQAJgAAAAQABAABAdNbF//AAdnA///AAAAQA  
EAAAAAQACAAMAAAAAHAA6AFwAAAACAAAAAAPqA2UABGAKAAA3IREjCQEjBRcBIQID6AL+Dv4  
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE  
EyWGWAZb+agICX/4+AcLSxv6CAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECP+zCI  
BMGEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA  
AAAAAAQAKAEAAQAAAAAGAhAAASAQAAAAAwAkABIQAaaaaaaBAakAbWAQAAAAABQALACY  
AAQAAAAABGaKADEAAQAAAAAACgAsAdSAQAAAAACWASAgCaAWABBakaAAACHkaAwABBakaAAQA  
UAHsaAwABBakaAgAOAI8AAwABBakaAwAUaj0AAwABBakaABAauALEAAwABBakaBQAWAMuaawABBaka  
ABgaUANsaAwABBakaCGBYAo8AAwABBakaCWkAuUcgZSlkyilpy29uc1JlZ3VsYXJlLWRiLWljbj25  
zZSlkyilpy29uc1Zlcnpb24gMS4wZS1kyilpy29uc0ZvbncGZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmddXNpb24uY29tACAAZQAtAGQAYgAtAGKaYwBVAG4  
mdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmddXNpb24uY29tACAAZQAtAGQAYgAtAGKaYwBVAG4Aacwb  
ACWSAGUAZWBlAGWAYQByAGUALQBKAgiAlQBPAgmabwBuAHMAZQAtAGQAYgAtAGKaYwBVAG4Aacwb  
WAGUAacgbZAgaKBwBUACAAMQAuADAazQatAGQAYgAtAGKaYwBVAG4AacwbGAG8AbGB0ACAAZwBlAG4  
AzQByAGEadABLAGQAIABLAHMaaQBuAGcaIBTAhkAbgBJAGyAdQBzagKabwBuACAATQBLaHQAcgB  
vACAAUwBOAHUAZABPaG8AdWB3AHcalGbZAhhkgBgBJAGyAdQBzagKabwBuAC4AywBVAG0AAAAAGa  
AAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAWEAAQUADG1lc3NhZ2UtbnWFfbmAtcyZFf  
klXVVumVhZAZzkZWldGUAAAAA==) 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;
```

Group popup items with listview component in EJ2 JavaScript 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.TS

```

import { DropDownButton, DropDownButtonModel, ItemModel, MenuEventArgs }
from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
import { ListView } from '@syncfusion/ej2-lists';
enableRipple(true);
// Initialize DropDownButton options.
let ddbOption: DropDownButtonModel = {
    target: '#listview',
    iconCss: 'e-icons e-down',
    cssClass: 'e-caret-hide'
};
// To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton(ddbOption);
// Render initialized DropDownButton.
drpDownBtn.appendTo('#element');
// Initialize datasource for ListView component.
let dataSource: { [key: string]: Object }[] = [
    { 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' }
];
// Initialize ListView component
let listViewInstance: ListView = new ListView({
    dataSource: dataSource,
    // Map the appropriate columns to fields property
    fields: { text: 'text', groupBy: 'category' },
    // To show CheckBox.
    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 EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// To initialize the DropDownButton component without down arrow.
let drpDownBtn: DropDownButton = new DropDownButton({ items: items,
cssClass: 'e-caret-hide' });
// Render initialized DropDownButton.
drpDownBtn.appendTo('#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: #008c8f;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
```

Open a dialog on popup item click in EJ2 JavaScript 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.TS

```
import { DropDownButton, ItemModel, MenuEventArgs, DropDownButtonModel }
from '@syncfusion/ej2-splitbuttons';
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// To initialize the Dialog component.
let dialog: Dialog = new 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',
    height: '150px',
    visible: false,
    position: {X: 100, Y: 100}
});
// Render initialized Dialog.
dialog.appendTo('#dialog');
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Archive'
    },
    {
```

```

        text: 'Inbox'
    },
    {
        text: 'HR Portal'
    },
    {
        separator: true
    },
    {
        text: 'Other Folder...'
    },
    {
        text: 'Copy to Folder'
    }
    ]];
// Initialize DropDownButton options.
let ddbOption: DropDownButtonModel = {
    iconCss: 'ddb-icons e-folder',
    cssClass: 'e-vertical',
    items: items,
    iconPosition: 'Top',
    select: (args: MenuEventArgs) => {
        if (args.item.text === 'Other Folder...') {
            dialog.show();
        }
    }
};
// To initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton(ddbOption,
'#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="https://cdn.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,AAEAAAAKAIAAAwAgTlMvMj0jSRoAAAEoAAAAVmNtYXNfudgAAABkAAAAADpnbHlmSrK
TCAAAAdgAAAC4aGVhZBktK8cAAADQAAAAANmhoZWEHmQNtAAAArAAAACRobXR4D7gAAAAAYAAAAA
QbG9jYQB4ADoAAAHMAAAACm1heHABEAAAYAAABCAAAACBuYWI1hH00mDAAAAPAAAAJJcG9zdIwksr0
AAATcAAAATQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAABABAAAAQAAGc/PS18
PPPUACwPoAAAAANfSc3wAAAAA19JzfAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAEAAAwAAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wPnBQNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAATIAAADAAAAFAADAAEAAAAUAAQAJgAAAAQABAABAADnBf//AADnA///AAAAQA
EAAAAAQACAMAAAAAAAAHAA6AFwAAAAACAAAAAPqA2UABgAKAAA3IREjCQEjBRcBIQID6AL+Dv4
NAQEY3QG4/I+IAsL+GAHonroBcwAAAAIAAAAAA8YD6gAFAAoAADchESMJASUHCQImA6AD/jL+MQE
EywGWAZb+agICX/4+AcLXsv6cAWQBZAAAAAEAAAAA+oD6gALAAATCQEXCQE3CQEnCQECATP+zcI
BMgEzwf7OATLB/s3+zgMp/s3+zsIBM/7NwgEyATPB/s4BMgAAAAASAN4AAQAAAAAAAAABAAAAQA
AAAAAQAKAAEAAQAAAAAAAgAHAAsAAQAAAAAAAwAKABIAAQAAAAABAAKABWAAQAAAAABQALACY
AAQAAAAABgAKADEAAQAAAAACgAsADsAAQAAAAACwASAGcAAwABBAkAAAACAHkAAwABBAkAAQA
UAHsAAwABBAkAAgAOAI8AAwABBAkAAwAUAJ0AAwABBAkABAAUALEAAwABBAkABQAWAMUAwABBAk
ABgAUANsAAwABBAkACgBYAO8AAwABBAkACwAkAUcgZS1kYi1pY29uc1JlZ3VsYXJlLWRiLWlj25
zZS1kYi1pY29uc1ZlcnNpb24gMS4wZS1kYi1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGQAYgAtAGkAYwBvAG4
AcwBSAGUAZwB1AGwAYQByAGUALQBkAGIALQBpAGMABwBuAHMAZQAtAGQAYgAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4
AZQByAGEAdABlAGQAIAB1AHMAAQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgB
vACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADG1lc3NhZ2UtbfWfPbAtyZWf
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 EJ2 JavaScript 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.TS

```

import { DropDownButton, ItemModel, OpenCloseMenuEventArgs,
BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
//Initialize action items.
let items: ItemModel[] = [
  {
    text: 'Cut'
  },
  {
    text: 'Copy'
  },
  {
    text: 'Paste'
  }
];
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton({
  items: items,
  cssClass: 'e-caret-up',
  // To position dropDownButton popup.
  open: (args: OpenCloseMenuEventArgs) => {
    args.element.parentElement.style.top =
drpDownBtn.element.getBoundingClientRect().top -
args.element.parentElement.offsetHeight + 'px';
  }
});
// 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%;
}
button {
  margin: 25% 5px 20px 30%;
}
.e-caret-up .e-caret::before {
  content: '\e918';
}
```

Underline a character in the item text in EJ2 JavaScript 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.TS

```
import { DropDownButton, DropDownButtonModel, ItemModel, MenuEventArgs }
from '@syncfusion/ej2-splitbuttons';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
// Initialize action items.
let items: ItemModel[] = [
    {
        text: 'Cut'
    },
    {
        text: 'Copy'
    },
    {
        text: 'Paste'
    }
];
// Initialize DropDownButton options.
let ddbOption: DropDownButtonModel = {
    items: items,
    beforeItemRender: (args: MenuEventArgs) => {
        if (args.item.text === 'Copy') {
            //To underline a particular text.
            args.element.innerHTML = '<u>C</u>opy';
        }
    }
};
// Initialize the DropDownButton component.
let drpDownBtn: DropDownButton = new DropDownButton(ddbOption);
// 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%;
}
```

DropDownList

Tags in EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
    placeholder:"Select a vegetable"
});
// render initialized DropDownList
dropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
    placeholder: "Select a vegetable"
});
// render initialized DropDownList
dropDownListObject.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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// define the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
let dropDownListObject: DropDownList = new 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');
```

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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
let dropDownListObject: DropDownList = new 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');

```

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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let countriesData: { [key: string]: Object }[] = [
  { 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' } }
    ];
    //initiate the DropDownList
    let dropDownListObject: DropDownList = new DropDownList({
        // bind the sports Data to datasource property
        dataSource: countriesData,
        // maps the appropriate column to fields property
        fields: { text: 'Country.Name', value: 'Code.Id' },
        //set the placeholder to DropDownList input
        placeholder: "Select a country"
    });
    //render the component
    dropDownListObject.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>

```

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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let customers: DropDownList = new DropDownList({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    //set the placeholder to DropDownList input
    placeholder: "Select a customer",
    //sort the resulted items
    sortOrder: 'Ascending'
});
//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>
<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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
  //bind the data manager instance to dataSource property
  dataSource: new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
  }),
  //bind the Query instance to query property
  query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
```

```
//map the appropriate columns to fields property
fields: { text: 'FirstName', value: 'EmployeeID' },
//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
DropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID' },
    //set the placeholder to DropDownList input
    placeholder: "Select an employee",
    //sort the resulted items
    sortOrder: 'Ascending',
    //set the template value to itemTemplate property
    itemTemplate: "<span><span>${FirstName}</span><span class
='city'>${City}</span></span>",
    //set the value to valueTemplate property
    valueTemplate: "<span>${FirstName} - ${City}</span>"
});
//render the component
DropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, Predicate, DataManager, ODataV4Adaptor } from
 '@syncfusion/ej2-data';
// form predicate to fetch the grouped data
let groupPredicate = new Predicate('City',
 'equal','london').or('City','equal','seattle');
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
 'City','EmployeeID']).take(5)
    .where(groupPredicate),
    //map the appropriate columns to fields property
    fields: { text: 'FirstName', value: 'EmployeeID', 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
DropDownListObject.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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6),

```



```
//map the appropriate columns to fields property
fields: { text: 'FirstName', value: 'EmployeeID' },
//set the placeholder to DropDownList input
placeholder:"Select an employee",
//sort the resulted items
sortOrder: 'Ascending',
//set the value to header template
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
DropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
let sportsData = [ "Basketball", "Cricket", "Football", "Golf"];
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: sportsData ,
    //set the placeholder to DropDownList input
    placeholder:"Select a game",
    //set the value to footer template
    footerTemplate:"<span class='foot'> Total list items: "+
sportsData.length + "</span>"
});
//render the component
DropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//initiates the component
let DropDownListObject: DropDownList = new 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
DropDownListObject.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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the data manager instance to dataSource property
    dataSource: new DataManager({
        // Here, use the wrong url to display the action failure
        template
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),
    //bind the Query instance to query property
    query: new 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>"
});
//render the component
DropDownListObject.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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the data with category
let vegetableData: { [key: string]: Object }[] = [
    { 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
let vegetables: DropDownList = new 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>
```

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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
  placeholder:"Select a vegetable"
});
// render initialized DropDownList
dropDownListObject.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>

```

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 EJ2 JavaScript 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.TS

```

import { DropDownList, FilteringEventArgs } from '@syncfusion/ej2-
dropdowns';
import { DataManager, Query } from '@syncfusion/ej2-data';
let searchData: { [key: string]: Object; }[] = [
{ Index: "s1", Country: "Alaska" }, { Index: "s2", Country: "California" },
{ Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia" }
];
let filter: DropDownList = new DropDownList({
    dataSource: searchData,
    // map the appropriate column
    fields: { text: "Country", value: "Index" },

```



```

// set placeholder to DropDownList input element
placeholder:"Select a country",
// set true to allowFiltering for enable filtering supports
allowFiltering: true,
//Bind the filter event
filtering: function (e: FilteringEventArgs) {
    let query = new Query();
    //frame the query based on search string with filter type.
    query = (e.text != "") ? query.where("Country", "startswith",
e.text, true) : query;
    //pass the filter data source, filter query to updateData method.
    e.updateData(searchData, query);
}
});
filter.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.TS

```
import { DropDownList, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: DropDownList = new DropDownList({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to DropDownList input element
    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: FilteringEventArgs) => {
        // 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; }
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, true) : query;
            e.updateData(searchData, query);
        }
    }
});
filter.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.TS

```

import { DropDownList, FilteringEventArgs } from '@syncfusion/ej2-
dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: DropDownList = new DropDownList({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to DropDownList input element
    placeholder: "Select a name",
    // set true to allowFiltering for enable filtering supports
    allowFiltering: true,
    //set the height of the popup element

```

```

popupHeight: "250px",
//sort the resulted items
sortOrder: 'Ascending',
//bind the filtering event handler
filtering: (e: FilteringEventArgs) => {
    // load overall data when search key empty.
    if(e.text == '') e.updateData(searchData);
    else{
        let query: Query = new Query().select(['ContactName',
'CustomerID']);
        // change the type of filtering
        query = (e.text !== '') ? query.where('ContactName', 'endswith',
e.text, true) : query;
        e.updateData(searchData, query);
    }
}
});
filter.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.TS

```
import { DropDownList, FilteringEventArgs } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
    url:
    'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let filter: DropDownList = new DropDownList({
    dataSource: searchData,
    query: new Query().select(['ContactName', 'CustomerID']).take(7),
    // map the appropriate column
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set placeholder to DropDownList input element
    placeholder: "Select a name",
    // set true to allow filtering for enable filtering supports
    allowFiltering: true,
    //set the height of the popup element
    popupHeight: "250px",
    //sort the resulted items
    sortOrder: 'Ascending',
    //bind the filtering event handler
    filtering: (e: FilteringEventArgs) => {
        // load overall data when search key empty.
        if(e.text == '') e.updateData(searchData);
        else{
            let query: Query = new Query().select(['ContactName',
            'CustomerID']);
            //enable the case sensitive filtering by passing false to 4th
            parameter.
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
            e.text, false) : query;
            e.updateData(searchData, query);
        }
    }
});
filter.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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
// create local data
let data: string[] = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aeromodelaje',
    'Águilas',
    'Ajedrez',
    'Ala Delta',
    'Álbumes de Música',
    'Alusivos',
    'Análisis de Escritura a Mano'];
// initialize DropDownList component
let ddlObj: DropDownList = new 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.TS

```
import { DropDownList, VirtualScroll } from '@syncfusion/ej2-dropdowns';
DropDownList.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i: number = 1; i <= 150; i++) {
    let item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //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: false,  
  //set the height of the popup element  
  popupHeight: '200px'  
});  
//render the component  
DropDownListObject.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.TS

```

import { DropDownList, VirtualScroll } from '@syncfusion/ej2-dropdowns';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
DropDownList.Inject(VirtualScroll);
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
    //bind the dataSource property
    dataSource: new DataManager({
        url: 'https://ej2services.syncfusion.com/js/development/api/orders',
        adaptor: new 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
DropDownListObject.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.TS

```
import { DropDownList, VirtualScroll } from '@syncfusion/ej2-dropdowns';
DropDownList.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i = 1; i <= 150; i++) {
  let item: { [key: string]: Object } = {};
  item.id = 'id' + i;
  item.text = `Item ${i}`;
  // Generate a random number between 1 and 4 to determine the group
  const randomGroup = Math.floor(Math.random() * 4) + 1;
  switch (randomGroup) {
    case 1:
      item.group = 'Group A';
      break;
    case 2:
      item.group = 'Group B';
      break;
    case 3:
      item.group = 'Group C';
      break;
    case 4:
      item.group = 'Group D';
      break;
    default:
      break;
  }
  records.push(item);
}
//initiates the component
let DropDownListObject: DropDownList = new DropDownList({
  //bind the dataSource 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
DropDownListObject.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.TS

```
import { DropDownList, VirtualScroll } from '@syncfusion/ej2-dropdowns';
DropDownList.Inject(VirtualScroll);
let records: { [key: string]: Object }[] = [];
for (let i: number = 1; i <= 150; i++) {
    let item = {
        id: 'id' + i,
        text: "Item " + i,
    };
    records.push(item);
}
//initiates the component
let DropDownListObject: DropDownList = new 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: true,
    //set the height of the popup element
    popupHeight: '200px'
});
//render the component
DropDownListObject.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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
// import L10n class for load function
import { L10n, setCulture } from '@syncfusion/ej2-base';
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
// bind remotedata to showcase actionFailureTemplate in offline.
let customerData: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
    adaptor: new ODataV4Adaptor,
    crossDomain: true
});
let dropDownObj: DropDownList = new 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 Query().select(['ContactName', 'CustomerID']).take(0);
    // set placeholder to DropDownList input element
    placeholder: 'Sélectionnez un élément'
  });
  dropDownObj.appendTo('#ddlelement');
  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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
let dropDownListObject: DropDownList = new DropDownList({
    //set the data to dataSource property
    dataSource: sportsData,
    // set placeholder to DropDownList input element
    placeholder: "Select a game"
    floatLabelType: "auto"
});
// render initialized DropDownList
dropDownListObject.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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
// defined the array of data
let gameList: { [key: string]: Object }[] = [
    { 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
let dropDownListObject: DropDownList = new 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 a game",
    // set the popup list height
    popupHeight: '200px'
  });
  // render initialized DropDownList
  dropDownListObject.appendTo('#ddlelement');
  document.onkeyup = function (e) {
    if (e.altKey && e.keyCode === 84 /* t */) {
      // press alt+t to focus the control.
      dropDownListObject.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>
  <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 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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
let dropDownListObject: DropDownList = new 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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';

```



```

//define the country DropDownList data
let countryData: { [key: string]: Object }[] = [
    { CountryName: 'United States', CountryId: '1' },
    { CountryName: 'Australia', CountryId: '2' }
];

//define the state DropDownList data
let stateData: { [key: string]: Object }[] = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];

//define the city DropDownList data
let cityData: { [key: string]: Object }[] = [
    { 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
let countryObj: DropDownList = new DropDownList({
    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 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
let stateObj: DropDownList = new 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 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
let cityObj: DropDownList = new 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>
</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 EJ2 JavaScript 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.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Button } from '@syncfusion/ej2-buttons';
// define the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
// initialize DropDownList component
let dropDownListObject: DropDownList = new 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('#ddlelement');
// 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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
// define the array of data
let sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
// initialize DropDownList component
let dropDownListObject: DropDownList = new 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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the data with category
let vegetableData: { [key: string]: Object }[] = [
  { 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
    let vegetables: DropDownList = new 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 EJ2 JavaScript 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.TS

```
// import highlightSearch module from ej2 dropdown package
import { DropDownList, FilteringEventArgs, highlightSearch } from
 '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ODataV4Adaptor,
  crossDomain: true
});
let text: string;
let filter: DropDownList = new DropDownList({
  dataSource: searchData,
  query: new Query().select(['ContactName', 'CustomerID']).take(7),
  // map the appropriate column
  fields: <{ [key: string]: Object }>{
    text: 'ContactName', value: 'CustomerID', itemCreated: (e: { [key:
string]: Object }) => {
      highlightSearch(e.item, text, true, 'StartsWith');
    }
  },
  // set placeholder to DropDownList input element
  placeholder: "Select a name",
  // set true to allowFiltering for enable filtering supports
  allowFiltering: true,
  //bind the filtering event handler
  filtering: (e: FilteringEventArgs) => {
    // take text for highlight the character in list items.
    text = e.text;
    let query: Query = new 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(searchData, query);
  }
});
filter.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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
let sortFormatData: { [key: string]: Object }[] = [
    { 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' }
];
let sortFormat: DropDownList = new DropDownList({
    dataSource: sortFormatData,
    // map the icon column to iconCSS field.
    fields: { text: 'Type', iconCss: 'Class', value: 'Id' },
    placeholder: 'Select 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 EJ2 JavaScript 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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
//initiates the component
let customers: DropDownList = new DropDownList({
    //bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',
        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }),

```

```

    //bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    //map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    //set the placeholder to DropDownList input
    placeholder:"Select a customer",
    //sort the resulted items
    actionComplete: function (e: any) {
        // 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
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>

```

Multiple cascading in EJ2 JavaScript Drop down list control

The following example demonstrate about how to preselect the list items in multiple cascading DropDownList.

INDEX.TS

```
import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query } from '@syncfusion/ej2-data';
//define the country DropDownList data
let countryData: { [key: string]: Object }[] = [
    { CountryName: 'United States', CountryId: '1' },
    { CountryName: 'Australia', CountryId: '2' }
];
//define the state DropDownList data
let stateData: { [key: string]: Object }[] = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
//define the city DropDownList data
let cityData: { [key: string]: Object }[] = [
    { 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
let countryObj: DropDownList = new 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
let stateObj: DropDownList = new 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
let cityObj: DropDownList = new 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(): void {
    // Query the data source based on state DropDownList selected value
    cityObj.query = new 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(): void {
    //Query the data source based on country DropDownList selected value
    stateObj.query = new 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 EJ2 JavaScript 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 [getItem](#) method.

The following example demonstrate how to get the total items count.

INDEX.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//import DataManager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
// initiates the component
let customers: DropDownList = new DropDownList({
    // bind the DataManager instance to dataSource property
    dataSource: new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/',

```

```

        adaptor: new ODataV4Adaptor,
        crossDomain: true
    }},
    // bind the Query instance to query property
    query: new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6),
    // map the appropriate columns to fields property
    fields: { text: 'ContactName', value: 'CustomerID' },
    // set the placeholder to DropDownList input
    placeholder: "Select a customer",
    // sort the resulted items
    actionComplete: function (e: any) {
        // get total items count
        console.log("Total items count: " + e.result.length);
        let element: HTMLElement = 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: HTMLElement = 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>

```

```

        <div style="margin: 50px">
            <button id="btn" class="e-btn e-control"> Get items</button>
        </div>
        <p id="event"> </p>
    </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>

```

Remove item in EJ2 JavaScript Drop down list control

The following example demonstrate about how to remove an item from DropDownList.

INDEX.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
let dropDownListObject: DropDownList = new 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
    let obj: any = 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 EJ2 JavaScript Drop down list control

The following example demonstrates about how to set limit the search result on filtering.

INDEX.TS

```
import { DropDownList, FilteringEventArgs } from '@syncfusion/ej2-
dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let searchData: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers',
  adaptor: new ODataV4Adaptor,
  crossDomain: true
```



```
});  
let filter: DropDownList = new DropDownList({  
    // set the remote data to dataSource property  
    dataSource: searchData,  
    // bind the Query instance to query property  
    query: new Query().select(['ContactName', 'CustomerID']).take(7),  
    // map the appropriate column  
    fields: { text: 'ContactName', value: 'CustomerID' },  
    // set placeholder to DropDownList input element  
    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: FilteringEventArgs) => {  
        // set limit as 4 to search result  
        let query: Query = new Query().select(['ContactName',  
'CustomerID']).take(4);  
        query = (e.text !== '') ? query.where('ContactName', 'startswith',  
e.text, true) : query;  
        e.updateData(searchData, query);  
    }  
});  
filter.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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Tooltip } from '@syncfusion/ej2-popups';
//Initialize DropDownList component
let listObj: DropDownList = new 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
let tooltip: Tooltip = new 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: TooltipEventArgs): void {
    // get the target element
    let listElement = document.getElementById('ddltooltip');
    let result: Object[] = listElement.ej2_instances[0].dataSource;
    let i: number;
    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 EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
//define the array of complex data
let sportsData: { [key: string]: Object }[] = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
//initiate the DropDownList
let dropDownListObject: DropDownList = new 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): void {
    let element: HTMLElement = 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 EJ2 JavaScript Drop down list control

yes, value for each list items should be unique.

Achieve virtual scrolling in EJ2 JavaScript 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.TS

```

import { DropDownList } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
  url: 'https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/',
  crossDomain: true
});
// initialize DropDownList component
let ddlObj: DropDownList = new DropDownList({
  // bind the DataManager instance to dataSource property
  dataSource: data,
  // bind the Query instance to query property
  query: new 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: any) {
    let operator: Query = new
Query().from('Customers').select('ContactName');
    let start: number = 7;
    let end: number = 12;

```

```

        let listElement: HTMLElement = this.list;
        listElement.addEventListener('scroll', () => {
            if ((listElement.scrollTop + listElement.offsetHeight >=
listElement.scrollHeight)) {
                let filterQuery = operator.clone();
                data.executeQuery(filterQuery.range(start,
end)).then((event: any) => {
                    start = end;
                    end += 5;
                    ddlObj.addItem(event.result as { [key: string]: Object
}[]);
                }).catch((e: Object) => {
                });
            }
        });
    });
    ddlObj.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>

```

Ej1 api migration in EJ2 JavaScript Drop down list control

This article describes the API migration process of DropDownList component from Essential JS 1 to Essential JS 2.

DataBinding

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *dataSource*
\$('#dropdown1').ejDropDownList({dataSource: List}); |

Property: *dataSource*
var dropDownListObject = new

ej.dropdowns.DropDownList({dataSource:

sportsData,});dropDownListObject.appendTo('#ddlelement');|

| **Fields for mapping** | **Property:** *fields*
\$('#dropdown1').ejDropDownList({fields: {text:

"text",value: "country",,});| **Property:** *fields*
var dropDownListObject = new

ej.dropdowns.DropDownList({fields: { text: 'Game', value: 'Id'

,});dropDownListObject.appendTo('#ddlelement');|

| **Query** | **Property:** *query*
\$('#dropdown1').ejDropDownList({query:

ej.Query().requiresCount(),});| **Property:** *query*
var dropDownListObject = new

ej.dropdowns.DropDownList({query: new Query().from('Customers').select(['ContactName',

'CustomerID']).take(6),});dropDownListObject.appendTo('#ddlelement');|

| **Begin event** | **Event:** *actionBegin*
\$('#dropdown1').ejDropDownList({actionBegin : function

(args) {/Do your changes /});| **Event:** *actionBegin*
var dropDownListObject = new

ej.dropdowns.DropDownList({actionBegin:

"actionBegin");dropDownListObject.appendTo('#ddlelement');|

| **Complete event** | **Event:** *actionComplete*
\$('#dropdown1').ejDropDownList({actionComplete

: function (args) {/Do your changes /});| **Event:** *actionComplete*
var dropDownListObject =

new ej.dropdowns.DropDownList({actionComplete:

"actionComplete");dropDownListObject.appendTo('#ddlelement');|

| **Failure event** | **Event:** *actionFailure*
\$('#dropdown1').ejDropDownList({actionFailure :

function (args) {/Do your changes /});| **Event:** *actionFailure*
var dropDownListObject = new

ej.dropdowns.DropDownList({actionFailure:

"actionFailure");dropDownListObject.appendTo('#ddlelement');|

| **Success event** | **Event:** *actionSuccess*
\$('#dropdown1').ejDropDownList({actionSuccess :

function (args) {/Do your changes /});| **Not Applicable** |

| **Data binding event** | **Event:** *dataBound*
 \$('#dropdown1').ejDropDownList({dataBound :

function (args) {/Do your changes /});| **Event:** *dataBind*
var dropDownListObject = new

ej.dropdowns.DropDownList({dataBind:

"dataBind");dropDownListObject.appendTo('#ddlelement');|

Filtering

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *enableFilterSearch*

\$(‘#dropdown1’).ejDropDownList({enableFilterSearch : true,}); | **Property:** *allowFiltering*

var dropDownListObject = new ej.dropdowns.DropDownList({allowFiltering: true});dropDownListObject.appendTo(‘#ddlelement’); |

| **Server filtering** | **Property:** *enableServerFiltering*

\$(‘#dropdown1’).ejDropDownList({enableServerFiltering : true,}); | **Property:** *allowFiltering*

var dropDownListObject = new ej.dropdowns.DropDownList({allowFiltering: true});dropDownListObject.appendTo(‘#ddlelement’); |

| **Filter type** | **Property:** *filterType*
\$(‘#dropdown1’).ejDropDownList({filterType : ej.FilterType.Contains,}); | <https://ej2.syncfusion.com/javascript/demos/#/material/drop-down-list/filtering.html> |

| **No Records Template** | **Not Applicable** | **Property:** *noRecordsTemplate*
 var dropDownListObject = new ej.dropdowns.DropDownList({noRecordsTemplate: “ NO DATA AVAILABLE”});dropDownListObject.appendTo(‘#ddlelement’); |

| **Filter Bar watermark text** | **Not Applicable** | **Property:** *filterBarPlaceholder*
var dropDownListObject = new ej.dropdowns.DropDownList({filterBarPlaceholder: “search”});dropDownListObject.appendTo(‘#ddlelement’); |

| **Ignore casing and diacritics** | **Not Applicable** | **Property:** *ignoreAccent*
var dropDownListObject = new ej.dropdowns.DropDownList({ignoreAccent: true});dropDownListObject.appendTo(‘#ddlelement’); |

| **Incremental search** | **Property:** *enableIncrementalSearch*
\$(‘#dropdown1’).ejDropDownList({enableIncrementalSearch : true,}); | **By default it is true** |

| **Case sensitivity** | **Property:** *caseSensitiveSearch*
\$(‘#dropdown1’).ejDropDownList({caseSensitiveSearch : true,}); | <https://ej2.syncfusion.com/javascript/demos/#/material/drop-down-list/filtering.html> |

| **Search event** | **Event:** *search*
\$(‘#dropdown1’).ejDropDownList({search : function (args) {/Do your changes /}}); | **Event:** *filtering*
var dropDownListObject = new ej.dropdowns.DropDownList({filtering: “search”});dropDownListObject.appendTo(‘#ddlelement’); |

Template

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *template*
\$(“#dropdown”).ejDropDownList({template: ‘<div class=“flag”> \${flag}> </div>’ + ‘<div class=“txt”> \${text} </div>’, width: “200px”}); | **Property:**


```
itemTemplate<br/>var dropDownListObject = new ej.dropdowns.DropDownList({itemTemplate:
"<span><span class='name'>${FirstName}</span><span class
='city'>${City}</span></span>"});dropDownListObject.appendTo('#ddlelement');|
```

| **Group Template** | **Not Applicable** | **Property:** *groupTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({ groupTemplate: "\${City}"});dropDownListObject.appendTo('#ddlelement');|

| **ValueTemplate** | **Not Applicable** | **Property:** *valueTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({valueTemplate: "\${FirstName} - \${City}"});dropDownListObject.appendTo('#ddlelement');|

| **Header Template** | **Property:** *headerTemplate*
\$("#dropdown").ejDropDownList({ headerTemplate: "<div class='header'>Flag Countries </div>"}); | **Property:** *headerTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({headerTemplate: "NameCity"});dropDownListObject.appendTo('#ddlelement');|

| **FooterTemplate** | **Not applicable** | **Property:** *footerTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({footerTemplate: " Total list items: " + sportsData.length + ""});dropDownListObject.appendTo('#ddlelement');|

| **No records Template** | **Not applicable** | **Property:** *noRecordsTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({noRecordsTemplate: " NO DATA AVAILABLE"});dropDownListObject.appendTo('#ddlelement');|

| **Action failure Template** | **Not applicable** | **Property:** *actionFailureTemplate*
var dropDownListObject = new ej.dropdowns.DropDownList({actionFailureTemplate: " Data fetch get fails"});dropDownListObject.appendTo('#ddlelement');|

Virtual Scrolling

```
<!-- markdownlint-disable MD010 -->
```

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *allowVirtualScrolling*

```
<br/>$("#dropdown1").ejDropDownList({allowVirtualScrolling : true,}); | Not applicable |
```

Applying CSS

```
<!-- markdownlint-disable MD010 -->
```

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *cssClass*
\$("#dropdown1").ejDropDownList({cssClass : "customClass",});|

| **Property:** *cssClass*
var dropDownListObject = new ej.dropdowns.DropDownList({cssClass: "customClass"});dropDownListObject.appendTo('#ddlelement');|

| **showRoundedCorner** | **Property:** *showRoundedCorner*

\$('#dropdown1').ejDropDownList({showRoundedCorner : true,}); | **Property:** *cssClass*

var dropDownListObject = new ej.dropdowns.DropDownList({cssClass:
 "customClass"});dropDownListObject.appendTo('#ddlelement');|

Sorting

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *enableSorting*
\$('#dropdown1').ejDropDownList({enableSorting :
 true,}); | **Acheivable through** [sortOrder](#) property |

| **Order of sorting** | **Property:** *sortOrder*
\$('#dropdown1').ejDropDownList({sortOrder :
 ej.sortOrder.Descending,}); | **Property:** *sortOrder*
var dropDownListObject = new
 ej.dropdowns.DropDownList({sortOrder:
 "Ascending"});dropDownListObject.appendTo('#ddlelement');|

Popup

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Popup height** | **Property:** *popupHeight*
\$('#dropdown1').ejDropDownList({popupHeight :
 "500px",}); | **Property:** *popupHeight*
var dropDownListObject = new
 ej.dropdowns.DropDownList({popupHeight:
 "300px"});dropDownListObject.appendTo('#ddlelement');|

| **Popup width** | **Property:** *popupWidth*

\$('#dropdown1').ejDropDownList({popupWidth : "500px",}); | **Property:** *popupWidth*

var dropDownListObject = new ej.dropdowns.DropDownList({popupWidth:
 "400px"});dropDownListObject.appendTo('#ddlelement');|

| **Popup show on load** | **Property:** *showPopupOnLoad*

 \$('#dropdown1').ejDropDownList({showPopupOnLoad : true,}); | **By default, the data load on
 demand.** |

| **enableAnimation** | **Property:** *enableAnimation*

\$('#dropdown1').ejDropDownList({enableAnimation : true,}); | **Not applicable** |

| **Popup resizing** | **Property:** *enablePopupResize*

\$('#dropdown1').ejDropDownList({enablePopupResize : true,}); | **Not applicable** |

| **Maximum Popup height** | **Property:** *maxPopupHeight*

\$('#dropdown1').ejDropDownList({maxPopupHeight : "500px",}); | **Not applicable** |

| **Minimum Popup height** | **Property:**
minPopupHeight
\$('#dropdown1').ejDropDownList({minPopupHeight : "500px",});
); |
Not applicable |

| **Maximum Popup width** | **Property:** *maxPopupWidth*
`
$('#dropdown1').ejDropDownList({maxPopupWidth : "500px",});` | **Not applicable** |

| **Minimum Popup width** | **Property:** *minPopupWidth*
`
$('#dropdown1').ejDropDownList({minPopupWidth : "500px",});` | **Not applicable** |

| **Loading data** | **Property:** *loadOnDemand* `
$('#dropdown1').ejDropDownList({loadOnDemand : true,});` | **By default, it is true** |

| **Popup showing manually** | **Method:** *showPopup*
`
$('#dropdown').ejDropDownList('showPopup')` | **Method:** *showPopup* `
var dropDownListObject = new ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')
dropDownListObject.showPopup();` |

| **Popup hiding manually** | **Method:** *hidePopup* `
$('#dropdown').ejDropDownList('hidePopup')` | **Method:** *hidePopup* `
var dropDownListObject = new ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')
dropDownListObject.hidePopup();` |

| **Before Popup hide event** | **Event:** *beforePopupHide*
`
$('#dropdown1').ejDropDownList({beforePopupHide : function (args) {/Do your changes /}});` | **Not applicable** |

| **Before Popup shown event** | **Event:** *beforePopupShown* `
$('#dropdown1').ejDropDownList({beforePopupShown : function (args) {/Do your changes /}});` | **Event:** *beforeOpen* `
var dropDownListObject = new ej.dropdowns.DropDownList({beforeOpen: "open"});dropDownListObject.appendTo('#ddlelement');` |

| **Popup hide event** | **Event:** *popupHide* `
$('#dropdown1').ejDropDownList({popupHide : function (args) {/Do your changes /}});` | **Event:** *close* `
var dropDownListObject = new ej.dropdowns.DropDownList({close: "close"});dropDownListObject.appendTo('#ddlelement');` |

| **Popup resize event** | **Event:** *popupResize* `
$('#dropdown1').ejDropDownList({popupResize : function (args) {/Do your changes /}});` | **Not applicable** |

| **Popup resize start event** | **Event:** *popupResizeStart* `
$('#dropdown1').ejDropDownList({popupResizeStart : function (args) {/Do your changes /}});` | **Not applicable** |

| **Popup resize stop event** | **Event:** *popupResizeStop* `
$('#dropdown1').ejDropDownList({popupResizeStop : function (args) {/Do your changes /}});` | **Not applicable** |

| **Popup shown event** | **Event:** *popupShown* `
$('#dropdown1').ejDropDownList({popupShown : function (args) {/Do your changes /}});` | **Event:** *open* `
var dropDownListObject = new ej.dropdowns.DropDownList({open: "open"});dropDownListObject.appendTo('#ddlelement');` |

Placeholder

`<!-- markdownlint-disable MD010 -->`

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Watermark text** | **Property:** *watermarkText*

```
<br/>$('#dropdown1').ejDropDownList({watermarkText : "Select"}); | Property: placeholder  
<br/>var dropDownListObject = new ej.dropdowns.DropDownList({placeholder:  
"select"});dropDownListObject.appendTo('#ddlelement');
```

| **Floating of watermark text** | **Not applicable** | **Property:** *floatLabelType*
var

```
dropDownListObject = new ej.dropdowns.DropDownList({floatLabelType:  
"Auto"});dropDownListObject.appendTo('#ddlelement');
```

Grouping

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Default** | **Property:** *fields.groupBy*
\$('#dropdown1').ejDropDownList({fields: {groupBy:
"text"},}); | **Property:** *fields.groupBy*
var dropDownListObject = new
ej.dropdowns.DropDownList({fields: {groupBy:
'ContactName',}});dropDownListObject.appendTo('#ddlelement');

| **Group Template** | **Not applicable** | **Property:** *groupTemplate*
var dropDownListObject = new
ej.dropdowns.DropDownList({ groupTemplate:
"\${City}");dropDownListObject.appendTo('#ddlelement');

Accessibility

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Globalization** | **Property:** *locale*
\$('#dropdown1').ejDropDownList({locale: "fr-FE"}); |
Property: *locale*
var dropDownListObject = new ej.dropdowns.DropDownList({ locale: "fr-
FE"});dropDownListObject.appendTo('#ddlelement');

| **Rtl support** | **Property:** *enableRTL*
\$('#dropdown1').ejDropDownList({enableRTL: true,}); |
Property: *enableRtl*
var dropDownListObject = new ej.dropdowns.DropDownList({ enableRtl:
true});dropDownListObject.appendTo('#ddlelement');

Miscellaneous

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| **Enable/disable** | **Property:** *enabled*
\$('#dropdown1').ejDropDownList({enabled: true,}); |
Property: *enabled*
var dropDownListObject = new ej.dropdowns.DropDownList({ enabled:
true});dropDownListObject.appendTo('#ddlelement');

| Read only | **Property:** *readOnly*
\$('#dropdown1').ejDropDownList({readOnly: true,}); |

Property: *readOnly*
var dropDownListObject = new ej.dropdowns.DropDownList({
 readOnly: true});dropDownListObject.appendTo('#ddlelement'); |

| Persistence of data | **Property:**
enablePersistence
\$('#dropdown1').ejDropDownList({enablePersistence: true,}); | **Property:**
enablePersistence
var dropDownListObject = new ej.dropdowns.DropDownList({
 enablePersistence: true});dropDownListObject.appendTo('#ddlelement'); |

| **Disable** | **Method:** *disable*
\$('#dropdown').ejDropDownList('disable') | **Property:**
enabled
var dropDownListObject = new ej.dropdowns.DropDownList({
 enabled:false});dropDownListObject.appendTo('#ddlelement'); |

| **Enable** | **Method:** *enable*
\$('#dropdown').ejDropDownList('enable') | **Property:**
enabled
var dropDownListObject = new ej.dropdowns.DropDownList({ enabled:
 true});dropDownListObject.appendTo('#ddlelement'); |

| **Height** | **Property:** *height*
\$('#dropdown1').ejDropDownList({height: "500px",}); | **Property:**
height
var dropDownListObject = new ej.dropdowns.DropDownList({ height:
 "300px"});dropDownListObject.appendTo('#ddlelement'); |

| **Width** | **Property:** *width*
\$('#dropdown1').ejDropDownList({width: "500px",}); |
Property: *width*
var dropDownListObject = new ej.dropdowns.DropDownList({ width:
 "300px"});dropDownListObject.appendTo('#ddlelement'); |

Selection

<!-- markdownlint-disable MD010 -->

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Selecting particular index | **Property:**
selectedIndex
\$('#dropdown1').ejDropDownList({selectedIndex: 3,}); | **Property:**
index
var dropDownListObject = new ej.dropdowns.DropDownList({ index:
 3});dropDownListObject.appendTo('#ddlelement'); |

| **Selecting particular value** | **Property:** *value*
\$('#dropdown1').ejDropDownList({value:
 "data",}); | **Property:** *value*
var dropDownListObject = new ej.dropdowns.DropDownList({
 value: "data"});dropDownListObject.appendTo('#ddlelement'); |

| **Selecting particular text** | **Property:** *text*
\$('#dropdown1').ejDropDownList({text:
 "data",}); | **Property:** *text*
var dropDownListObject = new ej.dropdowns.DropDownList({ text:
 "data"});dropDownListObject.appendTo('#ddlelement'); |

| **Target id** | **Property:** *targetId*
\$('#dropdown1').ejDropDownList({targetId: "Id",}); | **Not applicable** |

| **Selecting item using text** | **Method:** *selectItemByText*

\$('#dropdown').ejDropDownList('selectItemByText','car') | **Not applicable** |

| Unselect item using text | Method:

`unselectItemByText
$('#dropdown').ejDropDownList('unselectItemByText','car')` | **Not applicable** |

| Selecting item using value | Method:

`selectItemByValue
$('#dropdown').ejDropDownList('selectItemByValue','car')` | **Not applicable** |

| Getting data by using value | Method:

`getItemDataByValue
$('#dropdown').ejDropDownList('unselectItemByValue','car')` | **Method:**
`getDataByValue
var dropDownListObject = new`
`ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')
dropDownListObject.getDataByValue();|`

| Get selected value | Method:

`getSelectedItem
$('#dropdown').ejDropDownList('getSelectedItem')` | **Not applicable** |

| Get selected text | Method:

`getSelectedText
$('#dropdown').ejDropDownList('getSelectedText')` | **Property:** `text
var`
`dropDownListObject = new`
`ej.dropdowns.DropDownList({text:"data"});dropDownListObject.appendTo('#ddlelement')` |

| Select event | Event: `select
$('#dropdown1').ejDropDownList({select : function (args) {/Do your changes /}});|` **Event:** `select
var dropDownListObject = new`
`ej.dropdowns.DropDownList({select:`
`"onSelect"});dropDownListObject.appendTo('#ddlelement')` |

| Addition of Html attributes | Property:

`htmlAttributes
$('#dropdown1').ejDropDownList({htmlAttributes: { disabled: "disabled"},});|`
Property: `htmlAttributes
var dropDownListObject = new`
`ej.dropdowns.DropDownList({htmlAttributes:"attrib"});dropDownListObject.appendTo('#ddlelement')` |

Common

`<!-- markdownlint-disable MD010 -->`

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

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

| Adding new item | Method : `addItem
$('#dropdown').ejDropDownList("addItem",`
`{text:"India"});|` **Method:** `addItem
var dropDownListObject = new`
`ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')
dropDownListObject.addItem("data");|`

| Clearing the text | Method : `clearText
$('#dropdown').ejDropDownList('clearText')` |

Property: `value
var dropDownListObject = new`
`ej.dropdowns.DropDownList({value:""});dropDownListObject.appendTo('#ddlelement')` |

| Destroy the component | Method : `destroy
$('#dropdown').ejDropDownList('destroy')` |

Method: `destroy
var dropDownListObject = new`


```
ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')<br/>dropDownListObject.destroy(); |
```

| **Getting the data** | **Method** : *getListData*
\$('#dropdown').ejDropDownList('getListData')

| **Method** : *getItems*
var dropDownListObject = new
ej.dropdowns.DropDownList();dropDownListObject.appendTo('#ddlelement')
dropDownListObject.getItems(); |

| **Create event** | **Event**: *create*
\$('#dropdown1').ejDropDownList({create : function (args) {/Do your changes /}}); | **Event**: *created*
var dropDownListObject = new
ej.dropdowns.DropDownList({created:"create"});dropDownListObject.appendTo('#ddlelement')

| **Destroy event** | **Event**: *destroy*
\$('#dropdown1').ejDropDownList({destroy : function (args) {/Do your changes /}}); | **Event**: *destroyed*
var dropDownListObject = new
ej.dropdowns.DropDownList({destroyed:"create"});dropDownListObject.appendTo('#ddlelement')

| **Cascade event** | **Event**: *cascade*
\$('#dropdown1').ejDropDownList({cascade : function (args) {/Do your changes /}}); | <https://ej2.syncfusion.com/javascript/demos/#/material/drop-down-list/cascading.html> |

| **Change event** | **Event**: *change*
\$('#dropdown1').ejDropDownList({change : function (args) {/Do your changes /}}); | **Event**: *change*
var dropDownListObject = new
ej.dropdowns.DropDownList({change:"create"});dropDownListObject.appendTo('#ddlelement')

| **Focus out event** | **Event**: *focusOut*
\$('#dropdown1').ejDropDownList({focusOut : function (args) {/Do your changes /}}); | **Event**: *blur*
var dropDownListObject = new
ej.dropdowns.DropDownList({blur:"create"});dropDownListObject.appendTo('#ddlelement')

| **Focus in event** | **Event**: *focusIn*

\$('#dropdown1').ejDropDownList({focusIn : function (args) {/Do your changes /}}); | **Event**: *focus*
var dropDownListObject = new
ej.dropdowns.DropDownList({focus:"create"});dropDownListObject.appendTo('#ddlelement')

Dropdown Tree

Data binding in EJ2 JavaScript 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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
//define the nested array of JSON objects
let continents: { [key: string]: Object; }[] = [
  {
    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' }
    ]
},
];
let DropDownTreeObj: DropDownTree = new 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>
<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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let localData: { [key: string]: Object }[] = [
    { 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' },
    { 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' },
];
let DropDownTreeObj: DropDownTree = new 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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
//import data manager related classes
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
});
let query: Query = new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
let query1: Query = new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
let DropDownTreeObj: DropDownTree = new 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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let localData: { [key: string]: Object }[] = [
  { 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' },
  ];
  let DropDownTreeObj: DropDownTree = new 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>
<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 EJ2 JavaScript 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 $\{\dots\}$ interpolation syntax.

INDEX.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [
    { "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" }
];
let ddtreeObj: DropDownTree = new 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>
  <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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [
  { "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" }
];
let ddtreeObj: DropDownTree = new 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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [
    { "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" }
];
let ddtreeObj: DropDownTree = new 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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [ ];
let ddtreeObj: DropDownTree = new 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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
import { Query, DataManager, ODataV4Adaptor } from '@syncfusion/ej2-data';
let data: DataManager = new DataManager({
  url: 'https://services.odata.org/V4/Northwind/Northwind.svs',
  adaptor: new ODataV4Adaptor,
  crossDomain: true,
});
let query: Query = new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
let query1: Query = new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
let ddTreeObj: DropDownTree = new 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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [
  { "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" }
  ];
let checkList: DropDownTree = new 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>

```

```

    </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 following sample, the Dropdown Tree is rendered with custom value of the **customTemplate** property like **Selected items count: 2**.

INDEX.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let data: { [key: string]: Object }[] = [
    { "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" }
];
let checkList: DropDownTree = new 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 EJ2 JavaScript 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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let localData: { [key: string]: Object }[] = [
  { 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' },
  ];
  let DropDownTreeObj: DropDownTree = new 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>
</body>
</html>

```

```

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.TS

```

import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let localData: { [key: string]: Object }[] = [
    { 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' },
];
let DropDownTreeObj: DropDownTree = new 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.TS

```
import { DropDownTree } from '@syncfusion/ej2-dropdowns';
let localData: { [key: string]: Object }[] = [
  { 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' },
];
let DropDownTreeObj: DropDownTree = new 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 EJ2 JavaScript 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 EJ2 JavaScript 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=checkbox` | 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

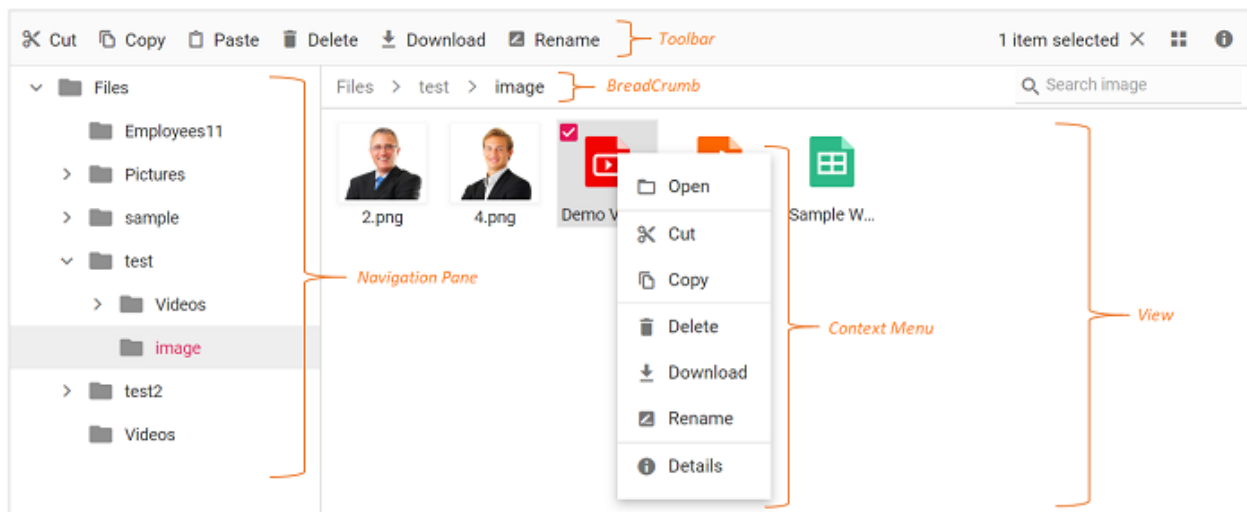
- [Accessibility in Syncfusion EJ2 JavaScript components](#)

FileManager

User interface in EJ2 JavaScript File manager control

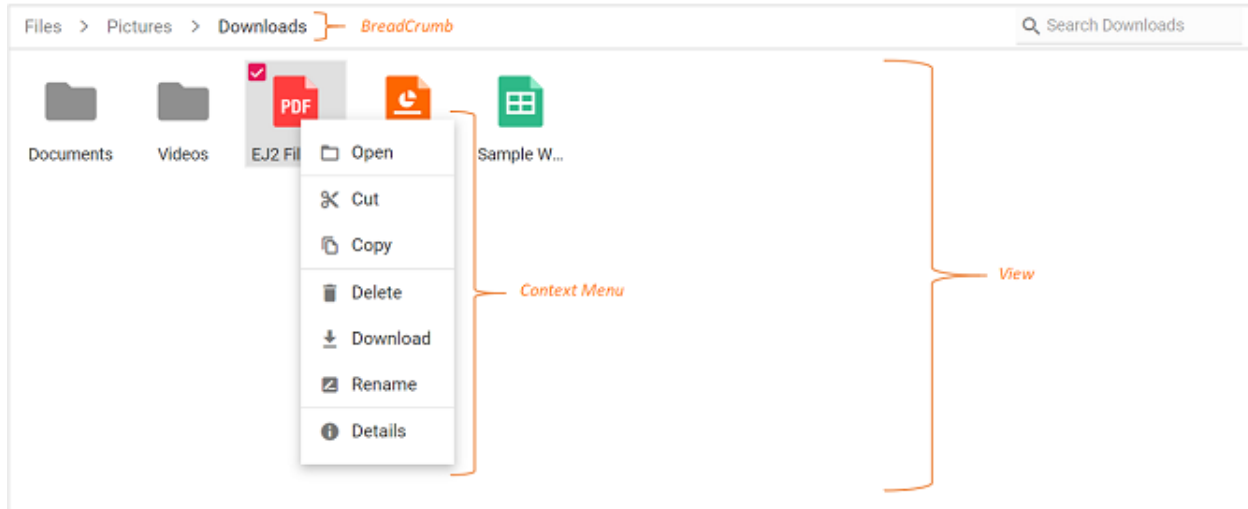
The file manager UI is comprised of several sections like view, toolbar, breadcrumb, context menu, and so on. The UI of the file manager is enhanced with injectable modules like **Details View** for browsing files and folders in a grid, **Navigation Pane** for folder navigation, and **Toolbar** for file 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)

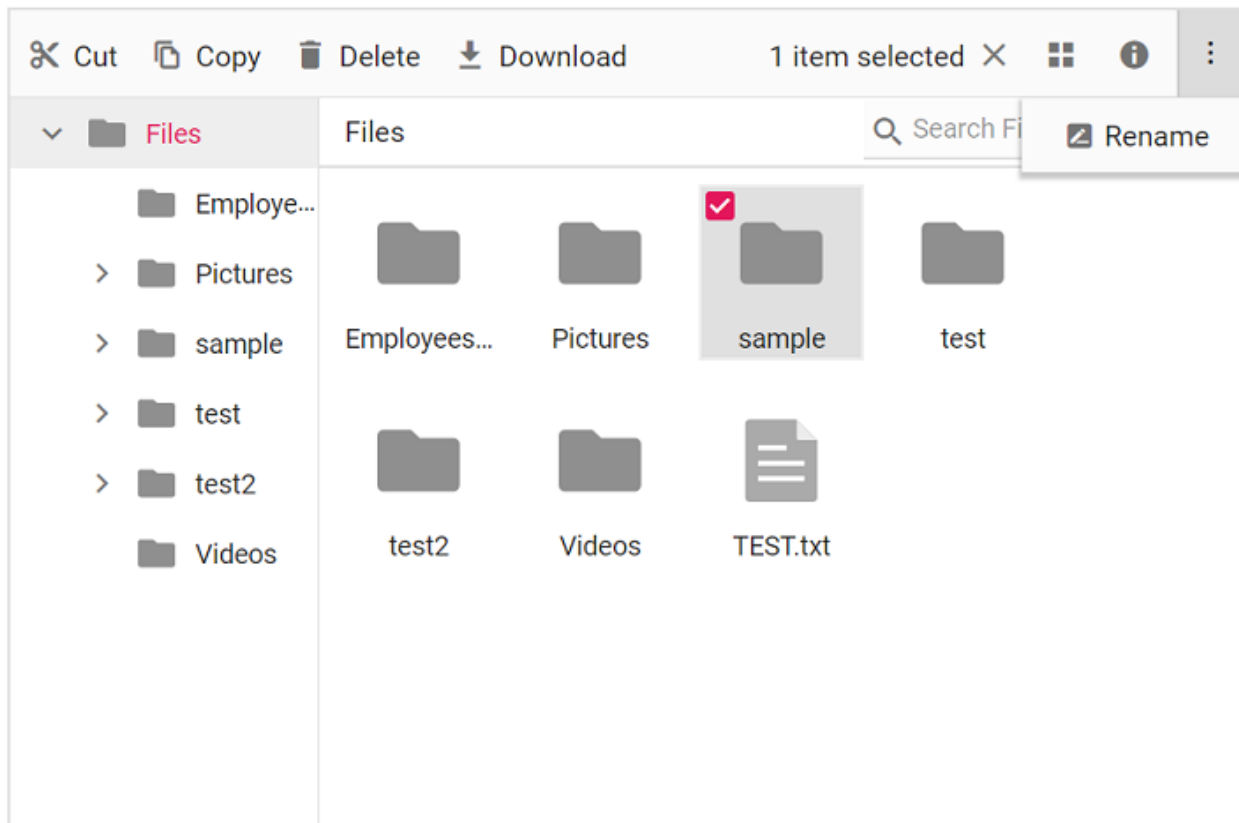


Toolbar

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

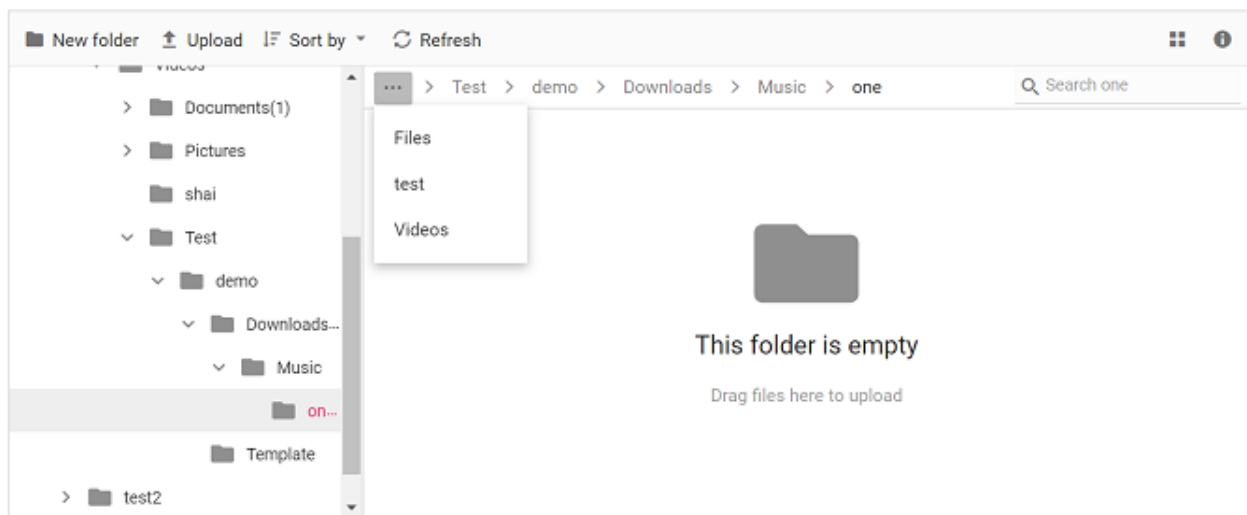
The navigation pane is an injectable module so, it should be injected before rendering the file manager to use its functionality.

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

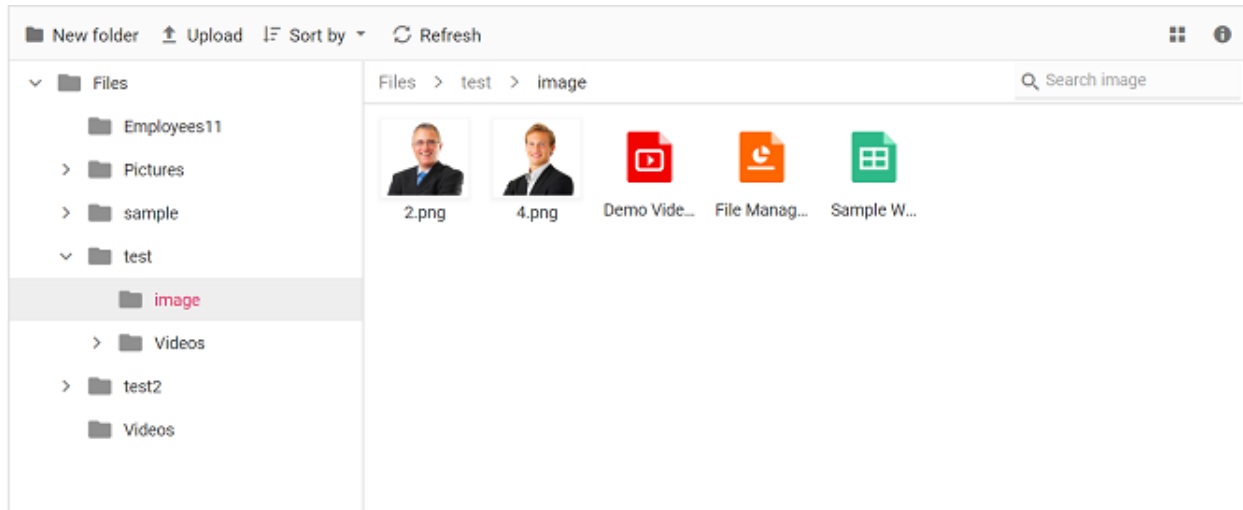
View is the section where the files and folders are displayed for the user to browse. The file manager has two types of views to display the files and folders.

- [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.

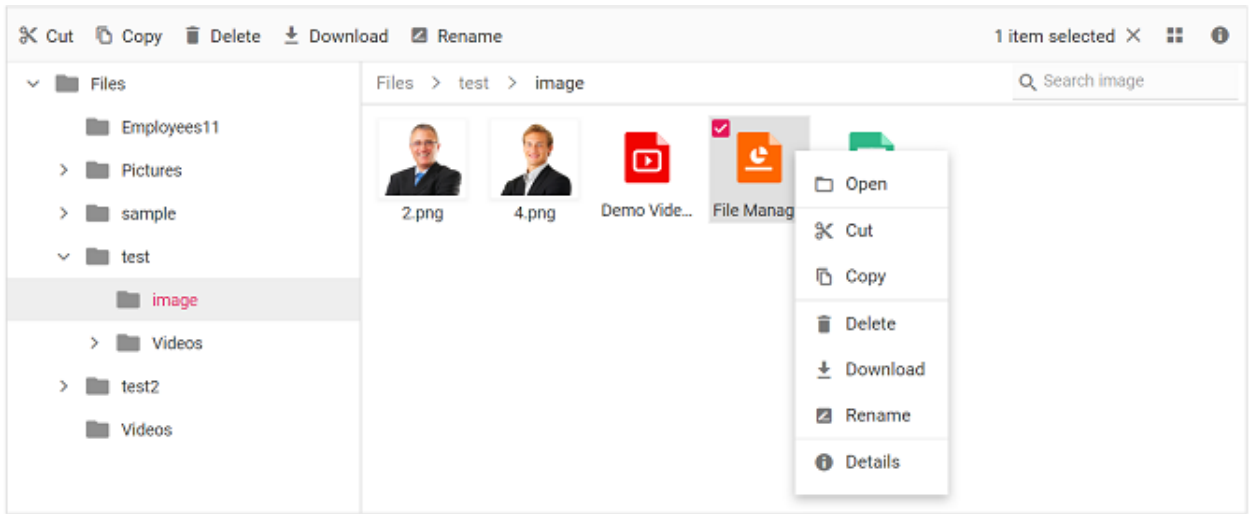
Name	Modified	Size
Employees11	July 19, 2019 16:52	
Pictures	July 18, 2019 15:45	
sample	July 19, 2019 16:33	
test	July 19, 2019 17:13	
test2	July 18, 2019 15:42	
Videos	July 18, 2019 15:51	
TEST.txt	July 19, 2019 02:18	0.019 KB

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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView, ContextMenu }
  from '@syncfusion/ej2-filemanager';
import { DropDownButton, ItemModel } from '@syncfusion/ej2-splitbuttons';
FileManager.Inject(Toolbar, NavigationPane, DetailsView, ContextMenu);
/**
 * File Manager folder upload sample
 */
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
let fileObject: FileManager = new 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');
let items: ItemModel[] = [{ text: 'Folder' }, { text: 'Files' }];
let drpDownBtn: DropDownButton = new DropDownButton({
  items: items,
  select: (args) => {
    if (args.item.text === 'Folder') {
      fileObject.uploadSettings.directoryUpload = true;
    } else {
      fileObject.uploadSettings.directoryUpload = false;
    }
    setTimeout(function () {
      let uploadBtn: HTMLElement = 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 **ActionResult 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|IList<IFormFile>|-|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",

fmpId: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,
fmpld: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> <i>Refresh</i> <i>NewFolder</i> Upload	<i>View Details</i>

1 (single item selected)	Delete Download* Rename	Selected items count View* Details
>1 (multiple selection)	Delete Download	Selected items count 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	SortBy View Refresh NewFolder Upload Details* Select all	Empty space in the view section (details view and large icon view area). Empty folder content.
Folders	Open Delete Rename Downloads* Details	* Folders in treeview, details view, and large icon view.
Files	Open Delete Rename Downloads* Details	* Files in details view and large icon view.

Views in EJ2 JavaScript 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](#)

Largelcons View

By Default, File Manager is rendered with largeicons view. The following example demonstrate this.

INDEX.TS

```
import { FileManager, Toolbar, NavigationPane } from '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
'@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)

```

```
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new FileManager({
  ajaxSettings: {
    url: hostUrl + 'api/FileManager/FileOperations',
    getImageUrl: hostUrl + 'api/FileManager/GetImage',
    uploadUrl: hostUrl + 'api/FileManager/Upload',
    downloadUrl: hostUrl + 'api/FileManager/Download'
  },
  // Context Menu settings customization
  contextMenuSettings: { file: ['Open', '|', 'Details'], folder: ['Open',
    '|', 'Details'], layout: ['SortBy', 'View', 'Refresh', '|', 'Details', '|'],
  visible: true}
});
// render initialized FileManager
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
'@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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",
    // 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 FileManager
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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 fileLoad event
    fileLoad: onBeforeFileLoad,
    // File Manager's fileOpen event

```

```

        fileOpen: onBeforeFileOpen
    });
    // render initialized FileManager
    filemanagerInstance.appendTo('#filemanager');
    // File Manager's file fileLoad function
    function onBeforeFileLoad(args: any){
        console.log(args.fileDetails.name + " is loading");
    }
    // File Manager's file fileOpen function
    function onBeforeFileOpen(args: any){
        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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
'@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
    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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView,
FileLoadEventArgs } from '@syncfusion/ej2-filemanager';
import { getValue, select } from '@syncfusion/ej2-base';
import { Tooltip, TooltipEventArgs } from '@syncfusion/ej2-popups';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize file manager component
let fileObj: FileManager = new FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download',
        getImageUrl: hostUrl + 'api/FileManager/GetImage'
    },
    created: () => { addTooltip(); },
    fileLoad: (args: FileLoadEventArgs) => {
        //Native tooltip customization to display additional information in
        new line
        let target: Element = args.element;
        if (args.module==='DetailsView') {
            let element: Element = select('[title]', args.element);
            let title: string = getValue('name', args.fileDetails) +
                '\n' + getValue('dateModified', args.fileDetails);
            element.setAttribute('title', title);
        } else if (args.module==='LargeIconsView') {
            let title: string = getValue('name', args.fileDetails) +
                '\n' + getValue('dateModified', args.fileDetails);
            target.setAttribute('title', title);
        }
    }
});

```

```

    }
  });
  // render initialized FileManager
  fileObj.appendTo('#filemanager');
  function addTooltip() {
    let tooltip: Tooltip = new Tooltip({
      target: '#' + fileObj.element.id + '_toolbar [title]',
      beforeRender: onTooltipBeforeRender
    });
    tooltip.appendTo('#' + fileObj.element.id + '_toolbar');
  }
  function onTooltipBeforeRender(args: TooltipEventArgs) {
    let buttonId: string = select('button', args.target).id;
    let description: string = '';
    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 (let i: number = 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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize file manager component
let filemanagerInstance: FileManager = new 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 FileManager
filemanagerInstance.appendTo('#filemanager');
// File Manager's file select event function
function onFileSelect(args: any){
  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 EJ2 JavaScript 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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize file manager component and add custom item to contextmenu
let filemanagerInstance: FileManager = new 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 false 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 FileManager
  filemanagerInstance.appendTo('#filemanager');
  // File Manager's file drag start event function
  function onFileDragStart(args: any){
    console.log("File drag start");
  }
  // File Manager's file drag stop event function
  function onFileDragStop(args: any){
    console.log("File drag stop");
  }
  // File Manager's file dragging event function
  function onFileDragging(args: any){
    console.log("File dragging");
  }
  // File Manager's file dropped event function
  function onFileDropped(args: any){
    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 EJ2 JavaScript 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');
```

```
,
```

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
`
```

Google Drive file system provider use the [Google Drive APIs](#) to read the file in the file system and uses the [OAuth 2.0](#) protocol for authentication and authorization. To authenticate from the client end, obtain OAuth 2.0 client credentials from the [Google API Console](#). To learn more about generating the client credentials from the from Google API Console, refer to this [link](#).

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 allows 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](#)

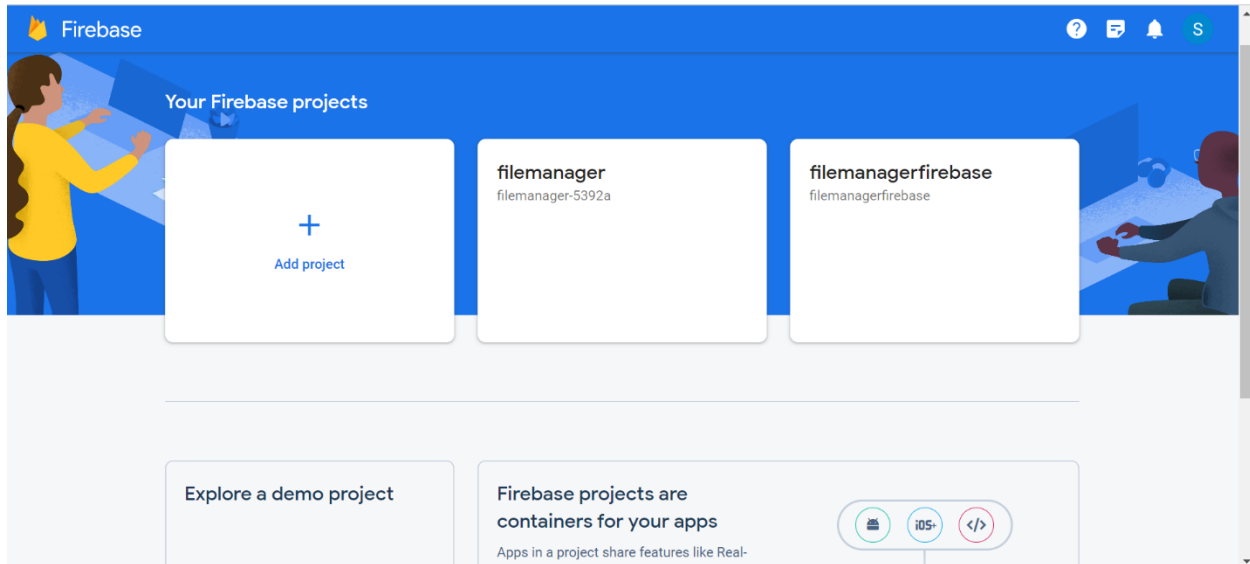
Firestore Realtime Database file system provider

The [Firestore 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.

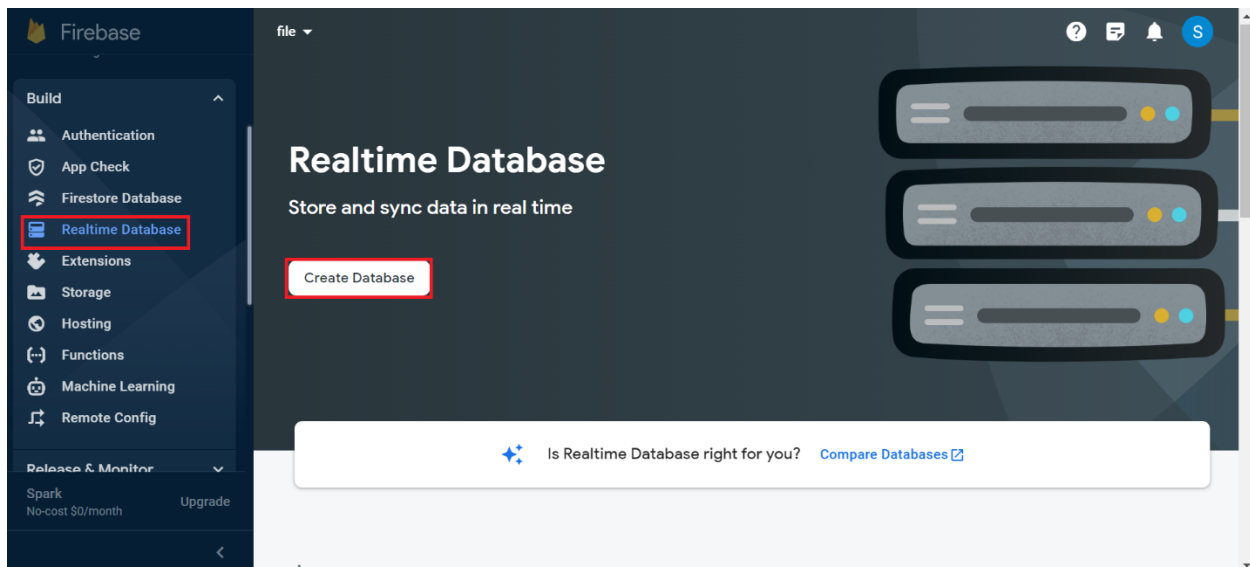
Generate Secret access key from service account

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:

```
`typescript
```

```
{
```

```
"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" : ""
```

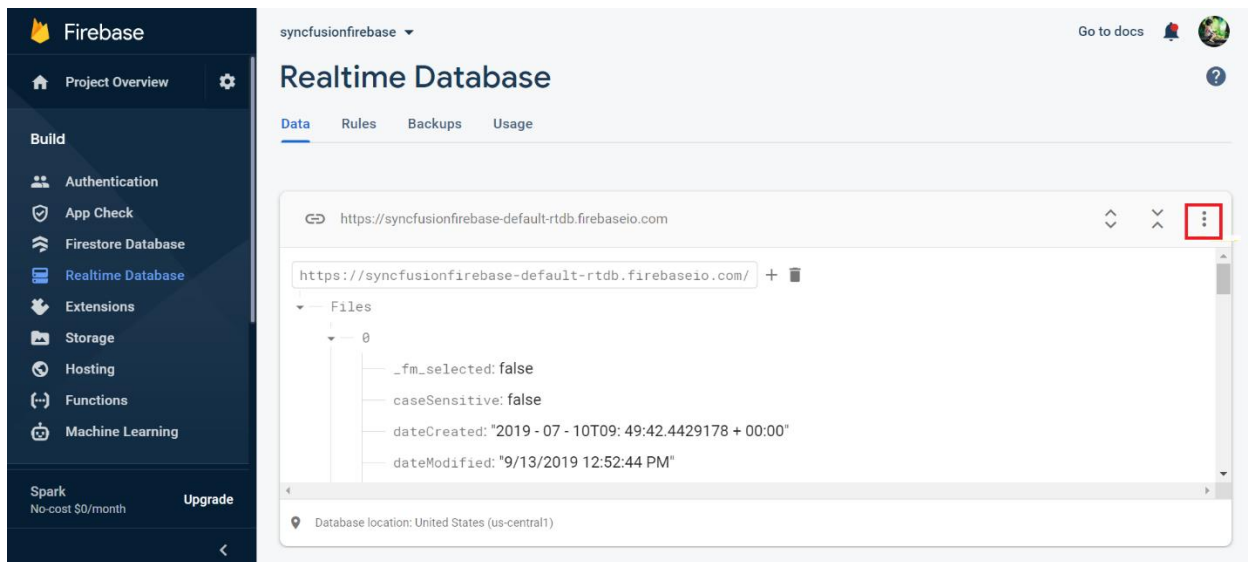
```

}}
}
,

```

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.

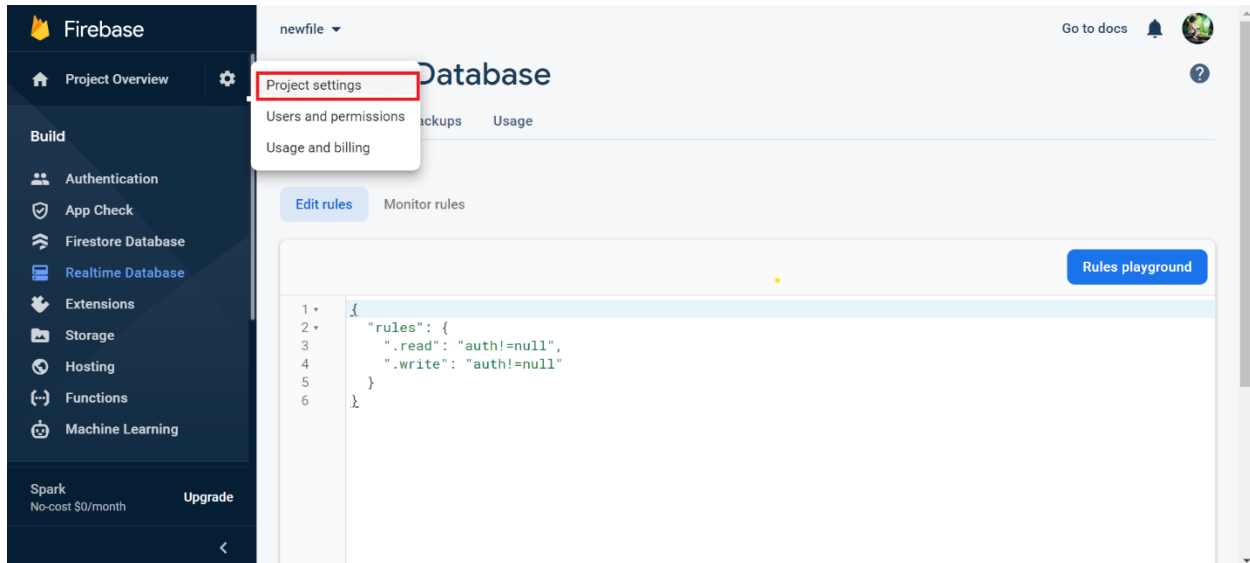
```

`typescript
{
/ Visit https://firebase.google.com/docs/database/security to learn more about security rules. /
"rules": {
".read": "auth!=null",
".write": "auth!=null"
}
}
,

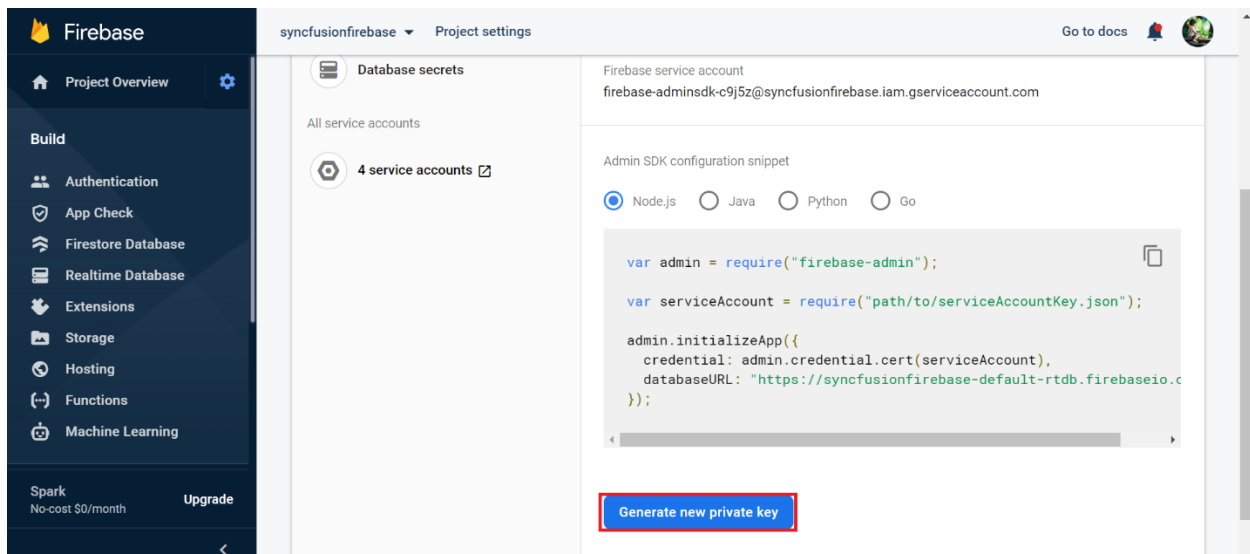
```

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.

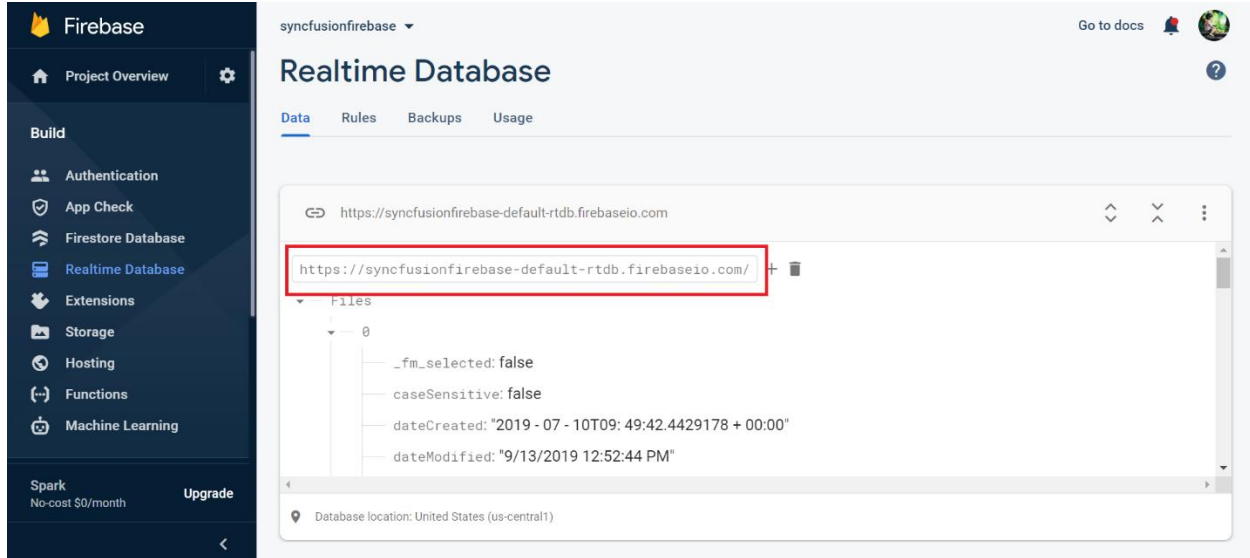


- To obtain the access key JSON file, simply click on the **Generate new private key** button and then confirm by clicking the **Generate key** button in the pop-up window that appears.



- 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.

```
`typescript
```

```
this.operation.RegisterFirebaseRealtimeDB(string apiUrl, string rootNode, string serviceAccountKeyPath)
`
```

Example:

```
`typescript
```

```
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.

```
`typescript
```

```
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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
import { L10n } from '@syncfusion/ej2-base';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
//Defining texts and messages corresponding to German culture
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"
    }
}
})
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component and add custom item to cuntextmenu
let filemanagerInstance: FileManager = new 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 FileManager
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 EJ2 JavaScript 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 `largelconsView` 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

The virtualization of the File Manager component is based on the height and width of the viewport. The items will be loaded in both `largelconsView` and `detailsView` based on the viewport size.

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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView,
enableVirtualization} from '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView,
enableVirtualization)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize file manager component and add custom item to contextmenu
let filemanagerInstance: FileManager = new 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';
}
});
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 EJ2 JavaScript 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 EJ2 JavaScript components](#)

Access control in EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView,
FileToolbarClickEventArgs } from '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component and add custom item to contextmenu
let filemanagerInstance: FileManager = new FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManagerAccess/FileOperations',
        getImageUrl: hostUrl + 'api/FileManagerAccess/GetImage',
        uploadUrl: hostUrl + 'api/FileManagerAccess/Upload',
        downloadUrl: hostUrl + 'api/FileManagerAccess/Download'
    },
    view: 'Details'
});
// render initialized FileManager
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 EJ2 JavaScript 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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView,
MenuClickEventArgs, MenuOpenEventArgs } from '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component and add custom item to contextmenu
let filemanagerInstance: FileManager = new 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: MenuOpenEventArgs) {
    for(let i: number = 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: MenuClickEventArgs) {
    if (args.item.text === 'Custom') {

```

```
        alert('You have clicked custom menu item')
    }
}
// render initialized FileManager
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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
import { CheckBox, ChangeEventArgs } from '@syncfusion/ej2-buttons';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
filemanagerInstance.appendTo('#filemanager');
// Render Checkbox in template
var checkbox: CheckBox = new CheckBox({ label: 'Select All', checked: false,
change: onChange }, '#checkbox');
// on checkbox change select all or clear selection
function onChange(args: ChangeEventArgs): void {
  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 EJ2 JavaScript 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.TS

```
import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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 EJ2 JavaScript 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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component
let filemanagerInstance: FileManager = new 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 FileManager
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 EJ2 JavaScript 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.TS

```
import { Uploader } from '@syncfusion/ej2-inputs';
import { Dialog } from '@syncfusion/ej2-popups';
import { FileManager, FileOpenEventArgs, Toolbar, NavigationPane,
DetailsView } from '@syncfusion/ej2-filemanager';
import { Button } from '@syncfusion/ej2-buttons';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
// Initialize the Uploader component
let uploadObject: Uploader = new Uploader();
uploadObject.appendTo('#fileupload');
// Initialize the Button component
let btnObj: Button = new Button();
btnObj.appendTo('#openBtn');
// Initialize the Dialog component
let dialogObj: Dialog = new Dialog({
    header: 'Open',
    showCloseIcon: true,
    closeOnEscape: false,
    width: '850px',
    visible: false,
    target: document.getElementById('target'),
    animationSettings: { effect: 'None' },
    open: dialogOpen,
    close: dialogClose
});
dialogObj.appendTo('#dialog');
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
document.getElementById('openBtn').onclick = (): void => {
    dialogObj.show();
    // Initialize the FileManager component
    let filemanagerInstance: FileManager = new FileManager({
        ajaxSettings: {
            url: hostUrl + 'api/FileManager/FileOperations',
            getImageUrl: hostUrl + 'api/FileManager/GetImage',
            uploadUrl: hostUrl + 'api/FileManager/Upload',
            downloadUrl: hostUrl + 'api/FileManager/Download'
        },
        allowMultiSelection: false,
        fileOpen : onFileOpen
    });
    filemanagerInstance.appendTo('#filemanager');
    dialogOpen();
};
// Uploader will be shown, if Dialog is closed
function dialogClose(): void {
    let filemanager: FileManager = (document.getElementById('filemanager')
as any).ej2_instances[0];
    filemanager.destroy();
    document.getElementById('container').style.display = 'block';
}
// Uploader will be hidden, if Dialog is opened
function dialogOpen(): void {
```

```

        document.getElementById('container').style.display = 'none';
    }
    // File Manager's fileOpen event function
    function onFileOpen(args: FileOpenEventArgs): void {
        let file: any = (args as any).fileDetails;
        if (file.isFile) {
            args.cancel = true;
            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>

```

```

        <div id="target" class="control-section">
            <div id="dialog">
                <div id="filemanager"></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>

```

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.TS

```

import { enableRipple } from '@syncfusion/ej2-base';
import { Tab } from '@syncfusion/ej2-navigations';
import { FileManager, Toolbar, NavigationPane, DetailsView, ContextMenu }
from '@syncfusion/ej2-filemanager';
FileManager.Inject(Toolbar, NavigationPane, DetailsView, ContextMenu);
enableRipple(true);
//Initialize Tab component
let tabObj: Tab = new Tab({
    heightAdjustMode: 'None',
    height: 320,
    showCloseButton: true,
    selected: onSelect,
});
//Render initialized Tab component
tabObj.appendTo('#tab_orientation');
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
let fileObject: FileManager = new FileManager({
    ajaxSettings: {
        url: hostUrl + 'api/FileManager/FileOperations',
        getImageUrl: hostUrl + 'api/FileManager/GetImage',
        uploadUrl: hostUrl + 'api/FileManager/Upload',
        downloadUrl: hostUrl + 'api/FileManager/Download'
    },
    view: 'Details'
});
fileObject.appendTo('#filemanager');
function onSelect(): void {
    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>

```

```

        </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 localization content in EJ2 JavaScript 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.TS

```

import { FileManager, Toolbar, NavigationPane, DetailsView } from
 '@syncfusion/ej2-filemanager';
import { L10n } from '@syncfusion/ej2-base';
FileManager.Inject(Toolbar, NavigationPane, DetailsView)
L10n.load({
    'en': {
        'filemanager': {
            // Change the File Upload text.
            "File-Upload": "Files to Upload",
            // Change the Empty folder text.
            "Folder-Empty": "Empty Folder",
        }
    }
})
let hostUrl: string = 'https://ej2-aspcore-service.azurewebsites.net/';
// initialize File Manager component.
let filemanagerInstance: FileManager = new 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 FileManager
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

Icons in EJ2 JavaScript Floating action button control

You can customize the icon and text of JavaScript(ES5) Floating Action Button(FAB) using [iconCss](#) and [content](#) properties.

FAB with icon

You can show icon only in Floating Action Button by setting [iconCss](#) property. You can show tooltip on hover to show additional details to end-user by setting [title](#) attribute.

```
`js
// Initialize the Floating Action Button control
var fab = new ej.buttons.Fab({
  iconCss: 'fab-icons fab-icon-people',
  target: '#targetElement'
});
// Render initialized Floating Action Button
fab.appendTo('#fab');
```

FAB with icon and text

You can show icon along with text in Floating Action Button by setting [iconCss](#) and [content](#) properties.

```
`js
// Initialize the Floating Action Button control
var fab = new ej.buttons.Fab({
  iconCss: 'fab-icons fab-icon-people',
  content: 'Contacts',
  target: '#targetElement'
});
// Render initialized Floating Action Button
fab.appendTo('#fab');
```

Icon position

You can change the position of icon when showing along with content by setting [iconPosition](#) property. By default, the icon is positioned on the left side together with text.

```
`js
// Initialize the Floating Action Button control
var fab = new ej.buttons.Fab({
  iconCss: 'fab-icons fab-icon-people',
  content: 'Contacts',
  target: '#targetElement',
  iconPosition: 'Right'
});
// Render initialized Floating Action Button
fab.appendTo('#fab');
```

Below example demonstrates a FAB with icon and text.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize the Floating Action Button control  
var fab = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    content: 'Contacts',  
    target: '#targetElement',  
    iconPosition: 'Right'  
});  
// Render initialized Floating Action Button  
fab.appendTo('#fab');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>  
    <title>EJ2 Floating Action 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 id="targetElement" style="position:relative;min-  
height:350px;border:1px solid;"></div>  
        <button id="fab" title="Contact"></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;
}
#container .col-sm-4 {
    min-height: 100px;
}
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
@font-face {
    font-family: 'fab-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAwAgTlMvMj0kSSoAAAEoAAAAMnTYXCCV5yuAAABlAAAAFRnbHlmHl6
slgAAAFQAAASQaGVhZCG5vSMAAADQAAAAANmhoZWEHowNkAAAArAAAAACRobXR4E6AAAAAAYAAAA
UbG9jYQKAYwAAAHoAAAAADGlheHABEGDDAAABCACAAACBuYWll0KnKeQAABOQAAAI9cG9zdBhgIA
AAAJEAAAAARwABAAADUv9qAfOEAAAA//QD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAAQAAZGIHNv8
PPPUACwPoAAAAAN9TvCUAAAAA3108JQAAAAAD9AP0AAAAACAACAAAAAAAAAAAAEAFAALCAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQPtAZAABQAAAnoCvAAAAITwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAUGZFZABA5wTnDANS/2oAWGP0AJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAPoPAAAAAAAAAAAAAAwAAABQAAwABAAAAFAAEAEAAAAAKAAgAAC5wTnCOCk5wz//wA
A5wTnCOCk5wz//wAAAAAAAAAAAAAEACgAKAAoACgAAAAEEAGADAQAQAAAAABGA5AFyAkGAAQAAAAA
D6gPqAAAsAAAEZESEVIREjESElIQHDegGu/lJ6/lIBrgPr/lJ6/lIBrnoAAgAAAAADkwP0AHQAAtgA
AJRUjFSElIzU/HjUjDxUvFSMVHx0DER8PPw8RLW8PDgHRiQF3ihISEhIRERAQDXaODg4NDQwLCws
JCQkHBWYGBAQDAgJXAgiDBAQFBQYHBWgIEhUWFxoahB4eHx8eHhwaGhcWFRICAcHBGUFBQAQDAgJ
XAgiDBAQGBgcICAkKCgsMDA0NDg8OEBAQEREREHMSdgECBQYICgoMDQ8PEBEREHMTehEREAE8PDQw
KCgQHBBQQCAQIFBgJCgwNDhAQERETEXMTEhEQDw8NDAsJBWyfArhbUVfbAgMDBAUFBgyHCAGICQo
KCgsLDAwMDQ0ODQ4PDg8PDxANDAsMCwwLCgsKCgkSEQ8NDAoHBGQBAGQGBwoMDQ8REGkKCgsKCww
LDAsMDRAPDW8ODw4NDg0NDawMCwsKCgoJCAgIBWYGBQUEAwMCpP64EA8ODg0NCwsJCQCCHBAQCAQE
CBAUGCakJCwwMBw0ODg8BUBAPDG4NDQsLCQkHBGUeAgEBAGQFBgcJCQsLDQ0ODg8AAAAAAwAAAAA
DxgPoABAAIqBmAAABHgIUdGiIlGIPgIyAR4CFA4CIi4CND4CMicOAhuUFhcOAQcuASMiDgIVFBY
XDgMMVzQ+AjiEahUzNC4CJz4BNTQ+AjiEahUzNC4CJz4BNTQuAiIBYBgkFRUKMTcwJBuvJDA3Aak
YJBuvJDE3MCQVFsqWN2kkNiArJic9FBxWLylJNiArJii2JxVDIDZJUkk2IEMVJziYisgnklSSTY
gQxUnNiImKyA2SVICCWSKMTcwJBuvJDA3MSQVAYYLJDE3MCQVFsqWNzEkFTMQNKKPLLYcFDOnJis
gNKKPLLYcETM+RyYpSTygIDZJKSZHPjMRHFYvKuk2ICA2SSkmRz4zERxVMclJNiAAAAADAAAAAA
0A/QAPwb/ALUAACUfdz8PLw8PDguFdZ8PLw8PDgmZEW8CFR8OIUhLwQ3IT8GEz8CNS8GIScjAsG
BAQIEBAUFBwcICAkJCgoKCgoKCQkICAcHBQUEBAIBAQBAGQEBQUHBWgICQkKCgoKCgoJCQgIBWY
GBQQEAgH+CwEBAGQEBQUHBWgICQkKCgoKCgoJCQgIBWcFBQQEAgEBAQECEBAQFBQcHCAGJCQgIBWY
KCgkJCAGHBGYFBAQCacLktUgIAQICBAQFBQcHCAGJCQkKCwJb/bsDAwIBASwBcQ8NDawKCAi8AwQ
CAGMFBWgJCv0VK6ZWcGoKCQkICAcGBGUEBAIBAQBAGQEBQUHBWgICQkKCgoKCgoJCQgIBWcFBQQ
EAgEBAQECEBAQFBQcHCAGJCQoKCgoKCgkJCAGHBGYFBAQCAQEBAQIEBAUFBWcICAkJCgoKCgoKCQk
ICAcHBQUEBAIBAQBAGQEBQUHBWgICQkKCgMW/on3JgwKCgoJCQgIBWYGBQQEAgEBZAEBawIJVAE
CBQUHCQoBUAMHBRAKQcghBQMCAAAAAAAAEgDeAEAAAAAAAAAAAAQAAAAEAAAAAAAAEACQABAAEAAAA
AAAIABwAKAAEAAAAAAAAAMACQARAEEEEAAAAAAQACQaaAAEAAAAAAAAAUACWAjaAEAAAAAAAAAYACQauAAE
AAAAAAaAlAA3AAEAAAAAAAAasAEgbJAAMAAQQJAAAAAGB1AAMAAQQJAAEAEGb3AAMAAQQJAAIADgC
JAAMAAQQJAAAEgCXAAAMAAQQJAAQAEgCPAAAMAAQQJAAUAFgC7AAMAAQQJAAAYAEGDRAAMAAQQJAAo
AWADJAAMAAQQJAAAsAJAE7IEZhYilJY29uc1JlZ3VsYXJGYWItSWNVbnNGYWItSWNVbnNWZXJzaW9
uIDEuMEZhYilJY29uc0ZvbncGZ2VuzXJhdGVkIHVzaW5nInFN5bmNmddXNpb24gTWV0cm8gU3RlZG1
vd3d3LnN5bmNmddXNpb24uY29tACAARGbBhAGIALQBjAGMAbwBuAHMAUGBlAGcAdQBsAGEAcgBGAGE
AYgAtAEkAYwBVAG4AcwBGAGEAYgAtAEkAYwBVAG4AcwBWAGUAcgBzAGkAbwBuACAAQQAuADAAARgB
hAGIALQBjAGMAbwBuAHMAARGbYAG4AdaAGAgBAZQBAGUAGUAcgBhAHQAZQBkACAAQDAQBzAGkAbgBnACA
AUwB5AG4AYwBmAUAUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB
5AG4AYwBmAUAUAcwBpAG8AbgAuAGMAbwBtAAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAA
```

```

AAAAABQECAQMBBAEFAQYAA2FkZANtaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
    font-family: 'fab-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.fab-icon-people:before {
    content: "\e70a";
}

```

Styles in EJ2 JavaScript Floating action button control

This section explains the different styles of Floating Action Button.

FAB styles

The JavaScript(ES5) Floating Action Button supports the following predefined styles that can be defined using the [cssClass](#) property. You can customize by replacing the [cssClass](#) property with the below defined class.

cssClass	Description
-----	-----
e-primary	Used to represent a primary action.
e-outline	Used to represent an appearance of button with outline.
e-info	Used to represent an informative action.
e-success	Used to represent a positive action.
e-warning	Used to represent an action with caution.
e-danger	Used to represent a negative action.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Floating Action Button control with applied warning style
var warningbtn = new ej.buttons.Fab({
    iconCss: 'e-icons e-edit',
    cssClass: 'e-warning',
    target: '#targetElement',
})
// Render initialized Floating Action Button
warningbtn.appendTo('#fab');

```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Floating Action 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 id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="fab"></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%;
}
/* start of onhover customization */
.e-fab.e-btn.fab-hover {
  padding: 6px 0px 10px 10px;
}
.fab-hover .text-container {
  overflow: hidden;
  width: 0;
```

```

margin: 0;
transition: width .5s linear 0s, margin .2s linear .5s;
}
.fab-hover:hover .text-container {
width: 35px;
margin: 0 5px;
transition: width .5s linear .2s, margin .2s linear 0s;
}
/* end of onhover customization */

```

Predefined Floating Action Button styles provide only the visual indication. So, Floating Action Button [content](#) property should define the Floating Action Button style for the users of assistive technologies such as screen readers.

Styles customization

To modify the Floating Action Button appearance, you need to override the default CSS of Floating Action Button control. Please find the list of CSS classes and its corresponding section in Floating Action Button control. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

| CSS Class | Purpose of Class |

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

| .e-fab .e-btn | To customize the FAB. |

| .e-fab .e-btn:hover | To customize the FAB on hover. |

| .e-fab .e-btn:focus | To customize the FAB on focus. |

| .e-fab .e-btn:active | To customize the FAB on active. |

| .e-fab .e-btn-icon | To customize the style of FAB icon. |

Show text on hover

By using [cssClass](#), you can customize the Floating Action Button to show text on hover with applied transition effect. For detailed information, refer [styles.css](#) file below.

The content will behave the same, when the [enableHtmlSanitizer](#) is enabled. Since we are adding only the valid tags in content, sanitizing the content will not affect it.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Floating Action Button control
var fab = new ej.buttons.Fab({
  iconCss: 'e-icons e-edit',
  content: '<span class="text-container"><span
class="textEle">Edit</span></span>',
  cssClass: 'fab-hover',
  target: '#targetElement',
});
// Render initialized Floating Action Button
fab.appendTo('#fab');

```

INDEX.HTML


```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 Floating Action 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 id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="fab"></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%;
}
/* start of onhover customization */
.e-fab.e-btn.fab-hover {
  padding: 6px 0px 10px 10px;
}
.fab-hover .text-container {
  overflow: hidden;
  width: 0;
  margin: 0;
  transition: width .5s linear 0s, margin .2s linear .5s;
}

```

```
}  
.fab-hover:hover .text-container {  
  width: 35px;  
  margin: 0 5px;  
  transition: width .5s linear .2s, margin .2s linear 0s;  
}  
/* end of onhover customization */
```

Positions in EJ2 JavaScript Floating action button control

The floating action button can be positioned anywhere on the [target](#) using the [position](#) property. If the target is not defined, then FAB is positioned based on the browser viewport.

The position values of Floating Action Button are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft
- BottomCenter
- BottomRight

`js

// Initialize the Floating Action Button control in BottomLeft position

```
var fab = new ej.buttons.Fab({  
  content: 'Add',  
  position: 'BottomLeft',  
  target: '#targetElement'  
});
```

// Render initialized Floating Action Button

```
fab.appendTo('#fab');
```

,

Below example demonstrates different supported positions of FAB.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize the Floating Action Button control in TopLeft position  
var fabObj1 = new ej.buttons.Fab({  
  iconCss: 'fab-icons fab-icon-people',  
  position: 'TopLeft',  
  target: '#target'  
});  
// Render initialized Floating Action Button
```

```
fabObj1.appendTo('#fab1');  
// Initialize the Floating Action Button control in TopCenter position  
var fabObj2 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'TopCenter',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj2.appendTo('#fab2');  
// Initialize the Floating Action Button control in TopRight position  
var fabObj3 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'TopRight',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj3.appendTo('#fab3');  
// Initialize the Floating Action Button control in MiddleLeft position  
var fabObj4 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'MiddleLeft',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj4.appendTo('#fab4');  
// Initialize the Floating Action Button control in MiddleCenter position  
var fabObj5 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'MiddleCenter',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj5.appendTo('#fab5');  
// Initialize the Floating Action Button control in MiddleRight position  
var fabObj6 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'MiddleRight',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj6.appendTo('#fab6');  
// Initialize the Floating Action Button control in BottomLeft position  
var fabObj7 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'BottomLeft',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj7.appendTo('#fab7');  
// Initialize the Floating Action Button control in BottomCenter position  
var fabObj8 = new ej.buttons.Fab({  
    iconCss: 'fab-icons fab-icon-people',  
    position: 'BottomCenter',  
    target: '#target'  
});  
// Render initialized Floating Action Button  
fabObj8.appendTo('#fab8');
```

```
// Initialize the Floating Action Button control in BottomRight position
var fabObj9 = new ej.buttons.Fab({
  iconCss: 'fab-icons fab-icon-people',
  position: 'BottomRight',
  target: '#target'
});
// Render initialized Floating Action Button
fabObj9.appendTo('#fab9');
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta 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="target" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <button id="fab1" title="Top Left"></button>
    <button id="fab2" title="Top Center"></button>
    <button id="fab3" title="Top Right"></button>
    <button id="fab4" title="Middle Left"></button>
    <button id="fab5" title="Middle Center"></button>
    <button id="fab6" title="Middle Right"></button>
    <button id="fab7" title="Bottom Left"></button>
    <button id="fab8" title="Bottom Center"></button>
    <button id="fab9" title="Bottom Right"></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 {
```

Copyright © 2001 -2024 Syncfusion Inc.

3593

```

5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAAAAIAAAAAAAACgAAAAAAAAAAAAAAAAAAAA
AAAAABQECAQMBBAEFAQYAA2FkZANtaWMGcGVvcGxlCHNob3BwaW5nAAAA)
format('trueType');
    font-weight: normal;
    font-style: normal;
}
[class^="fab-icon-"],
[class*=" fab-icon-"] {
    font-family: 'fab-icons' !important;
    speak: none;
    font-size: 55px;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.fab-icon-people:before {
    content: "\e70a";
}

```

Custom position

You can define the custom position of the Floating Action Button by override the **top**, **left**, **right**, and **bottom** CSS properties using [cssClass](#). For detailed information, refer [styles.css](#) file below.

INDEX.JS

```

ej.base.enableRipple(true);
// Initialize the Floating Action Button control
var fab = new ej.buttons.Fab({
    iconCss: 'e-icons e-edit',
    cssClass: 'custom-position',
    target: '#targetElement',
})
// Render initialized Floating Action Button
fab.appendTo('#fab')

```

INDEX.HTML

```

<!DOCTYPE html><html lang="en"><head>
  <title>EJ2 SplitButton</title>
  <meta charset="utf-8">
  <meta 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="fab"></button>
        <div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    </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-fab.e-btn.custom-position {
    left: 40px;
    top: 40px;
    bottom: unset;
    right: unset;
}
```

Events in EJ2 JavaScript Floating action button control

This section explains the available events in Floating Action Button control.

created

Event triggers after the creation of Floating Action Button.

```
`js
```

```
// Initialize the Floating Action Button control
```

```
var fab = new ej.buttons.Fab({
```

```
iconCss: 'e-icons e-edit',
```

```
content:'Edit',
```

```
created:()=>{  
  //Your required action here  
}  
});  
// Render initialized Floating Action Button  
fab.appendTo('#fab');  
`
```

onclick

Event triggers when the Floating Action Button is clicked.

```
`js  
// Initialize the Floating Action Button control  
var fab = new ej.buttons.Fab({  
  iconCss: 'e-icons e-edit',  
  content: 'Edit'  
});  
// Render initialized Floating Action Button  
fab.appendTo('#fab');  
// onclick event handler  
fab.element.onclick = function() {  
  //Your required action here  
};  
`
```

Below example demonstrates the click event of the Floating Action Button.

INDEX.JS

```
ej.base.enableRipple(true);  
// Initialize the Floating Action Button control  
var fab = new ej.buttons.Fab({  
  iconCss: 'e-icons e-edit',  
  content: 'Edit' });  
// Render initialized Floating Action Button  
fab.appendTo('#fab');  
// onclick event handler  
fab.element.onclick = function() {  
  alert("Edit is clicked!");  
};
```

INDEX.HTML

```
<!DOCTYPE html><html lang="en"><head>
```



```

<title>EJ2 SplitButton</title>
<meta charset="utf-8">
<meta 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="targetElement" style="position:relative;min-
height:350px;"></div>
        <button id="fab"></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: 'button-icons';
    src:url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKIAAAwAgT1MvMjluSf8AAAEoAAAAVmNtYXDOXM6wAAABtAAAAFRnbHlmcV/
SKgAAAIQAAAJAaGVhZBnt0QcAAADQAAAAANmhoZWEIUQQOAAAArAAAACRobXR4NAAAAAAYAAAAA
0Bg9jYQNWAA+AAAAIIAAAAHG1heHABGQAZAAABCAAAACBuYW1lASvfhQAABGQAAAJhcG9zdFAouWk
AAABIAAAA2AABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAADQABAAAAQAAYD3WXF8
PPPUACwQAAAAAANGtxgsAAAAA2C3GCwAAAAAD9AP0AAAACAACAAAAAAAAAAAEAAAAANAA0AAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wHnDQQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA

```

```

EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAIAAADAAAAFAA
DAAEAAAAUAAQAQAAAAAYABAABALnCOcN//8AAOcB5wr//wAAAAAAQAGABQAAAABAAMABAHAAT
ACgAJAAgABQAGAAASADAAAAAADgAkAEQAWgByAIoApgDAAOAA+AEMASAAAQAAAAADYQP0AAIAADc
JAZ4CxP08DAH0AfQAAAIAAAAA9QD9AADAACaACUHESEBIREhAm4BZv6a/b4BZv6aDAPo/BgD6AA
AAgAAAAADpwP0AAMADAAANyE1ISUBBwkBJwERI1kDTvyyAYH+4y4BeQGAnv7UTaxNlwEIPf6eAWI
9/ukDEwAAAAIAAAAA/QDngADAACaADchNSETAyEBDAPo/Bj6+gPo/gxipgFy/t0CRwAAAQAAAAA
D9AP0AAsAAAEhFSERMxEhNSERIwHC/koBtNwBtv5KfAI+fP5KAbZ8AbYAAQAAAAAD9AP0AAsAAAE
hFSERMxEhNSERIwHh/isB1T4B1f4rPgIfPv4rAdU+AdUAAgAAAAAD9A0LAAMADAAANyE1ISUnBxc
3JwCRIwD6PwYAcWjLO7uLKI/Wj+hoSvs6iyhAm0AAAABAAAAAAP0A/QACwAAAREhFSERMxEhNSE
RAeH+KwHVPgHV/isD9P4rPv4rAdU+AdUAAAAAgAAAAADdwP0AAMADAAANyE1ISUBBwkBJwERI4k
C7v0SAVj+0SkBdgF4Kf7RPgw+rQEJL/64AUgv/vgC/AAAAEAAAAA/QD9AALAAABIRUhETMRITU
hESMB2v4yAc5Mac7+MkwCJkz+MgHOTAHOAAIAAAAA/QDzQADAACaADchNSE1KQEBDAPo/BgB9AH
0/gwzpZUCYAACAAAAAAP0A80AAwAHAAA3ITUhNSkBAQwD6PwYafQB9P4MM6WVAmAAAAASAN4AAQA
AAAAAAAABAAAAAQAAAAAAQAMAAEAAQAAAAAAAgAHAA0AAQAAAAAAAwAMABQAAQAAAAABAAMACA
AAQAAAAABQALACwAAQAAAAABgAMADcAAQAAAAACgAsAEMAAQAAAAACwASAG8AAwABBAkAAAA
CAIEAAwABBAkAAQAYAIMAAwABBAkAAgAOAJsAAwABBAkAAwAYAKkAAwABBAkABAAYAMEAAwABBAk
ABQAWANKAAwABBAkABgAYAO8AAwABBAkACgBYAQcAAwABBAkACwAkAV8gYnV0dG9uLWljb25zUmV
ndWxhcmJldHRvb1pY29uc2JldHRvb1pY29uc1Zlcnpb24gMS4wYnV0dG9uLWljb25zRm9udCB
nZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20
AIABiAHUAdAB0AG8AbgAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGIAdQB0AHQAbwBuAC0AaQB
jAG8AbgBzAGIAdQB0AHQAbwBuAC0AaQBjAG8AbgBzAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABIAHU
AdAB0AG8AbgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEAdAB1AGQAIAB1AHMAaQB
uAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaGAAAAAaKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAANAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4ACm1lZG1hLXBsYXkLbWVkaWEtcGF
lc2UQLWRvd25sb2FkLTAyLXdmLQltZW51bmQHYWRkLW5ldwtuZXctbWFpbC13Zhb1c2VyLWR
vd25sb2FkLXdmDGV4cGFuZC0wMy13Zg5kb3dubG9hZC0wMi13ZgphZGQtbmV3XzAxZC21lZG1hLWV
qZWNOdm1lZG1hLWVqZWNOlTAxAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.e-btn-sb-icons {
    font-family: 'button-icons';
    line-height: 1;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-play-icon::before {
    content: '\e701';
}
.e-pause-icon::before {
    content: '\e705';
}
}

```

Form Validator

Validation rules in EJ2 JavaScript Form validator control

Default Rules

The **FormValidator** has following default validation rules, which are used to validate the form.

| Rules | Description | Example |

Rule	Message
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 EJ2 JavaScript 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.TS

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';
let options: FormValidatorModel = {
  rules: {
    'user': { required: true },
    'password': { minLength: [6, 'Need atleast 6 character length'] }
  }
}
let formObject: FormValidator = new 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.TS

```

import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';
let options: FormValidatorModel = {
    rules: {
        'user': { required: [true, 'User Name: required'] },
        'password': { minLength: [5, 'Password: need atleast 5 characters'] }
    },
    customPlacement: (inputElement: HTMLElement, error: HTMLElement)=>{
        document.getElementById('error').appendChild(error);
    }
}
let formObject: FormValidator = new 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 EJ2 JavaScript 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.TS

```
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { L10n } from '@syncfusion/ej2-base';
// Load `German` culture to override spin buttons tooltip text
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"
    }
  },
});
let options: FormValidatorModel = {
  rules: {
    email: { email: true },
  },
  locale: "ar-AR"
}
let formObject: FormValidator = new 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;
}

```

