

USER GUIDE

Essential Studio

for EJ2 Angular

Version - v25.2.3 | Release Date - May 08, 2024

Signature	31
Getting started with Angular Signature component	31
Dependencies.....	31
Setup Angular environment.....	31
Create an Angular application	31
Installing Syncfusion Signature Package	31
Adding Signature module	32
Adding Syncfusion Signature component.....	32
Adding CSS reference.....	33
Running the application.....	33
Customization in Angular Signature component.....	34
Stroke Width	34
Stroke Color	35
Background Color.....	35
Background Image	36
See Also	37
Open save in Angular Signature component	37
Open Signature	37
Save Signature.....	38
Save With Background	40
Draw in Angular Signature component.....	42
Draw.....	42
User interaction in Angular Signature component	43
Undo.....	44
Clear	44
Disabled.....	44
ReadOnly.....	44
User Integration sample.....	44
Accessibility in Angular Signature component	46
Keyboard interaction	47
Ensuring accessibility	47
See also	47
How To	47
Integrate toolbar in Angular Signature component.....	47
Skeleton	53

Getting started with Angular Skeleton component.....	53
Dependencies.....	53
Setup Angular environment.....	53
Create an Angular application	53
Installing Syncfusion Notifications package.....	53
Adding Skeleton module	54
Add Skeleton into application.....	55
Adding CSS reference.....	55
Running the application.....	56
Shapes in Angular Skeleton component	56
Circle skeleton shape	56
Square skeleton shape	57
Rectangle skeleton shape	57
Text skeleton shape	57
Shimmer effect in Angular Skeleton component.....	59
Styles in Angular Skeleton component	61
cssClass.....	61
Visible.....	62
Accessibility in Angular Skeleton component.....	62
WAI-ARIA attributes.....	63
Ensuring accessibility	63
See also	63
Smith Chart	63
Getting started with Angular Smithchart component	63
Setup Angular Environment.....	63
Create an Angular Application	64
Installing Syncfusion SmithChart package.....	64
Registering TreeMap Module	65
Module Injection.....	66
Add Series to Smithchart	67
Add title to SmithChart	69
Enable Marker to Smithchart.....	70
Enable DataLabel to Smithchart Marker.....	72
Enable Legend for Smithchart.....	73
Enable Tooltip for Smithchart Series	75

Working with data in Angular Smithchart component.....	77
Data Binding.....	77
Smith chart dimensions in Angular Smithchart component.....	78
Size for Container.....	78
Size for Smithchart.....	79
Title subtitle in Angular Smithchart component.....	81
Enable title	81
Title trim.....	82
Smith chart axis in Angular Smithchart component	83
Labels Customization	84
Gridlines	85
Axisline	87
Smith chart legend in Angular Smithchart component	89
Position and Alignment.....	89
Customization	93
Toggle Visibility	97
Smith chart tooltip in Angular Smithchart component	99
Smith chart marker in Angular Smithchart component.....	100
Marker.....	100
Datalabels	102
Smith chart series in Angular Smithchart component.....	105
points or datasource	105
Series customization	106
Smith chart print in Angular Smithchart component	107
Print.....	107
Export.....	108
Accessibility in Angular Smithchart component	110
WAI-ARIA attributes.....	111
Keyboard interaction	111
Ensuring accessibility	111
See also	111
Sparkline	111
Getting started with Angular Sparkline component.....	111
Setup Angular Environment.....	111
Create an Angular Application	111

Adding Syncfusion Sparkline package.....	112
Registering Sparkline Module	112
Module Injection.....	113
Bind data source to Sparkline	114
Change the type of Sparkline	115
Enable tooltip for Sparkline	116
Sparkline dimension in Angular Sparkline component.....	117
Size for container	117
Size for sparkline	118
Sparkline types in Angular Sparkline component	119
Axis customization in Angular Sparkline component	125
Change value type of the sparkline.....	125
Change min and max values of axis	127
Change value of axis.....	128
Axis line customization	129
Special points customization in Angular Sparkline component.....	130
Range band in Angular Sparkline component	132
Range band customization.....	132
Multiple range band customization	133
Marker in Angular Sparkline component.....	134
Adding marker to the sparkline	134
Adding marker to special point.....	135
Customizing markers.....	135
Data labels in Angular Sparkline component.....	136
Enable data label.....	136
Customize data label.....	137
Format data label text.....	138
User interaction in Angular Sparkline component	139
Tooltip	139
Track line	142
Appearance in Angular Sparkline component	143
Sparkline border.....	143
Sparkline padding.....	144
Sparkline area customization.....	144
Sparkline theme	145

Localization in Angular Sparkline component	146
Tooltip format	146
Rtl	147
Accessibility in Angular Sparkline component	148
WAI-ARIA attributes	149
Keyboard interaction	149
Ensuring accessibility	149
See also	149
Ej1 api migration in Angular Sparkline component	149
Sparkline Types	149
Databinding	150
Markers	150
Data labels	151
Range band	152
Special points customization	152
Axis customization	153
Appearance customization	154
Tooltip	155
Rendering	156
Localization	156
Methods	157
Events	157
Speed Dial	158
Dependencies	159
Setup Angular environment	159
Create an Angular application	159
Installing Syncfusion Buttons package	159
Adding SpeedDial module	160
Adding Syncfusion SpeedDial component	160
Adding CSS reference	161
Running the application	161
Items in Angular Speed dial component	162
Icons in Speed Dial items	162
Animation	165
Template	166

Positions in Angular Speed dial component	166
Opens items on hover	167
Programmatically show/hide Speed Dial items	168
Programmatically refresh the position	169
Display modes in Angular Speed dial component	170
Linear display mode	170
Radial display mode (Radial Menu)	171
Radial menu in Angular Speed dial component.....	171
Radial menu direction	171
Radial menu start and end angle	172
Offset.....	173
Template in Angular Speed dial component.....	174
Item template	174
Popup template	175
Styles in Angular Speed dial component	177
SpeedDial button	177
Disabled.....	178
cssClass.....	179
Visible.....	180
Tooltip	180
Opens on hover.....	181
Modal in Angular Speed dial component	181
Event in Angular Speed dial component.....	182
clicked	182
created	183
beforeOpen	183
onOpen	184
beforeClose	185
onClose.....	185
beforeItemRender	186
Accessibility in Angular Speed dial component	187
WAI-ARIA attributes.....	188
Keyboard interaction	189
Ensuring accessibility	189
See also	189

Spinner	189
Getting started with Angular Spinner component	189
Getting Started with Angular CLI	189
Installing Syncfusion Popups package	190
Adding CSS reference.....	191
Adding Spinner	191
Create the Spinner globally.....	192
Template in Angular Spinner component.....	193
Types in Angular Spinner component.....	196
Style in Angular Spinner component	198
Customizing the spinner	198
SplitButton	199
Getting started with Angular Split button component.....	199
Dependencies.....	199
Setup Angular environment.....	199
Create an Angular application	199
Installing Syncfusion SplitButton package	199
Adding SplitButton module.....	200
Adding Syncfusion SplitButton component	201
Adding CSS reference.....	201
Running the application	201
Icons and separator in Angular Split button component.....	202
SplitButton Icons	202
Separator.....	204
See Also	205
Popup items in Angular Split button component	205
Icons	205
Template	206
See Also	208
Accessibility in Angular Split button component.....	208
WAI-ARIA attributes.....	209
Keyboard interaction	210
Ensuring accessibility	210
See also	210
How To	210

Create right to left splitbutton in Angular Split button component	210
Group items in popup in Angular Split button component	211
Open a dialog on popup item click in Angular Split button component	212
Set the disabled state in Angular Split button component.....	214
Underline a character in a text in Angular Split button component	215
Ej1 api migration in Angular Split button component	216
Properties.....	216
Methods.....	217
Events.....	218
Splitter.....	219
Getting started with Angular Splitter component	219
Getting Started with Angular CLI	219
Installing Syncfusion Splitter package.....	220
Adding Splitter module	221
Adding Splitter component.....	222
container {	223
Adding CSS reference.....	223
Load content to the pane.....	223
Running the application	223
See Also	224
Split panes in Angular Splitter component	224
Horizontal layout.....	224
Vertical layout	225
Multiple panes	226
Separator.....	228
Nested Splitter	229
Add or remove pane	231
See Also	233
Resize in Angular Splitter component.....	233
Min and Max validation	233
Prevent resizing.....	235
Refresh content on resizing	236
Customize Resize-gripper and Cursor	236
See Also	237
Pane content in Angular Splitter component	237

Template	237
Angular UI Components	238
Plain content	238
HTML Markup	239
Pane content using selector	240
Expand collapse in Angular Splitter component	241
Collapsible panes	241
Programmatically control the expand and collapse action	243
Specify initial state to panes	244
See Also	245
Pane sizing in Angular Splitter component	245
Auto-size panes	246
Fixed pane	248
Different layouts in Angular Splitter component	249
Code editor style layout	249
Outlook style layout	251
groupedList.e-listview .e-list-group-item {	254
splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {	254
buttonSection {	254
See Also	255
Style in Angular Splitter component	255
Customizing the split bar	255
Customizing the split bar resize handle	255
Customizing the split bar arrows	256
To hide the resize handle in Splitter	257
Globalization in Angular Splitter component	258
RTL	258
See Also	259
Accessibility in Angular Splitter component	259
Keyboard interaction	260
Ensuring accessibility	260
See also	260
Ej1 api migration in Angular Splitter component	260
Common	260
Accessibility and Localization	261

Control State	262
State Maintenance	262
Pane Properties	262
Animation	264
Spreadsheet	264
Overview of the Angular Spreadsheet component	264
Key features	265
Getting started with Angular Spreadsheet component	266
Dependencies	266
Setup Angular Environment	266
Create an Angular Application	266
Installing Syncfusion Spreadsheet package	267
Registering Spreadsheet Module	267
Adding CSS reference	268
Add Spreadsheet control	268
Run the application	269
See Also	269
Data binding in Angular Spreadsheet component	270
Local data	270
Remote data	271
Cell data binding	274
Dynamic data binding and Datasource change event	275
Note	277
See Also	277
Open save in Angular Spreadsheet component	277
Open	277
Save	282
Server Configuration	288
Server Dependencies	289
Supported File Formats	289
Note	290
See Also	290
Worksheet in Angular Spreadsheet component	290
Add sheet	290
Delete sheet	291

Rename sheet	292
Headers	292
Gridlines	292
Sheet visibility	293
Note	295
See Also	295
Cell range in Angular Spreadsheet component	296
Wrap text	296
Merge cells.....	298
Data Validation.....	300
Auto Fill	304
Clear	307
Note	309
See Also	309
Editing in Angular Spreadsheet component	309
Edit cell.....	309
Save cell.....	309
Cancel editing.....	310
Limitations.....	311
Note	311
See Also	311
Formulas in Angular Spreadsheet component	312
Usage.....	312
Culture-Based Argument Separator.....	312
Create User Defined Functions / Custom Functions.....	314
Formula bar	319
Named Ranges	319
Supported Formulas.....	322
Formula Error Dialog.....	322
Note	323
See Also	323
Formatting in Angular Spreadsheet component	323
Number Formatting	323
Text and cell formatting.....	327
Conditional Formatting	331

Note	334
See Also	334
Freeze pane in Angular Spreadsheet component.....	334
Apply freezepanes on UI	334
FrozenRows	334
FrozenColumns	335
Limitations.....	336
Note	336
See Also	336
Context menu in Angular Spreadsheet component	336
Context Menu Items in Row Cell.....	336
Context Menu Items in Row Header / Column Header	337
Context Menu Items in Pager	337
Context Menu Customization	337
Note	340
See Also	340
Template in Angular Spreadsheet component	340
Note	343
See Also	343
Illustrations in Angular Spreadsheet component	344
Image.....	344
Chart.....	382
Note	387
See Also	387
Rows and columns in Angular Spreadsheet component	387
Insert	387
Delete.....	390
Limitations.....	392
Hide and show	392
Row	392
Column.....	392
Changing text in column headers	394
Note	395
See Also	395
Filter in Angular Spreadsheet component.....	395

Apply filter on UI	395
Filter by criteria	395
Filter by cell value	396
Clear filter.....	396
Clear filter on a field.....	397
Reapply filter	397
Known error validations.....	397
Limitations.....	397
Note	397
See Also	397
Sort in Angular Spreadsheet component.....	398
Sort by cell value	398
Data contains header	399
Case sensitive sort.....	399
Sort multiple columns	400
Custom sort comparer	402
Known error validations.....	402
Limitations.....	402
Note	402
See Also	402
Link in Angular Spreadsheet component.....	402
Insert Link.....	403
Edit Hyperlink.....	403
Remove Hyperlink	403
How to change target attribute	403
Limitations.....	405
Note	405
See Also	405
Clipboard in Angular Spreadsheet component.....	405
Cut	405
Copy	405
Paste.....	406
Prevent the paste functionality	407
Limitations.....	409
Note	409

Selection in Angular Spreadsheet component	409
Cell selection	410
Row selection	410
Column selection	411
How to remove selection in the spreadsheet.....	412
Limitations.....	413
Note	414
Scrolling in Angular Spreadsheet component.....	414
Finite Scrolling.....	414
Virtual Scrolling.....	414
Finite scrolling with defined rows and columns	415
Note	416
Protect sheet in Angular Spreadsheet component.....	416
Protect Sheet	416
Unprotect Sheet.....	418
Unlock the particular cells in the protected sheet.....	418
Protect Workbook.....	420
Unprotect Workbook	422
Note	422
See Also	422
Searching in Angular Spreadsheet component.....	422
Find.....	422
Replace.....	423
Go to.....	423
Limitations.....	424
Note	425
Keyboard shortcuts in Angular Spreadsheet component.....	425
Note	428
See Also	428
Ribbon in Angular Spreadsheet component.....	428
Ribbon Customization	428
Note	432
See Also	432
Global local in Angular Spreadsheet component	432
Localization	432

Internationalization.....	445
Right to left (RTL)	447
Note	448
See Also	448
Undo redo in Angular Spreadsheet component.....	448
Undo.....	448
Redo	449
Update custom actions in UndoRedo collection.....	449
Note	450
See Also	450
Accessibility in Angular Spreadsheet component.....	450
WAI-ARIA attributes.....	451
Keyboard interaction	452
Ensuring accessibility	453
See also	454
Styles in Angular Spreadsheet component.....	454
Customizing the Spreadsheet	454
Header.....	454
Sheet	455
Ribbon Items	456
Footer.....	457
Ej1 api migration in Angular Spreadsheet component.....	458
Editing	458
Selection.....	459
Clipboard.....	460
Formulas	460
Formatting.....	461
Filtering	463
Sorting.....	463
Hyperlink.....	463
Protection	464
Find and Replace.....	464
Ribbon.....	465
Undo and Redo	466
Worksheet.....	467

Open and Save	468
Data Binding.....	470
Context Menu	470
Cell Template	471
Merge.....	472
Insert and Delete.....	472
Clear	473
Data Validation.....	474
Wrap.....	475
Scrolling.....	475
Comparision between EJ1 and EJ2 Spreadsheet features	476
Common Properties	478
Common Methods	479
Common Events	481
How To	481
Sort a range by custom list in Angular Spreadsheet component	481
Print in Angular Spreadsheet component.....	483
Import an excel document using file uploader in ##Platform_Name## Spreadsheet component..	487
Mobile responsiveness in Angular Spreadsheet component	494
Note	495
Stepper.....	495
Getting started with Angular Stepper component	495
Dependencies.....	495
Setup Angular environment.....	496
Create an Angular application	496
Installing Syncfusion Stepper Package.....	496
Adding Stepper module	497
Adding CSS reference.....	497
Adding Syncfusion Stepper component.....	497
Adding Steps	498
Running the application	499
Configure icon and label	499
Steps in Angular Stepper component	502
Adding steps.....	502
Optional steps	505

Disabling steps	507
Setting active step.....	510
Step status.....	512
Step styling.....	515
Step validation	517
Step types in Angular Stepper component.....	517
Default type	517
Label type.....	520
Indicator type	525
Orientations in Angular Stepper component.....	526
Horizontal.....	526
Vertical	528
Linear Flow in Angular Stepper component	530
Steps validation in Angular Stepper component	533
Template in Angular Stepper component	536
Tooltip in Angular Stepper component	539
Tooltip template	541
Animation in Angular Stepper component.....	544
Globalization in Angular Stepper component.....	545
Localization	545
RTL.....	547
Accessibility in Angular Stepper component	550
WAI-ARIA attributes.....	551
Keyboard interaction	551
Ensuring accessibility	551
See also	551
Events in Angular Stepper component	551
created	551
stepChanged	552
stepChanging.....	553
stepClick	554
beforeStepRender.....	554
Stock Chart.....	555
Getting started with Angular Stock chart component.....	555
Setup Angular Environment.....	555

Create an Angular Application	555
Installing Syncfusion StockChart package	556
Registering StockChart Module	556
Module Injection	558
Populate Chart with Data	558
Add Stock Chart Title	560
Add Stock Chart Crosshair	561
Trackball	562
Working with data in Angular Stock chart component	564
Local Data	564
Remote Data	565
See Also	567
Chart dimensions in Angular Stock chart component	567
Size for Container	567
Size for Stock Chart	568
Axis types in Angular Stock chart component	570
DateTime axis	570
DateTimeCategory axis	571
Logarithmic axis	572
See also	573
Axis customization in Angular Stock chart component	573
Axis Crossing	573
Title	574
Tick Lines Customization	575
Grid Lines Customization	576
Multiple Axis	577
Inversed Axis	579
Opposed Position	580
Series types in Angular Stock chart component	581
Line	581
Spline	581
Area	581
Hilo	581
HiloOpenClose	581
HollowCandle	581

Candle	581
Trend lines in Angular Stock chart component.....	582
Linear.....	583
Exponential	583
Logarithmic	583
Polynomial	583
Power	583
Moving Average	583
Technical indicators in Angular Stock chart component	584
Accumulation Distribution	584
Average True Range (ATR)	584
Bollinger Band.....	584
Exponential Moving Average (EMA)	585
Momentum	585
Moving Average Convergence Divergence (MACD)	585
Relative Strength Index (RSI).....	585
Simple Moving Average (SMA)	585
Stochastic	585
Triangular Moving Average (TMA).....	585
Tool tip in Angular Stock chart component	587
Default tooltip.....	587
Format the tooltip.....	588
Position the tooltip	589
Tooltip template	590
Customize the appearance of the tooltip	591
Cross hair in Angular Stock chart component.....	593
Tooltip for axis	594
Customization	595
Legend in Angular Stock chart component.....	596
Position and Alignment.....	596
Customization	599
Collapsing Legend Item	603
Legend Title.....	604
Period selector in Angular Stock chart component	605
Periods	605

Visibility of period selector	607
Range selector in Angular Stock chart component.....	608
Visibility of range selector.....	609
Export print in Angular Stock chart component	610
Disable Export and print	611
Appearance in Angular Stock chart component	612
Stock Chart Title	612
Stock Chart Theme	614
See Also	615
Stock events in Angular Stock chart component	615
Accessibility in Angular Stock chart component.....	622
WAI-ARIA attributes.....	623
Keyboard interaction	624
Ensuring accessibility	624
See also	624
Internationalization in Angular Stock chart component.....	624
Localization in Angular Stock chart component	626
How To	628
Live data in Angular Stock chart component	628
Getting started with Angular Switch component	631
Dependencies.....	631
Setup Angular environment.....	631
Create an Angular application	631
Installing Syncfusion Switch package.....	631
Adding Switch module	632
Adding Syncfusion Switch component.....	632
Adding CSS reference.....	633
Running the application	633
Set text on Switch	634
Accessibility in Angular Switch component	634
WAI-ARIA attributes.....	635
Keyboard interaction	635
Ensuring accessibility	636
See also	636
How To	636

Bind data using two way binding in Angular Switch component	636
Change size in Angular Switch component	638
Customize the appearance of a switch in Angular Switch component	639
Enable ripple for switch label in Angular Switch component	644
Enable rtl in Angular Switch component	645
Set disabled state in Angular Switch component	646
Submit name and value in form in Angular Switch component	647
Change switch state using toggle method in Angular Switch component	648
Tab	649
Getting started with Angular Tab component	649
Dependencies	649
Setup Angular Environment	649
Create an Angular Application	649
Installing Syncfusion Tab Package	649
Registering Tab Module	650
Adding CSS reference	651
Add Tab component	651
Initialize the Tab using JSON items collection	651
Run the application	653
Initialize the Tab using template	654
Initialize the Tab using HTML elements	657
See Also	658
Adaptive in Angular Tab component	658
Scrollable	659
Popup	662
See Also	665
Header in Angular Tab component	665
Styles	665
Icon positions	668
See Also	670
Localization in Angular Tab component	671
Loading translations	671
Orientation in Angular Tab component	672
Drag and drop in Angular Tab component	676
Drag and drop item between tabs	678

Drag and drop items to external source	681
Drag and drop items from external source.....	684
Accessibility in Angular Tab component.....	687
ARIA attributes.....	688
Keyboard interaction	688
Ensuring accessibility	689
See also	689
Style in Angular Tab component.....	689
Customizing Tab.....	689
Customizing the Tab items.....	689
Customizing Tab's header	690
Customizing Tab's header icon	690
Customizing Tab's content.....	690
Customizing the hover state of Tab control.....	690
Customizing selected item of Tab control	691
How To	691
Load content through post in Angular Tab component	691
Prevent content swipe selection in Angular Tab component.....	693
Customize selected tab styles in Angular Tab component	694
Customize tab scroll step in Angular Tab component	696
Create wizard using tab in Angular Tab component.....	698
Load tab with data source in Angular Tab component.....	709
Add nested tabs in Angular Tab component	710
Set state persistence of the tab component in Angular Tab component.....	714
Set custom animation in Angular Tab component	716
Disable default tab animation effects in Angular Tab component.....	718
Load tab items dynamically in Angular Tab component.....	720
Create collapsible tabs in Angular Tab component	721
Render other components in tab using angular template in Angular Tab component.....	724
Add reactive forms within a tab in Angular Tab component.....	726
Adding dynamic items with content reuse in Angular Tab component	731
Customize tab content height in Angular Tab component.....	733
Reorder active tab in Angular Tab component.....	736
Tab selection in Angular Tab component	739
Show hide tab item in Angular Tab component	742

Enabling tab key navigation in Tabs.....	744
Ej1 api migration in Angular Tab component	747
Accessibility and Localization.....	747
AjaxSettings.....	747
Animation.....	748
Header.....	749
Items	749
Common.....	751
TextArea.....	753
Getting started with Angular TextArea Component.....	753
Dependencies.....	753
Setup angular environment	753
Create a new application	753
Installing syncfusion TextArea package	753
Adding TextArea to the application	754
Adding CSS reference.....	755
Running the application	755
Getting and setting values	756
Floating Label in Angular TextArea Component	758
Placeholder with localization	759
Rows and Columns in Angular TextArea Component.....	760
Resize in Angular TextArea Component	762
Width of angular TextArea component	763
Maximum Length in Angular TextArea Component	764
Form Support in Angular TextArea Component	765
Integration of angular TextArea component with FormValidator component	767
Sizing in Angular TextArea Component	768
Filled and outline mode	769
Custom styling with cssClass API in TextArea	770
Setting the disabled state in TextArea	771
Set the readonly TextArea	772
Set the rounded corner in TextArea	772
Static clear button in TextArea	773
Customize the TextArea background color and text color	773
Change the floating label color of the TextArea	774

Adding mandatory asterisk to placeholder.....	776
Events in Angular TextArea Component.....	777
Created event.....	777
Input event.....	777
Change event	778
Focus event	779
Blur event.....	779
Destroyed event.....	780
Methods in Angular TextArea Component	780
FocusIn method	780
FocusOut method	781
GetPersistData method.....	783
TextBox	784
Getting started with Angular Textbox component.....	784
Dependencies.....	784
Setup angular environment	784
Create a new application	784
Installing Syncfusion TextBox Package.....	785
Adding TextBox to the application.....	785
Adding CSS reference.....	786
Adding icons to the TextBox	786
Running the application.....	787
Floating label.....	788
See Also	789
Groups in Angular Textbox component.....	789
With icon and floating label	790
With clear button and floating label	793
Floating Label without required attribute	793
Multi-line Input with Floating Label.....	794
See Also	795
Sizing in Angular Textbox component	795
Validation in Angular Textbox component	797
Adding mandatory asterisk to placeholder and float label.....	798
Multiline in Angular Textbox component	798
Create multiline textbox	798

Implementing floating label	799
Auto resizing	800
Disable resizing	801
Limit the text length.....	802
Count characters.....	803
Accessibility in Angular Textbox component.....	804
WAI-ARIA attributes.....	805
Ensuring accessibility	805
See also	805
Style appearance in Angular Textbox component.....	805
Customizing the appearance of TextBox wrapper element	805
Customizing the TextBox placeholder	806
Toggle password visibility using eye icon	806
How To	807
Set the rounded corner in Angular Textbox component.....	807
Set the disabled state in Angular Textbox component.....	808
Set the read only textbox in Angular Textbox component.....	808
Add textbox programmatically in Angular Textbox component.....	809
Add floating label to textbox programmatically in Angular Textbox component	810
Change the floating label color of the textbox in Angular Textbox component	812
Change the color of the textbox based on its value in Angular Textbox component.....	813
Add floating label to read only textbox in Angular Textbox component.....	814
Customize the textbox background color and text color in Angular Textbox component.....	816
Migration from css textbox to angular textbox in Angular Textbox component	817
Normal textbox	817
Floating label textbox.....	817
Timeline.....	818
Getting started with Angular Timeline component.....	818
Dependencies.....	818
Setup Angular environment.....	818
Create an Angular application	818
Installing Syncfusion Timeline Package.....	818
Adding Timeline module	819
Adding CSS reference.....	820
Adding Syncfusion Timeline component	820

Adding Items	820
Running the application	821
Items in Angular Timeline component.....	822
Adding content.....	822
Adding opposite content	825
Dot item	827
Disabling items	828
cssClass.....	829
Orientations in Angular Timeline component	829
Vertical	829
Horizontal.....	831
Alignment in Angular Timeline component.....	832
Before.....	832
After	833
Alternate	834
Alternate reverse	836
Reverse in Angular Timeline component.....	837
Template in Angular Timeline component	838
Customization in Angular Timeline component	841
Connector styling	841
Dot styling	844
Accessibility in Angular Timeline component	850
WAI-ARIA attributes.....	851
Ensuring accessibility	851
See also	852
Events in Angular Timeline component.....	852
created	852
beforeItemRender	853
TimePicker.....	854
Getting started with Angular Timepicker component	854
Dependencies.....	854
Setup Angular environment.....	855
Create a new application	855
Installing Syncfusion TimePicker package.....	855
Registering TimePicker module	856

Adding CSS reference.....	857
Adding TimePicker component.....	857
Running the application	858
Setting the selected, min, and max time	858
Setting the time format	859
See Also	860
Time range in Angular Timepicker component.....	860
Time masking in Angular Timepicker component	861
Configure Mask Placeholder	863
Globalization in Angular Timepicker component	864
Right-To-Left	867
Strict mode in Angular Timepicker component.....	869
Accessibility in Angular Timepicker component.....	871
WAI-ARIA attributes.....	872
Keyboard Interaction	872
Ensuring accessibility	873
See also	874
Style appearance in Angular Timepicker component.....	874
Customizing the appearance of TimePicker wrapper element	874
Customizing the TimePicker icon element.....	874
Customizing the TimePicker popup	874
Customizing the TimePicker popup content.....	875
Full screen mode support in mobiles and tablets.....	875
How To	877
Json data binding with timepicker in Angular Timepicker component	877
Two way binding in Angular Timepicker component	878
Css customization in Angular Timepicker component.....	879
Custom event emitter in Angular Timepicker component	880
Custom validation using form validator in Angular Timepicker component.....	881
Reactive form in Angular Timepicker component	882
Template driven forms in Angular Timepicker component.....	884
Ej1 api migration in Angular Timepicker component	885
Time Selection.....	886
Time Format.....	886
Time Range.....	886

Disabled Time Ranges	886
Customization	886
Accessibility	888
Persistence	888
Validation	888
Common	889
Globalization	890
Strict Mode	890
Open and Close	890
Toast	891
Getting started with Angular Toast component	891
Getting Started with Angular CLI	891
Installing Syncfusion notifications Package	892
Adding Toast module	893
Adding Toast component	894
Adding CSS reference	894
Initialize the Toast with message	895
Running the application	895
See Also	896
Config in Angular Toast component	896
Title and content template	896
Specifying custom target	897
Close Button	897
Progress bar	897
Newest on top	897
Width and height	898
See Also	901
Position in Angular Toast component	901
Predefined	901
Custom	901
See Also	905
Action buttons in Angular Toast component	906
See Also	907
Timeout in Angular Toast component	907
Static toast	908

See Also	909
Template in Angular Toast component	909
See Also	914
Animation in Angular Toast component.....	914
Toast services in Angular Toast component	916
Show Toast with predefined types	916
Show Toast with ToastModel	917
Style in Angular Toast component.....	918
Customizing the toast title.....	918
Customizing the toast content.....	919
Customizing the toast icon.....	919
Customizing the toast background	919
Accessibility in Angular Toast component	920
WAI-ARIA attributes.....	921
ARIA attributes.....	921
Ensuring accessibility	922
See also	922
How To	922
Prevent duplicate toast display in Angular Toast component.....	922
Restrict the maximum toast to show in Angular Toast component	924
Customize progress bar theme and sizing in Angular Toast component	925
Play an audio before open the toast in Angular Toast component.....	927
Show different types of toast in Angular Toast component.....	928
Show multiple toasts in various positions in Angular Toast component.....	930
Close the toast with click tap in Angular Toast component	931
Add dynamic template in Angular Toast component	932
Render template in toast using angular template in Angular Toast component	933
Prevent toast close with mobile swipe in Angular Toast component	934

Signature

Getting started with Angular Signature component

This section explains how to create a simple Signature, and demonstrate the basic usage of the Signature module in an Angular environment.

Dependencies

The list of dependencies required to use the Signature module in your application is given below:

```
`javascript
|-- @syncfusion/ej2-angular-inputs
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-base
`,`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`
npm install -g @angular/cli
`,`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`
ng new my-app
cd my-app
`,`
```

Installing Syncfusion Signature Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-inputs](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-inputs@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-inputs:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Signature module

Import Signature module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-inputs`.

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Imported Syncfusion Signature module from inputs package.
import { SignatureModule } from '@syncfusion/ej2-angular-inputs';
import { AppComponent } from './app.component';

@NgModule({
  imports: [BrowserModule, SignatureModule], // Registering EJ2 Signature Module.
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
`
```

Adding Syncfusion Signature component

Modify the template in `app.component.ts` file with `ejs-signature` to render the Signature component.


```
`javascript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render Signature. -->
<canvas ej2-signature #signature id="signature" ></canvas>`
})
export class AppComponent {}
`
```

Adding CSS reference

Add Signature component's styles as given below in `style.css`.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
`
```

Running the application

Run the application in the browser using the following command:

```
ng serve
```

The following example shows a basic `Signature` component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Sign here</h4>
    <!-- To Render Signature. -->
    <canvas ej2-signature #signature id="signature"></canvas></div>`
})
export class AppComponent {}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customization in Angular Signature component

The Signature component draws stroke/path using `moveTo()` and `lineTo()` methods to connect one or more points while drawing in canvas. The stroke width can be modified by using its color and width. And the background can be modified by using its background color and background image.

Stroke Width

The variable stroke width is based on the values of [maxStrokeWidth](#), [minStrokeWidth](#) and [velocity](#) for smoother and realistic signature. The default value of minimum stroke width is set as 0.5, maximum stroke width is set as 2.5 and velocity is set as 0.7.

In the following example, minimum stroke width is set as 0.5, maximum stroke width is set as 3 and velocity is set as 0.7.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Sign here</h4>
    <!-- To Render Signature. -->
    <canvas ej2-signature #signature id="signature"
    [maxStrokeWidth]="3" [minStrokeWidth]="0.5"
    [velocity]="0.7"></canvas></div>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Stroke Color

Color of the stroke can be specified by using [strokeColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#000000”.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="input">
      <input type="text" id="text" placeholder="Enter the Stroke
Color Value" >
      <button ejs-button cssClass="e-primary"
(click)="setColor()">Set Stroke Color</button>
    </div>
    <div id="signature-control">
      <canvas ejs-signature #signature id="signature"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  setColor(): void {
    let color = (document.getElementById('text') as any).value;
    this.signature!.strokeColor = color;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Background Color

Background color of a signature can be specified by using [backgroundColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#ffffff”.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="input">
      <input type="text" id="text" placeholder="Enter the
Background Color Value" >
      <button ejs-button cssClass="e-primary"
(click)="setBgColor()">Set Background Color</button>
    </div>
    <div id="signature-control">
      <canvas ejs-signature #signature id="signature"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  setBgColor(): void {
    let bgColor = (document.getElementById('text') as any).value;
    this.signature!.backgroundColor = bgColor;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Background Image

Background image of a signature can be specified by using [backgroundImage](#) property. The background image can be set by either hosting the image in our local IIS or online image.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'

```

```
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="input">
      <input type="text" id="text" placeholder="Enter the URL of
the background Image" >
      <button ejs-button cssClass="e-primary"
(click)="setBgImage()">Set Background Image</button>
    </div>
    <div id="signature-control">
      <canvas ejs-signature #signature id="signature"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  setBgImage(): void {
    let bgImage = (document.getElementById('text') as any).value;
    this.signature!.backgroundImage = bgImage;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Save With Background](#)

Open save in Angular Signature component

The Signature component supports to open the signature by using hosted/online URL or base64. And it also supports various save options like image, base64, and blob.

Open Signature

The signature component opens a pre-drawn signature as either base64 or hosted/ online URL using the [load](#) method. It supports the PNG, JPEG, and SVG image's base64.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="input">
      <input type="text" id="text" placeholder="Enter the Base64
or URL of signature" >
      <button ejs-button cssClass="e-btn e-primary"
(click)="open()">Open</button>
    </div>
    <div id="signature-control">
      <canvas ejs-signature #signature id="signature"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  open(): void {
    let sign = (document.getElementById('text') as any).value;
    this.signature!.load(sign);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Save Signature

The Signature component saves the signature as base64, blob, and image like PNG, JPEG, and SVG.

Save as Base64

The `getSignature` method is used to get the signature as base64 with the PNG, JPEG, and SVG type. This can be loaded to signature using [load](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'

```

```

import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
enableRipple(true);
@Component({
  imports: [
    FormsModule, DialogModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <h4>Sign here</h4>
    <div id="signature-control">
      <canvas ej2-signature #signature id="signature"></canvas>
    </div>
    <button ej2-button id="save" cssClass="e-primary"
(click)="onSave()">Save as Base64</button>
    <ejs-dialog #dialog header="Base64 of the signature"
[animationSettings]="animationSettings" showCloseIcon='true' width="80%"
visible="false"></ejs-dialog></div>`
  })
  export class AppComponent {
    @ViewChild('signature')
    public signature?: SignatureComponent;
    @ViewChild('dialog')
    public dialog?: DialogComponent;
    public animationSettings: Object = { effect: 'Zoom', duration: 400 };
    onSave() {
      this.dialog!.content = this.signature!.getSignature();
      this.dialog!.show();
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Save as Blob

The [saveAsBlob](#) method is used to save the signature as Blob. It is defined as the chunk of binary data stored as a single entity in a database system.

Save As Image

The [save](#) method is used to save the signature as an image. And it accepts file name and file type as parameter. The file type parameter supports PNG, JPEG, and SVG and the default file type is PNG.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { Signature, SignatureFileType } from '@syncfusion/ej2-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, SplitButtonModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div>
      <span>Sign here</span>
      <ejs-splitbutton content="Save" [items]='items' iconCss="e-sign-
icons e-save" (select)="onSelect($event)"></ejs-splitbutton>
    </div>
    <div id="signature-control">
      <canvas ej-signature #signature id="signature"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  public items: ItemModel[] = [
    { text: 'Png' },
    { text: 'Jpeg' },
    { text: 'Svg' }
  ];
  onSelect(args: MenuEventArgs) {
    this.signature?.save(args.item.text as SignatureFileType,
'Signature');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Save With Background

The [saveWithBackground](#) property is used to save the signature with its background and its default value is true. So, by default the signature is saved with its background.

In the following sample, the background color is set as 'rgb(103 58 183)' and save with background as true.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { Signature, SignatureFileType } from '@syncfusion/ej2-inputs';
enableRipple(true);
@Component({
  imports: [
    FormsModule, SplitButtonModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div>
      <span>Sign here</span>
      <ejs-splitbutton content="Save" [items]='items' iconCss="e-sign-icons e-save" (select)="onSelect($event)"></ejs-splitbutton>
    </div>
    <div id ="signature-control">
      <canvas ej2-signature #signature id="signature"
background-color="rgb(103 58 183)" saveWithBackground="true"></canvas>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('signature')
  public signature?: SignatureComponent;
  public items: ItemModel[] = [
    { text: 'Png' },
    { text: 'Jpeg' },
    { text: 'Svg' }
  ];
  onSelect(args: MenuEventArgs) {
    this.signature?.save(args.item.text as SignatureFileType,
'Signature');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Draw in Angular Signature component

Draw

The [draw](#) method is used to draw a text as signature with different font families like Arial, Serif, with different font sizes. It accepts text, font family, font size as its parameters. The default font family is "Arial", and the default font size is "30".

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
@Component({
  imports: [
    FormsModule, DropDownListModule, ButtonModule, SignatureModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="signature-control">
      <canvas ej2-signature #signature id="signature"></canvas>
    </div>
    <div id="input">
      <table>
        <tbody>
          <tr>
            <td><div>Enter the Text:</div></td>
            <td><input type="text" id="text" placeholder="Enter
the Text"></td>
          </tr>
          <tr>
            <td><div>Font Family:</div></td>
            <td><ejs-dropdownlist id='font' #font
[dataSource]='fontItems' [value]='fontValue' [fields]='fontfields'
[popupHeight]='height'></ejs-dropdownlist>
            </td>
          </tr>
          <tr>
            <td><div>Font Size:</div></td>
            <td><ejs-dropdownlist id='size' #size
[dataSource]='sizeData' [value]='sizeValue' [fields]='sizefields'
[popupHeight]='height'></ejs-dropdownlist>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  `
})
```

```

        <br>
        <button ejs-button cssClass="e-primary"
(click)="onDraw()" ">Draw</button>
    </div>
</div>`
}))
export class AppComponent {
    @ViewChild('signature')
    public signature?: SignatureComponent;
    @ViewChild('font')
    public font?: DropDownListComponent;
    @ViewChild('size')
    public size?: DropDownListComponent;
    public fontValue: string = 'Arial';
    public sizeValue: number = 20;
    public height: string = '200px';
    public fontItems: Object[] = [
        { value: 'Arial' },
        { value: 'Serif' },
        { value: 'Sans-serif' },
        { value: 'Cursive' },
        { value: 'Fantasy' }
    ];
    public fontfields: Object = { text: 'value', value: 'value' };
    public sizeData: Object[] = [
        { value: '20' },
        { value: '30' },
        { value: '40' },
        { value: '50' }
    ];
    public sizefields: Object = { text: 'value', value: 'value' };
    onDraw(): void {
        let text = (document.getElementById('text') as any).value;
        this.signature!.draw(text, (this.font as any).value, (this.size as
any).value);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

User interaction in Angular Signature component

The below interactions were available in Signature, and we can walk through one by one.

- Undo and Redo
- Clear
- Disabled
- ReadOnly

Undo

In the Signature, every action can be maintained as a snap for undo and redo operations. And maintained SnapIndex for indexing the snap collection.

The [undo](#) method reverts the last action of signature by decreasing SnapIndex value to index previous snap. Here, [canUndo](#) method is used to ensure whether undo can be performed or not.

The [redo](#) method reverts the last undo action of the signature by increasing the SnapIndex to get the next snap. Here, [canRedo](#) method is used to ensure whether redo can be performed or not.

Clear

The [clear](#) method is used to clear the signature and makes the canvas empty. This is also considered in Undo/ Redo. Here, [isEmpty](#) method is used to ensure whether the signature is empty or not.

Disabled

The [disabled](#) property is used to enable/disable the signature component. In the disabled state, the user is not allowed to draw signature. And it can't be focused until the user enabled the signature.

ReadOnly

The [isReadOnly](#) property is used to enable/disable the ReadOnly Signature. It can be focused but it prevents drawing in Signature.

The following sample explains about user interactions available in signature.

User Integration sample

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs'
import { CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { Button, ChangeEventArgs } from '@syncfusion/ej2-buttons';
import { getComponent } from '@syncfusion/ej2-base';
enableRipple(true);
@Component({
  imports: [
    FormsModule, CheckBoxModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="option">
      <table>
        <tr>
          <td>
            <button ej2-button id="undo" #undo cssClass="e-
primary" (click)="onUndo()" disabled={true}>UNDO</button>
          </td><td>
            <button ej2-button id="redo" #redo cssClass="e-
primary" (click)="onRedo()" disabled={true}>REDO</button>
          </td></tr>
        </table>
      </div>
    </div>`
})
```

```

        </td><td>
            <button ejs-button id="clear" #clear cssClass="e-
primary" (click)="onClear()" disabled={true}>CLEAR</button>
        </td><td>
            <div id="checkboxbox1"><ejs-checkbox label="Disable"
(change)="onDisableChange($event)"></ejs-checkbox></div>
            <div id="checkboxbox1"><ejs-checkbox label="ReadOnly"
(change)="onReadOnlyChange($event)"></ejs-checkbox></div>
        </td>
    </tr>
</table>
</div>
<div id="signature-control">
    <canvas ejs-signature #signature id="signature"
(change)="change()"></canvas>
</div>
</div>`
}))
export class AppComponent {
    @ViewChild('signature')
    public signature?: SignatureComponent;
    onUndo(): void {
        if (!this.signature.disabled && !this.signature.isReadOnly) {
            this.signature.undo();
        }
    }
    onRedo(): void {
        if (!this.signature.disabled && !this.signature.isReadOnly) {
            this.signature.redo();
        }
    }
    onClear(): void {
        if (!this.signature.disabled && !this.signature.isReadOnly) {
            this.signature.clear();
        }
    }
    onDisableChange(args: ChangeEventArgs): void {
        this.signature.disabled = args.checked;
    }
    onReadOnlyChange(args: ChangeEventArgs): void {
        this.signature.isReadOnly = args.checked;
    }
    change(): void {
        let undoButton: Button =
getComponent(document.getElementById("undo"), 'btn');
        let redoButton: Button =
getComponent(document.getElementById("redo"), 'btn');
        let clearButton: Button =
getComponent(document.getElementById("clear"), 'btn');
        if (!this.signature.disabled && !this.signature.isReadOnly) {
            if (this.signature.canUndo()) {
                undoButton.disabled = false;
            } else {
                undoButton.disabled = true;
            }
            if (this.signature.canRedo()) {
                redoButton.disabled = false;
            }
        }
    }
}

```

```

    } else {
      redoButton.disabled = true;
    }
    if (!this.signature.isEmpty()) {
      clearButton.disabled = false;
    } else {
      clearButton.disabled = true;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

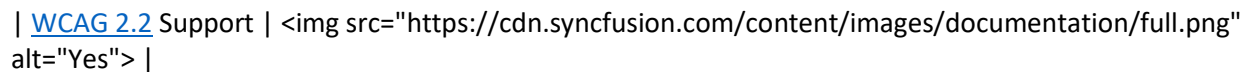
Accessibility in Angular Signature component

The Signature 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 Signature component is outlined below.

| Accessibility Criteria | Compatibility |

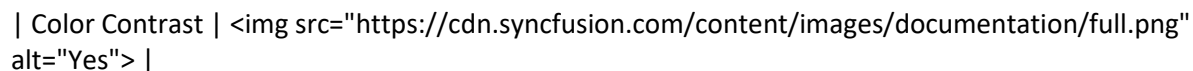
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support | Not Applicable |

| 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 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 Signature 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 Signature component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + Z** | **Undo the last action.** |

| **Ctrl + Y** | **Redo the last action.** |

| **Ctrl + S** | **To save the signature.** |

| **delete** | **Erases all the signature strokes signed by user.** |

Ensuring accessibility

The Signature component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Signature component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Signature component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Integrate toolbar in Angular Signature component

The Signature component integrates with the toolbar and the interaction performed using the `change` event of the toolbar.

In that, [canUndo](#), [canRedo](#) and [isEmpty](#) methods were used to enable/disable undo, redo, and clear buttons.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { ColorPickerModule, SignatureModule } from '@syncfusion/ej2-angular-inputs'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { ToolbarModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, ViewChild } from '@angular/core';
import { addClass, createElement, getComponent } from '@syncfusion/ej2-base';
import { SignatureComponent } from '@syncfusion/ej2-angular-inputs';
import { SplitButton, ItemModel, MenuEventArgs, DropDownButton } from '@syncfusion/ej2-splitbuttons';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { Button, ChangeEventArgs, CheckBox } from '@syncfusion/ej2-buttons';
import { ColorPicker, ColorPickerEventArgs, NumericTextBox, PaletteTileEventArgs, Signature, SignatureFileType } from '@syncfusion/ej2-inputs';
import { DropDownList } from '@syncfusion/ej2-angular-dropdowns';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
@Component({
  imports: [
    FormsModule, DropDownListModule, SplitButtonModule, ToolbarModule,
    ButtonModule, SignatureModule, ColorPickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id="signature-toolbar-control">
      <ejs-toolbar id="toolbar" (created)="onCreate($event)"
(clicked)="clicked($event)" width="100%">
        <e-items>
          <e-item text='Undo' prefixIcon='e-icons e-undo'
tooltipText='Undo (Ctrl + Z)'></e-item>
          <e-item text='Redo' prefixIcon='e-icons e-redo'
tooltipText='Redo (Ctrl + Y)'></e-item>
          <e-item type='Separator'></e-item>
          <e-item tooltipText= 'Save (Ctrl + S)' type='Button'
template= '<button id="save-option"></button>'>
            </e-item>
          <e-item type='Separator'></e-item>
          <e-item tooltipText= 'Stroke Color' template= '<input
id="stroke-color" type="color"/>'>
            </e-item>
          <e-item type='Separator'></e-item>
          <e-item tooltipText= 'Background Color' type= 'Input'
template= '<input id="bg-color" type="color"/>'></e-item>
          <e-item type='Separator'></e-item>
          <e-item tooltipText= 'Stroke Width' type= 'Input' template=
'<input id="stroke-width" type="text"/>'></e-item>
          <e-item type='Separator'></e-item>
          <e-item text= 'Clear' prefixIcon= 'e-sign-icons e-clear'
tooltipText= 'Clear'></e-item>
        </e-items>
      </div>
    </div>
  `
})

```



```

        <e-item tooltipText= 'Disabled' type='Input'
template='<input id="chkelement1" type="checkbox"/>' align='Right'></e-item>
        </e-items>
    </ejs-toolbar>
    <div id="signature-control">
        <canvas ej-s-signature #signature id="signature"
[maxStrokeWidth]="strokeWidth" style="height: 100%; width: 100%;"
(change)="change()" "></canvas>
    </div>
</div>
</div>`
    })
    export class AppComponent {
        @ViewChild('signature') signature?: SignatureComponent;
        public strokeWidth: number = 2;
        public items: ItemModel[] = [
            {
                text: 'Png'
            },
            {
                text: 'Jpeg'
            },
            {
                text: 'Svg'
            }
        ];
        public templateCheckbox1: any = new CheckBox({
            label: 'Disabled',
            checked: false,
            change: (args: ChangeEventArgs) => {
                (this.signature as SignatureComponent).disabled = args.checked
as boolean;
            }
        });
        public change(): void {
            let saveBtn: SplitButton =
getComponent((document.getElementById("save-option") as HTMLElement),
'split-btn');
            if (!this.signature?.isEmpty()) {
                this.clearButton();
                saveBtn.disabled = false;
            }
            this.updateUndoRedo();
        }
        public onCreate (e: any) {
            this.templateCheckbox1.appendTo('#chkelement1');
            let ddl: DropDownList = new DropDownList({
                dataSource: [1, 2, 3, 4, 5],
                width: '60',
                value: 2,
                change: function(args) {
                    let signature: Signature =
getComponent((document.getElementById("signature") as HTMLElement),
'signature');
                    signature.maxStrokeWidth = args.value;
                }
            });
            ddl.appendTo('#stroke-width');

```

```

    new SplitButton({
      iconCss: 'e-sign-icons e-save',
      items: this.items,
      content: 'Save',
      select: (args: MenuEventArgs) => {
        this.signature?.save(args.item.text as SignatureFileType,
'Signature');
      },
      disabled: true
    }, '#save-option');
    let strokeColor: ColorPicker = new ColorPicker({
      modeSwitcher: false,
      columns: 4,
      presetColors: {
        'custom': ['#000000', '#e91e63', '#9c27b0', '#673ab7',
'#2196f3', '#03a9f4', '#00bcd4',
'#009688', '#8bc34a', '#cddc39', '#ffeb3b', '#ffc107']
      },
      beforeTileRender: (args: PaletteTileEventArgs) => {
        args.element.classList.add('e-circle-palette');
        args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
      },
      showButtons: false,
      mode: 'Palette',
      cssClass: 'e-stroke-color',
      change: (args: ColorPickerEventArgs) => {
        if (this.signature?.disabled) {
          return;
        }
        let selElem: HTMLElement = (strokeColor as
any).element.nextElementSibling.querySelector('.e-selected-color') as
HTMLElement;
        selElem.style.borderBottomColor = args.currentValue.rgba;
        this.signature!.strokeColor = args.currentValue.rgba;
      }
    });
    strokeColor.appendTo('#stroke-color');
    let bgColor: ColorPicker = new ColorPicker({
      modeSwitcher: false,
      columns: 4,
      noColor: true,
      presetColors: {
        'custom': ['#ffffff', '#f44336', '#e91e63', '#9c27b0',
'#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
'#009688', '#8bc34a', '#cddc39', '#ffeb3b']
      },
      beforeTileRender: (args: PaletteTileEventArgs) => {
        args.element.classList.add('e-circle-palette');
        args.element.appendChild(createElement('span', { className:
'e-circle-selection' }));
      },
      showButtons: false,
      mode: 'Palette',
      cssClass: 'e-bg-color',
      change: (args: ColorPickerEventArgs) => {
        if (this.signature!.disabled) {

```

```

        return;
    }
    let selElem: HTMLElement = (bgColor.element as
HTMLInputElement | any).nextElementSibling.querySelector('.e-selected-
color') as HTMLElement;
    this.signature!.backgroundColor = args.currentValue.rgba;
    selElem.style.borderBottomColor = args.currentValue.rgba;
}
});
bgColor.appendTo('#bg-color');
addClass([(strokeColor.element as
any).nextElementSibling.querySelector('.e-selected-color')], 'e-sign-
icons');
addClass([(bgColor.element as
any).nextElementSibling.querySelector('.e-selected-color')], 'e-sign-
icons');
(document.getElementById('save-option') as
HTMLElement).addEventListener('click', this.saveBtnClick);
this.clearButton();
let toolbarItems: NodeListOf<Element> =
document.querySelectorAll('.e-toolbar .e-toolbar-items .e-toolbar-item .e-
tbar-btn.e-tbtn-txt');
for (var i = 0; i < toolbarItems.length; i++) {
    if (toolbarItems[i].children[0].classList.contains('e-undo')) {
        let undoButton: Button = getComponent(toolbarItems[i] as
HTMLElement, 'btn');
        undoButton.disabled = true;
    }
    if (toolbarItems[i].children[0].classList.contains('e-redo')) {
        let redoButton: Button = getComponent(toolbarItems[i] as
HTMLElement, 'btn');
        redoButton.disabled = true;
    }
}
}
public clicked(args: ClickEventArgs): void {
    let saveBtn: SplitButton =
getComponent((document.getElementById("save-option") as HTMLElement),
'split-btn');
    if (this.signature?.disabled && args.item.tooltipText != 'Disabled')
    {
        return;
    }
    switch (args.item.tooltipText) {
        case 'Undo (Ctrl + Z)':
            if (this.signature?.canUndo()) {
                this.signature.undo();
                this.updateUndoRedo();
                this.updateSaveBtn();
            }
            break;
        case 'Redo (Ctrl + Y)':
            if (this.signature?.canRedo()) {
                this.signature.redo();
                this.updateUndoRedo();
                this.updateSaveBtn();
            }
    }
}

```

```

        break;
    case 'Clear':
        this.signature?.clear();
        if (this.signature?.isEmpty()) {
            this.clearButton();
            saveBtn.disabled = true;
        }
        break;
    }
}

public saveBtnClick(): void {
    let signature: Signature =
getComponent((document.getElementById("signature") as HTMLElement),
'signature');
    signature.save();
}

public clearButton() {
    let tlItems: NodeListOf<Element> = document.querySelectorAll('.e-
toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
    for (var i = 0; i < tlItems.length; i++) {
        if (tlItems[i].children[0].classList.contains('e-clear')) {
            let clrBtn: Button = getComponent(tlItems[i] as HTMLElement,
'btn');

            if (this.signature?.isEmpty()) {
                clrBtn.disabled = true;
            } else {
                clrBtn.disabled = false;
            }
        }
    }
}

public updateSaveBtn() {
    let saveBtn: SplitButton =
getComponent((document.getElementById("save-option") as HTMLElement),
'split-btn');
    if (this.signature?.isEmpty()) {
        saveBtn.disabled = true;
    }
}

public updateUndoRedo() {
    let undoButton: Button; let redoButton: Button
    let tlItems: NodeListOf<Element> = document.querySelectorAll('.e-
toolbar .e-toolbar-items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
    for (var i = 0; i < tlItems.length; i++) {
        if (tlItems[i].children[0].classList.contains('e-undo')) {
            undoButton = getComponent(tlItems[i] as HTMLElement, 'btn');
        }
        if (tlItems[i].children[0].classList.contains('e-redo')) {
            redoButton = getComponent(tlItems[i] as HTMLElement, 'btn');
        }
    }
    if (this.signature?.canUndo()) {
        undoButton!.disabled = false;
    } else {
        undoButton!.disabled = true;
    }
    if (this.signature?.canRedo()) {

```

```
        redoButton!.disabled = false;
    } else {
        redoButton!.disabled = true;
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Skeleton

Getting started with Angular Skeleton component

This section explains how to create a simple Skeleton and demonstrate the basic usage of the Skeleton component in an Angular environment.

Dependencies

The list of dependencies required to use the Skeleton component in your application is given as follows:

```
`js
|-- @syncfusion/ej2-angular-notifications
|-- @syncfusion/ej2-angular-base
`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`
npm install -g @angular/cli
`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`
ng new my-app
cd my-app
`
```

Installing Syncfusion Notifications package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-notifications](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-notifications --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-notifications@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-notifications@ngcc --save
`
```

To mention the `ngcc` package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-notifications:"20.3.0.47-ngcc"
`
```

Note: If the `ngcc` tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Skeleton module

After installing the package, the component modules are ready to configure in your application from installed syncfusion package. Syncfusion Angular package provides two different types of `ngModules`.

Refer to [Ng-Module](#) to learn about `ngModules`.

Refer the following snippet to import the `SkeletonModule` in `app.module.ts` from the `@syncfusion/ej2-angular-notifications`.

```
`javascript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
// Imported syncfusion SkeletonModule from ej2-angular-notifications package
```

```
import { SkeletonModule } from '@syncfusion/ej2-angular-notifications';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    // Registering EJ2 Skeleton Module
    SkeletonModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
`
```

Add Skeleton into application

Modify the `template` in `app.component.ts` file to render the Skeleton component.

[src/app/app.component.ts]

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: <ejs-skeleton height='15px'></ejs-skeleton>
})
export class AppComponent { }
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder. This can be referenced in [src/styles.css] using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-notifications/styles/material.css';
`
```

Running the application

Run the application in the browser using the following command:

`

ng serve

`

The following example shows a basic Skeleton component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SkeletonModule } from '@syncfusion/ej2-angular-notifications'
import { Component } from '@angular/core';
@Component({
  imports: [
    SkeletonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `<ejs-skeleton height='15px'></ejs-skeleton>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Shapes in Angular Skeleton component

The Skeleton control support various built-in shape variants to design layout of the page. You can use the [shape](#) property to create a preview of any layout.

The Skeleton component supports the following content shapes:

Circle skeleton shape

`typescript

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
  selector: 'my-app',
```

```
  template: <ejs-skeleton shape= 'Circle' width= "48px"></ejs-skeleton>
```

```
})
```

```
export class AppComponent { }
```

`

Square skeleton shape

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: <ejs-skeleton shape= 'Square' width= "48px"></ejs-skeleton>
})
export class AppComponent { }
,
```

Rectangle skeleton shape

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: <ejs-skeleton shape= 'Rectangle' width= "50px" height="25px"></ejs-skeleton>
})
export class AppComponent { }
,
```

Text skeleton shape

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: <ejs-skeleton shape= 'Text' width= "50%" height="15px"></ejs-skeleton>
})
export class AppComponent { }
,
```

Below example demonstrates the above functionalities of a Skeleton component.

APP.COMPONENT.TS

```
import { NgModule } from 'angular/core'
import { BrowserModule } from 'angular/platform-browser'
import { SkeletonModule } from '@syncfusion/ej2-angular-notifications'
import { Component } from 'angular/core';
@Component({
  imports: [
    SkeletonModule
```

```

    ],
    standalone: true,
    selector: 'my-app',
    template: `<div id="skeletonCard">
      <div class='cardProfile'>
        <ejs-skeleton id="cardProfile" shape="Circle"
width="60px"></ejs-skeleton>
      </div>
      <div class="cardinfo">
        <ejs-skeleton id="text1" width="30%" height='15px'></ejs-
skeleton><br/>
        <ejs-skeleton id="text2" width="15%" height='15px'></ejs-
skeleton>
      </div>
      <div class="cardContent">
        <ejs-skeleton id="cardImage" shape="Rectangle" width="100%"
height='150px'></ejs-skeleton>
      </div>
      <div class="cardoptions">
        <ejs-skeleton id="rightOption" shape="Rectangle" width="20%"
height='32px'></ejs-skeleton>
        <ejs-skeleton id="leftOption" shape="Rectangle" width="20%"
height='32px'></ejs-skeleton>
      </div>
    </div>`
  })
  export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

/* Represents the styles for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
#skeletonCard {
  padding: 10px;
  line-height: inherit;
  height: 330px;
}
#skeletonCard .cardProfile {
  float: left;
  margin-right: 15px;
}
#skeletonCard .cardinfo {

```

```

margin-top: 10px;
overflow: hidden;
}
#skeletonCard .cardContent {
margin: 20px 0px 20px;
}
#skeletonCard .cardoptions {
display: flex;
justify-content: space-between;
}

```

Shimmer effect in Angular Skeleton component

You can use the [shimmerEffect](#) property to change animation effect in the skeleton control. Skeleton supports **Wave**, **Pulse** and **Fade** effects and by default, the **shimmerEffect** is set to **Wave** effect.

`typescript

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
selector: 'my-app',
```

```
template: <ejs-skeleton shape= 'Circle' width= "60px" shimmerEffect= 'Pulse'></ejs-skeleton>
```

```
})
```

```
export class AppComponent { }
```

```
,
```

Below example demonstrates a list with pulse effect skeleton.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SkeletonModule } from '@syncfusion/ej2-angular-notifications'
import { Component } from '@angular/core';
@Component({
imports: [

    SkeletonModule
],
standalone: true,
selector: 'my-app',
template: `<div>
    <ul id="skeleton-list" class="e-card">
        <li>
            <div class='listProfile'>
                <ejs-skeleton shape="Circle" width="40px"
shimmerEffect='Pulse'></ejs-skeleton>
            </div>
            <div>
                <ejs-skeleton width="60%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton><br>
                <ejs-skeleton width="40%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton>

```

```

        </div>
      </li>
      <li>
        <div class='listProfile'>
          <ejs-skeleton shape="Circle" width="40px"
shimmerEffect='Pulse'></ejs-skeleton>
        </div>
        <div>
          <ejs-skeleton width="60%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton><br>
          <ejs-skeleton width="40%" height='15px'
shimmerEffect='Pulse'></ejs-skeleton>
        </div>
      </li>
    </ul>
  </div>`
  })
  export class AppComponent {}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

/* Represents the styles for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
#skeleton-list {
  padding-left: 12px;
  padding-top: 7px;
  line-height: inherit;
}
#skeleton-list li {
  list-style: none;
  display: flow-root;
  margin-bottom: 9px;
}
.listProfile {
  float: left;
  margin-right: 15px;
}

```

Styles in Angular Skeleton component

You can customize skeleton control in the below ways.

cssClass

You can customize the style of a Skeleton control by using [cssClass](#). The appearance of Angular Skeleton can be customized by changing the wave color, background color, width, and height. For detailed information, refer [index.css](#) file below.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SkeletonModule } from '@syncfusion/ej2-angular-notifications'
import { Component } from '@angular/core';
@Component({
  imports: [

    SkeletonModule
  ],
  standalone: true,
  selector: 'my-app',
  template: `<ejs-skeleton shape= 'Circle' width= "60px" cssClass= "e-
customize"></ejs-skeleton>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

INDEX.CSS

```
/* Represents the styles for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
/* Represents the styles for shimmer-wave */
.e-customize.e-skeleton.e-shimmer-wave::after{
  background-image: linear-gradient(90deg, transparent calc(50% - 100px),
  rgb(30 128 234 / 50%) 50%,
  transparent calc(50% + 100px));
}
/* Represents the background-color for skeleton-circle */
.e-customize.e-skeleton.e-skeleton-circle{
  background-color: #a8c1f2;
}
```

Visible

You can use the [visible](#) property which defines the visible state of Skeleton.

`typescript

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
  selector: 'my-app',
```

```
  template: <ejs-skeleton shape= 'Circle' width= "60px" [visible]='false'></ejs-skeleton>
```

```
})
```

```
export class AppComponent { }
```

```
,
```

Accessibility in Angular Skeleton component

The Skeleton 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 Skeleton 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 Skeleton component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Skeleton component:

Attributes	Purpose
-----	-----
<code>role=alert</code>	Used to convey important, time-sensitive or contextual message to the user.
<code>aria-label</code>	Attribute provides the text label for the Skeleton.
<code>aria-live</code>	Attributes which indicates the content changes which are not interactable are live regions.
<code>aria-busy</code>	Set to true until loading is complete, then set to false.

Ensuring accessibility

The Skeleton component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Skeleton component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Skeleton component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Smith Chart

Getting started with Angular Smithchart component

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

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
```

```
npm install -g @angular/cli
```

`

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion SmithChart package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-charts](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-charts --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-charts@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-charts@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-charts:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering TreeMap Module

Import TreeMap module into Angular application(app.module.ts) from the package `@syncfusion/ej2-angular-charts` [src/app/app.module.ts].

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the SmithchartModule for the Smithchart component
import { SmithchartModule } from '@syncfusion/ej2-angular-charts';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ej2-angular-smithchart module into NgModule
  imports: [ BrowserModule, SmithchartModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-charts` component

[src/app/app.component.ts].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the Smithchart component
  template: <ejs-smithchart style='display: block;' id='container'></ejs-smithchart>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

<!-- markdownlint-disable MD033 -->

Now use the `<code>app-container</code>` in the index.html instead of default one.

```
`html
<app-container></app-container>
```

- Now run the application in the browser using the below command.

```
npm start
```

The below example shows a basic Smithchart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  // specifies the template string for the Smithchart component
  template: `<ejs-smithchart id="smithchart-container"
height='350px'></ejs-smithchart>`
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Module Injection

Smithchart component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature service in the AppModule. In the current application, we are going to modify the above basic smithchart to visualize transmission lines.

For this application we are going to use tooltip and legend feature of the smithchart. Please find relevant feature service name and description as follows.

- SmithchartLegendService - Inject this provider to use legend feature.
- TooltipRenderService - Inject this provider to use tooltip feature.

These modules should be injected to the provider section as follows,

```
`javascript
```

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { SmithchartComponent } from '@syncfusion/ej2-angular-charts';
import { SmithchartLegendService, TooltipRenderService } from '@syncfusion/ej2-angular-charts';
@NgModule({
  imports: [
    BrowserModule,
  ],
  declarations: [AppComponent, SmithchartComponent],
  bootstrap: [AppComponent],
  providers: [ SmithchartLegendService, TooltipRenderService ]
})

```

Add Series to Smithchart

Smithchart had two type of specification for adding series.

- **dataSource** - Using this, Data object can bind directly by specifying the resistance and reactance values, series add to smithchart.
- **points** - Using this, collection of resistance and reactance values can bind directly to render series.

Below sample demonstrate adding two series to smithchart both ways.

- First series Transmission1 shows dataSource bound series.
- Second series Transmission2 shows points bound series.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px'>
<e-seriesCollection>

```

```

    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public seriesdata1: object[] = [
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
  ];
  public seriespoints: object[] = [
    { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15
  },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add title to SmithChart

smithchart **title** API used to add title for smithchart. In that **text** API used to set text of the title.

API **visible** used to toggle the title.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { TitleModel } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
[title]='title' height='350px'>
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public title: TitleModel = { text: 'Transmission lines applied for both
impedance and admittance' };
  public seriesdata1: object[] = [
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
```

```

    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
  ];
  public seriespoints: object[] = [
    { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15
},
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Marker to Smithchart

To use series marker and its customization in smithchart, use series `marker`. To display marker for particular series, need to specify `marker visible` as true. Below sample marker enabled for first series only.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
[title]='title' height='350px'>
<e-seriesCollection>

```

```

        <e-series [marker]='marker' [dataSource]='seriesdata1'
name='Transmission1' reactance='reactance' resistance='resistance'></e-
series>
        <e-series [points]='seriespoints' name='Transmission2'> </e-series>
    </e-seriesCollection>
    </ejs-smithchart>`
    })
    export class AppComponent {
        public title: object = { text: 'Transmission lines applied for both
impedance and admittance' };
        public marker: object = {
            visible: true
        };
        public seriesdata1: object[] = [
            { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
            { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
            { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
            { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
            { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
},
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
            { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
            { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
            { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
            { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
            { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
        ];
        public seriespoints: object[] = [
            { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15
},
            { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
            { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
            { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
            { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
            { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
        ];
    }

```



```

    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
  ];
  public seriespoints: object[] = [
    { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15
  },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Legend for Smithchart

Smithchart had a legend feature, which is used to denote the correspond series. To enable the legend, need to inject `SmithchartLegendService` module in the AppModule and smithchart `legendSettings` visible as true. Following example sample shows enabling legend for smithchart. Series name can customize using series name.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'

```

```

import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
[title]='title' [legendSettings]='legendSettings' height='350px'>
  <e-seriesCollection>
    <e-series [marker]='marker' [dataSource]='seriesdata1'
name='Transmission1' reactance='reactance' resistance='resistance'></e-
series>
    <e-series [points]='seriespoints' name='Transmission2'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public title: object = { text: 'Transmission lines applied for both
impedance and admittance' };
  public marker: object = {
    visible: true,
    dataLabel: {
      visible: true
    }
  };
  public legendSettings: object = {
    visible: true
  };
  public seriesdata1: object[] = [
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
  ];
  public seriespoints: object[] = [
    { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15 }
  ],

```

```

    { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable Tooltip for Smithchart Series

Smithchart had a tooltip feature, which is used to show the current point's values. To enable the tooltip, need to inject `TooltipRenderService` module in the AppModule and smithchart series `tooltip visible` as true. Following example sample shows enabling tooltip for smithchart series collection.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
[title]='title' [legendSettings]='legendSettings' height='350px'>
  <e-seriesCollection>
    <e-series [marker]='marker' [tooltip]='tooltip'
[dataSource]='seriesdata1' name='Transmission1' reactance='reactance'
resistance='resistance'></e-series>
    <e-series [points]='seriespoints' [tooltip]='tooltip'
name='Transmission2'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})

```

```

export class AppComponent {
  public title: object = { text: 'Transmission lines applied for both
impedance and admittance' };
  public marker: object = {
    visible: true,
    dataLabel: {
      visible: true
    }
  };
  public tooltip: object = {
    visible: true
  };
  public legendSettings: object = {
    visible: true
  };
  public seriesdata1: object[] = [
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0, reactance: 0.05 }, { resistance: 0, reactance: 0.05
  },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5, reactance:
0.2 },
    { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0, reactance:
0.5 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5, reactance:
0.0 },
    { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0, reactance:
-1.0 }
  ];
  public seriespoints: object[] = [
    { resistance: 0, reactance: 0.15 }, { resistance: 0, reactance: 0.15
  },
    { resistance: 0, reactance: 0.15 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3, reactance:
0.2 },
    { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0, reactance:
0.8 },
    { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5, reactance:
1.6 },
    { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0, reactance:
4.5 },
    { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25 }
  ];
}

```

```
];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Working with data in Angular Smithchart component

Smithchart can visualise the data bound from local data. The data you bind for the smithchart, should be an array of object and that should contain the field resistance and reactance. This should be bind to points or datasource in the smithchart.

Data Binding

You can bind simple JSON data to smithchart using point property in series. JSON data should contain [resistance] and [reactance] fields. This JSON data should be bind to points or datasource in the smithchart. You can any number of JSON for points or datasource as per your requirement.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px'>
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
```

```

        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    public seriespoints: object[] = [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Smith chart dimensions in Angular Smithchart component

You can render the smithchart either corresponding to its container size or you can set the size of the smithchart as per your requirement. To render the smithchart corresponding to its container size, you need to set the size for the smithchart container. Else to set the size for the smithchart as per your requirement, you can use the width and height properties in the smithchart.

Size for Container

You can render smithchart to its container size. To achieve this, you need to specify the width and height of the smithchart's container via inline or CSS as demonstrated below.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'>

```

```

    <e-seriesCollection>
      <e-series [dataSource]='seriesdata1' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
  </ejs-smithchart>`
})
export class AppComponent {
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Size for Smithchart

<!-- markdownlint-disable MD036 -->

You can also set size for smithchart directly through `[width]` and `[height]` properties. Using this properties, you can directly mention the width and height of the smithchart in pixels or you can set the width and height in percentage.

In Pixel

In smithchart's width and height property, you can directly give values in pixels like below demonstration. This will render smithchart in same size as you mentioned in you code.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-smithchart style='display: block;' id='container'
height='300px' width = '650px'>
    <e-seriesCollection>
        <e-series [dataSource]='seriesdata1' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
    </ejs-smithchart>`
  })
  export class AppComponent {
    public seriesdata1: object[] = [
      { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
      { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
      { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
      { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
      { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
      { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

In percentage

You can also specify the width and height of the smithchart in percentage. If you mention the width and height in percentage, then smithchart will be render as per the percentage of it's container size. You can set the values in percentage like below demonstration.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='85%' width = '90%'>
    <e-seriesCollection>

```



```

        <e-series [dataSource]='seriesdata1' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
</ejs-smithchart>`
    })
    export class AppComponent {
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title subtitle in Angular Smithchart component

Enable title

Title and subtitle is used to depicts the information about the data plotted in the smithchart. You can set the title and subtitle of the smithchart using the [text] property in title and subtitle. By default visibility of the title as well as subtitle is enabled. You need to set simply text for title and subtitle in your sample as like below.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
[title] ='title'>
    <e-seriesCollection>

```

```

        <e-series [dataSource]='seriesdata1' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
</ejs-smithchart>`
    })
    export class AppComponent {
        public title:object = {
            text: 'Impedance Transmission',
            subtitle: {
                text: 'Transmission'
            }
        }
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Title trim

Both title and subtitle of the smithchart can be trimmed if it exceeds the certain length. Trimming is enabled using `[enableTrim]` for title as well as subtitle. This length can be changed using the property `[maximumWidth]`. Also `[font]`, `[textAlignment]` and `[visibility]` can be customized for title as well as subtitle.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
    imports: [
        SmithchartModule
    ],
    providers: [TooltipRenderService, SmithchartLegendService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-smithchart style='display: block;' id='container'
[title] ='title'>
    <e-seriesCollection>
        <e-series [dataSource]='seriesdata1' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
    </ejs-smithchart>`
  })
export class AppComponent {
  public title:object = {
    enableTrim: true,
    maximumWidth: 70,
    textAlignment: 'Center',
    visible: true,
    text: 'Impedance Transmission',
    subtitle: {
      text: 'Transmission',
      visible: true,
      textAlignment: 'Far',
      enableTrim: true,
      maximumWidth: 40
    }
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Smith chart axis in Angular Smithchart component

Like chart, Smithchart is having support for two types of axis.

- Horizontal axis - axis drawn as straight line in the horizontal direction of the chart.
- Radial axis - axis is drawn as circular path.

Labels Customization

Axis labels are used to denote what kind of data is bound for smithchart. Using axis labels, you can easily identify in which interval chart is rendered. Using following properties we can customize the axis labels for horizontal and radial axis.

- `[labelPosition]` - used to place the labels either inside or outside the axis line.
- `[labelIntersectAction]` - used to hide the labels when intersect with other one.
- `[labelStyle]` - used to customize the properties such as font size, family, weight, opacity.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [horizontalAxis] = 'horizontalAxis' [radialAxis] =
'radialAxis'>
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' reactance='reactance'
resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public radialAxis :Object = {
    labelPosition: 'Inside',
    labelIntersectAction: 'Hide',
    labelStyle: {
      fontFamily: 'Times New Roman',
      fontWeight: 'bold',
      fontStyle: 'Italic',
      opacity: 0.75,
      size: '14px'
    }
  };
  public horizontalAxis: Object = {
    labelPosition: 'Inside',
    labelIntersectAction: 'Hide',
    labelStyle: {
      fontFamily: 'Times New Roman',
      fontWeight: 'bold',
      fontStyle: 'Italic',
      opacity: 0.75,
      size: '14px'
    }
  };
}
```

```

    }
    };
    public seriesdata1: object[] = [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    public seriespoints: object[] = [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Gridlines

To make the data in a chart that displays axes easier to read, you can display horizontal and radial axis gridlines. Gridlines extend from any horizontal and radial axes across the plot area of the smithchart.

Both horizontal and radial axis are having support for major as well as minor gridlines. Major gridlines are drawn from the position in which labels are rendered. Minor gridlines are drawn between two major gridlines as per the count we set in settings.

We can customize following things, in major as well as minor gridlines.

- **[width]** - used to customize the width of gridlines.

- `[dashArray]` - used to customize whether gridline have to render as normal line or dashed line.
- `[visible]` - used to enable or disable the visibility of the gridlines.
- `[opacity]` - used to customize the opacity of the major gridlines.
- `[count]` - used to customize the count of the minor gridlines.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [horizontalAxis] = 'horizontalAxis' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' reactance='reactance'
resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public horizontalAxis: Object = {
    majorGridLines: {
      visible: true,
      opacity: 0.8,
      width: 10
    }
  };
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
  public seriespoints: object[] = [
    { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
```

```

    { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
    { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Axisline

As name suggests that, it is a line in smithchart that can be configured to denotes the axis. By default, visibility of the axis line is true. You can customize its visibility by using visible property in axis Line. Other than visibility of the axis line, you can customize the following properties of the axis line.

- [width] - used to customize the width of the axis line.
- [dashArray] - used to render the axis line as dashed line.
- [visible] - used to enable or disable the visibility of the axis line.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [horizontalAxis] = 'horizontalAxis' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>

```

```

        <e-series [points]='seriespoints' reactance='reactance'
resistance='resistance'> </e-series>
    </e-seriesCollection>
</ejs-smithchart>`
    })
    export class AppComponent {
        public horizontalAxis: Object = {
            majorGridLines: {
                visible: true,
                opacity: 0.8,
                width: 10
            },
            axisLine: {
                width: 10,
                visible: false
            }
        };
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ];
        public seriespoints: object[] = [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```



```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

```
<!-- markdownlint-disable MD036 -->
```

Smith chart legend in Angular Smithchart component

Legend is a key used in smithchart, that contains symbol and descriptions. It provides valuable information for interpreting what the smithchart is displaying and can be represented in various colors, shapes or other identifiers based on the data. In simple words, we can define that legend is used to denote the series rendered in the smithchart.

Position and Alignment

By default visibility of the legend is false. To enable the legend, kindly set visible as true in legendSettings. Default position for the legend is bottom. By using [position] property, you can change the position of the legend. You can either place the legend at bottom, top, right and left side of the smithchart. To use the legend in smithchart, you need to import and inject the SmithchartLegend from chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top'
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
```

```

        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    public seriespoints: object[] = [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Other than these positions, you can place the legend anywhere in the smithchart. To achieve this, you have to set position as custom in legendSettings and specify the x and y coordinates using the x and y properties in the location.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>

```

```

        <e-series [points]='seriespoints' name ='Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
    </e-seriesCollection>
</ejs-smithchart>`
    })
    export class AppComponent {
        public legendSettings: Object = {
            visible: true,
            position: 'Custom',
            location:{
                x: 80,
                y: 100
            }
        }
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ];
        public seriespoints: object[] = [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Alignment

Other than positioning the legend in the smithchart, you can customize its alignment also. By default, legend is aligned at center. Using the `[alignment]` property, you can align the legend in near and far locations of the smithchart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name = 'Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top',
    alignment: 'Near'
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
  public seriespoints: object[] = [
    { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
    { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
```

```

    { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

Legend Shape

By default, legend is rendered in the circle shape and the color of the shape is as same as series color in the smithchart. Using the property `[shape]` in legend settings, you can change the icon shape of the legend as rectangle, triangle and so on.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name = 'Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top',

```

```

        shape: 'Rectangle'
    }
    public seriesdata1: object[] = [
        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    public seriespoints: object[] = [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';

```

```

@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top',
    height: 100,
    width: 200
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
  public seriespoints: object[] = [
    { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
    { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
    { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
  ];
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Padding

You can customize the space between two legend items and space between legend shape and text as per your requirement. For customizing the space between two legend items, you can use `[itemPadding]` property. To control space between legend shape and text, you can use `[shapePadding]` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top',
    itemPadding: 5,
    shapePadding: 10
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
```



```

        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    public seriespoints: object[] = [
        { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
        { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
        { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
        { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
        { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
        { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
        { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
        { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Toggle Visibility

By default series name is displayed in the legend. You can collapse the visibility of the series by clicking the legend for the particular series. You can toggle the series visibility as true or false using the `[toggleVisibility]` property. By default it is true.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
<e-seriesCollection>

```

```

    <e-series [dataSource]='seriesdata1' name='Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    <e-series [points]='seriespoints' name='Transmission2'
reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
}))
export class AppComponent {
  public legendSettings: Object = {
    visible: true,
    position: 'Top',
    toggleVisibility: true
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
  public seriespoints: object[] = [
    { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
    { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
    { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Smith chart tooltip in Angular Smithchart component

Smithchart will display details about the points through tooltip, when the mouse is moved over the point. By default, tooltip is disabled. To enable the tooltip for smithchart, you need to import and inject TooltipRender module from chart. And also set the property visible as true, in tooltip settings. You can customize the tooltip's visibility and appearance differently each series in the smithchart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container'
height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' [marker]='marker'
[tooltip]='tooltip' name='Transmission1' reactance='reactance'
resistance='resistance'> </e-series>
    <e-series [marker]='marker' [tooltip]='tooltip'
[points]='seriespoints' name='Transmission2' reactance='reactance'
resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart>`
})
export class AppComponent {
  public legendSettings: Object = { };
  public marker: object = {
    visible: true
  };
  public tooltip: object = {
    visible: true
  };
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
  public seriespoints: object[] = [
```

```

    { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
    { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
    { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
    { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
    { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
    { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
    { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
    { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Smith chart marker in Angular Smithchart component

Markers and Datalabels are used to provide information about the data points in the series. You can add a shape to adorn each data point. By default marker and datalabel both are disabled in smithchart. You can enable both of them by setting visible property as true in marker and datalabel settings

Marker

Default visibility of marker is false. You can enable the marker by setting property visible as true in marker settings. This will add marker for each point in the series. Using marker setting, you can customize marker differently for each series in smithchart.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container' >
<e-seriesCollection>

```

```

        <e-series [marker] ='marker' [dataSource]='seriesdata1' name
        ='Transmission1' reactance='reactance' resistance='resistance'> </e-series>
        </e-seriesCollection>
    </ejs-smithchart>`
    })
    export class AppComponent {
        public marker:object = {
            visible: true
        }
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
            ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Marker Customization

Using marker settings in series, you can customize the marker for each series differently. Using marker settings, you can customize following properties differently for each series in the smithchart.

- [width] - To control the width of the marker.
- [height] - To control the height of the marker.
- [fill] - Used to customize the fill color of the marker.
- [opacity] - Used to customize the opacity of the marker.
- [border] - Used to control the width and color of the marker's border.
- [shape] - Used to change the shape of the marker.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
    imports: [

```

```

        SmithchartModule
    ],
    providers: [TooltipRenderService, SmithchartLegendService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-smithchart style='display: block;' id='container' >
        <e-seriesCollection>
            <e-series [marker] ='marker' [dataSource]='seriesdata1' name
            ='Transmission1' reactance='reactance' resistance='resistance'> </e-series>
        </e-seriesCollection>
    </ejs-smithchart>`
  })
  export class AppComponent {
    public marker:object = {
      visible: true,
      height: 10,
      width: 10,
      fill: '#ff99ff',
      opacity: 1,
      shape: 'rectangle',
      border: {
        color: '#cc00cc',
        width: 2
      }
    }
    public seriesdata1: object[] = [
      { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
      { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
      { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
      { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
      { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
      { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Datalabels

By default, datalabel is disabled. You can enable the datalabel by setting property visible as true in datalabel settings. For each point in series, data label is created. Datalabel for each series can be customized differently using datalabel settings.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container' >
    <e-seriesCollection>
      <e-series [marker] = 'marker' [dataSource]='seriesdata1' name
      = 'Transmission1' reactance='reactance' resistance='resistance'> </e-series>
    </e-seriesCollection>
  </ejs-smithchart>`
})
export class AppComponent {
  public marker:object = {
    dataLabel: {
      visible: true
    }
  };
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Datalabel customization

Using datalabel settings in marker, you can customize the datalabel for each series differently. In datalabel, you can customize the following properties differently for each series.

- **[fill]** - Used to changes the fill color of the data label's shape.

- **[opacity]** - Used to control the opacity of the data label's shape.
- **[border]** - Used to customize the width and color of the border.
- **[textStyle]** - Used to customize the font color, width and size.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container' >
    <e-seriesCollection>
      <e-series [marker] = 'marker' [dataSource]='seriesdata1' name
= 'Transmission1' reactance='reactance' resistance='resistance'> </e-series>
    </e-seriesCollection>
  </ejs-smithchart>`
})
export class AppComponent {
  public marker:object = {
    dataLabel: {
      visible: true,
      fill: '#99ffcc',
      opacity: 1,
      border: {
        color: '#1aff8c',
        width: 2,
      }
    }
  };
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
}
```

MAIN.TS


```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Smith chart series in Angular Smithchart component

You can add any number of series to the smithchart as per your requirement. You can use series setting to either add or customize the data. For the points or datasource added in the series, line is drawn. You can customize the each series as per your requirement with marker, datalabel, animation, opacity and so on.

points or datasource

For adding values in the smithchart, you can use either points or datasource in the series. Points and datasource both should be array of object which should contain the field names resistance and reactance.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container' >
    <e-seriesCollection>
      <e-series [dataSource]='seriesdata1' name = 'Transmission1'
reactance='reactance' resistance='resistance'> </e-series>
    </e-seriesCollection>
  </ejs-smithchart>`
})
export class AppComponent {
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Series customization

Using following options in series settings, you can customize each series in smithchart as per your requirement.

- **[fill]** - Used to customize the fill color for the series.
- **[enableSmartLabels]** - Used to place the data labels on the smithchart without overlapping with each other.
- **[visibility]** - Used to handle the visibility of the series.
- **[opacity]** - Used to control the opacity of the series line.
- **[width]** - Used to customize the width of the series line.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart style='display: block;' id='container' >
    <e-seriesCollection>
      <e-series [dataSource]='seriesdata1' [marker] = 'marker' name
='Transmission1' fill= '#009933' visibility= 'visible'
reactance='reactance' width= '2.5' opacity= '0.75' resistance='resistance'>
    </e-series>
    </e-seriesCollection>
  </ejs-smithchart>`
})
export class AppComponent {
  public marker: Object = {
    dataLabel: {
      visible: true
    }
  }
  public seriesdata1: object[] = [
    { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
    { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
    { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
```

```

    { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
    { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
    { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Smith chart print in Angular Smithchart component

Print

The rendered smithchart can be printed directly from the browser by calling the public method print. ID of the smithchart's div element must be passed as argument to that methods.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart #smithchart style='display: block;'
id='container' >
  <e-seriesCollection>
    <e-series [marker] = 'marker' [dataSource]='seriesdata1' name
='Transmission1' reactance='reactance' resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart> <div>
    <button id="togglebtn2"
(click)='print()'>Print</button>
  </div>`
})
export class AppComponent {
  @ViewChild('smithchart')
  public smithchart?: any;
  public marker:object = {
    dataLabel: {
      visible: true
    }
  };
  public seriesdata1: object[] = [

```

```

        { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
        { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
        { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
        { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
        { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
        { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
    ];
    print() {
        this.smithchart.print();
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Export

The rendered smithchart can be exported to JPEG , PNG, SVG or PDF format by using export method in smithchart. Input parameters for this method are Export Type for format and fileName of result.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SmithchartModule, TooltipRenderService, SmithchartLegendService }
from '@syncfusion/ej2-angular-charts'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    SmithchartModule
  ],
  providers: [TooltipRenderService, SmithchartLegendService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-smithchart #smithchart style='display: block;'
id='container' height='350px' [legendSettings] = 'legendSettings' >
  <e-seriesCollection>
    <e-series [dataSource]='seriesdata1' [marker]='marker'
[tooltip]='tooltip' name='Transmission1' reactance='reactance'
resistance='resistance'> </e-series>
    <e-series [marker]='marker' [tooltip]='tooltip'
[points]='seriespoints' name='Transmission2' reactance='reactance'
resistance='resistance'> </e-series>
  </e-seriesCollection>
</ejs-smithchart> <div>

```

```

        <button id="togglebtn2"
(click)='export()'>Export</button>
        </div>`
    })
    export class AppComponent {
        @ViewChild('smithchart')
        public smithchart?: any;
        public marker: object = {
            visible: true
        };
        public legendSettings: Object = { };
        public tooltip: object = {
            visible: true
        };
        public seriesdata1: object[] = [
            { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
            { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
            { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
            { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
            { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
            { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
        ];
        public seriespoints: object[] = [
            { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
            { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
            { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
            { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
            { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
            { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
            { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
            { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },
        ];
        export() {
            this.smithchart.export('PNG', 'SmithChart');
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Accessibility in Angular Smithchart component

The Smith chart 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 Smith chart component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader 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 Smith chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Smith chart component:

- `img` (role)
- `region` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)

Keyboard interaction

The Smith chart 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 Smith chart component.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the next element in the Smith chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Smith chart. |

| **Ctrl + P** | Prints the Smith chart. |

Ensuring accessibility

The Smith chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Smith chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Smith chart component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Sparkline

Getting started with Angular Sparkline component

This section explains you the steps required to create a Sparkline and demonstrate the basic usage of the Sparkline control.

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`bash
```

```
npm install -g @angular/cli
```

```
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Adding Syncfusion Sparkline package

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) registry.

To install sparkline component, use the following command.

```
`bash
npm install @syncfusion/ej2-angular-charts --save
`
```

The **--save** will instruct NPM to include the sparkline package inside of the **dependencies** section of the **package.json**.

Registering Sparkline Module

- Import Sparkline module into Angular application(**app.module.ts**) from the package **@syncfusion/ej2-angular-charts** [**src/app/app.module.ts**].

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the SparklineModule for the Sparkline component
import { SparklineModule } from '@syncfusion/ej2-angular-charts';
import { AppComponent } from './app.component';
@NgModule({
//declaration of sparkline module into NgModule
imports: [ BrowserModule, SparklineModule ],
declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
`
```

Modify the template in **app.component.ts** file to render the **ej2-angular-charts** component [**src/app/app.component.ts**].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
```



```
@Component({
  selector: 'app-container',
  // specifies the template string for the Sparkline component
  template: <ejs-sparkline id='sparkline-container'></ejs-sparkline>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

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

Now use the `<app-container>` in the index.html instead of default one.

```
`html
<app-container></app-container>
```

Now run the application in the browser using the below command.

```
npm start
```

The below example shows a basic Sparkline.

```
`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-container',
  // specifies the template string for the Sparkline component
  template: <ejs-sparkline id="sparkline-container"></ejs-sparkline>
})
export class AppComponent {
}
```

As we didn't specify `dataSource` to the Sparkline, no shape will be rendered and only an empty SVG element is appended to the Sparkline container.

Module Injection

Sparkline component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature service in the AppModule. Find the relevant feature service available in Sparkline and its description as follows.

- SparklineTooltipService - Inject this provider to use tooltip series.

In the current application, we are going to modify the above basic Sparkline to visualize the sparkline types.

For this application we are going to use tooltip features of the Sparkline.

Now import the SparklineTooltipService from Sparkline package and inject it into the AppModule.

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { SparklineComponent, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
@NgModule({
  imports: [
    BrowserModule,
  ],
  declarations: [AppComponent, SparklineComponent],
  bootstrap: [AppComponent],
  providers: [ SparklineTooltipService ]
})
`
```

Bind data source to Sparkline

The `[dataSource]` property is used for binding data source to Sparkline. This property takes collection value as input. For example, the list of objects can be provided as input.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='400px' height='350px'
[dataSource]="data" xName="xval" yName="yval">
</ejs-sparkline>`
})
```

```
export class AppComponent {
  public data: object[] = [
    { xval: 1, yval: 20090440 },
    { xval: 2, yval: 20264080 },
    { xval: 3, yval: 20434180 },
    { xval: 4, yval: 21007310 },
    { xval: 5, yval: 21262640 },
    { xval: 6, yval: 21515750 },
    { xval: 7, yval: 21766710 },
    { xval: 8, yval: 22015580 },
    { xval: 9, yval: 22262500 },
    { xval: 10, yval: 22507620 },
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Change the type of Sparkline

We can change the Sparkline types by setting the [type] property as

[Line],[Column],[WinLoss],[Pie],[Area]. Here, we have given the Sparkline type as [Area].

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='400px' height='350px'
[dataSource]="data" xName="xval" yName="yval" type="Area">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { xval: 1, yval: 20090440 },
    { xval: 2, yval: 20264080 },
    { xval: 3, yval: 20434180 },
    { xval: 4, yval: 21007310 },
    { xval: 5, yval: 21262640 },
    { xval: 6, yval: 21515750 },
    { xval: 7, yval: 21766710 },
```

```

    { xval: 8, yval: 22015580 },
    { xval: 9, yval: 22262500 },
    { xval: 10, yval: 22507620 },
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable tooltip for Sparkline

Sparkline will displays the sparkline details through tooltip, when the mouse is moved over the sparkline. You can enable tooltip by setting the [visible] property as true in [tooltipSettings] object and by injecting SparklineTooltipService module in the AppModule.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
import { SparklineTooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='400px' height='350px'
[dataSource]="data" xName="xval" yName="yval" type="Area"
[tooltipSettings]="tooltipSettings">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { xval: 1, yval: 20090440 },
    { xval: 2, yval: 20264080 },
    { xval: 3, yval: 20434180 },
    { xval: 4, yval: 21007310 },
    { xval: 5, yval: 21262640 },
    { xval: 6, yval: 21515750 },
    { xval: 7, yval: 21766710 },
    { xval: 8, yval: 22015580 },
    { xval: 9, yval: 22262500 },
    { xval: 10, yval: 22507620 },
  ];
  public tooltipSettings: SparklineTooltipSettingsModel = {

```

```

        visible: true,
        format: '${xval} : ${yval}'
    };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Sparkline dimension in Angular Sparkline component

Size for container

Sparkline can be rendered to its container size. You can set the size through inline or CSS as shown in the following code.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='650px' height='350px'
[dataSource]="data" xName="xval" yName="yval">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Size for sparkline

<!-- markdownlint-disable MD036 -->

You can also set the size for sparkline directly using the [width](#) and [height](#) properties.

In pixel

You can set the size for sparkline in pixel as demonstrated in the following code.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350' height='150'
[dataSource]="data" xName="xval" yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

In percentage

By setting values in percentage, sparkline gets its dimension with respect to its container. For example, when the height is set to '50%', sparkline is rendered to half of its container height.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='80%' height='50%'
[dataSource]="data" xName="xval" yName="yval">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sparkline types in Angular Sparkline component

Different types of shapes can be used to represent the sparkline. You can change the sparkline type by setting the type property. Sparkline supports the following types:

- Line
- Column

- Win-Loss
- Pie
- Area

The following code sample shows different types of sparkline.

<!-- markdownlint-disable MD036 -->

Line

The [Line] type is used to render the sparkline series as line.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
type="Line" [dataSource]="data" xName="xval" yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Column

The [Column] type is used to render the sparkline series as column.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
type="Column" [dataSource]="data" xName="xval" yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Pie

The [Pie] type is used to render the sparkline series as pie.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
```

```

@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
type="Pie" [dataSource]="data" xName="xval" yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Pie

The [Pie] type is used to render the sparkline series as pie.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
type="Pie" [dataSource]="data" xName="xval" yName="yval">
  </ejs-sparkline>`
})

```

```

    })
    export class AppComponent {
        public data: object[] = [
            { x: 0, xval: '2005', yval: 20090440 },
            { x: 1, xval: '2006', yval: 20264080 },
            { x: 2, xval: '2007', yval: 20434180 },
            { x: 3, xval: '2008', yval: 21007310 },
            { x: 4, xval: '2009', yval: 21262640 },
            { x: 5, xval: '2010', yval: 21515750 },
            { x: 6, xval: '2011', yval: 21766710 },
            { x: 7, xval: '2012', yval: 22015580 },
            { x: 8, xval: '2013', yval: 22262500 },
            { x: 9, xval: '2014', yval: 22507620 },
        ];
    };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Win Loss

The [WinLoss] type is used to render the sparkline series as Win Loss.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
type="WinLoss" [dataSource]="data" xName="xval" yName="yval">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },

```

```

        { x: 7, xval: '2012', yval: 22015580 },
        { x: 8, xval: '2013', yval: 22262500 },
        { x: 9, xval: '2014', yval: 22507620 },
    ];
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Area

The [Area] type is used to render the sparkline series as area.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='70%' height='100px'
  type="Area" [dataSource]="data" xName="xval" yName="yval">
    </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: '2005', yval: 20090440 },
    { x: 1, xval: '2006', yval: 20264080 },
    { x: 2, xval: '2007', yval: 20434180 },
    { x: 3, xval: '2008', yval: 21007310 },
    { x: 4, xval: '2009', yval: 21262640 },
    { x: 5, xval: '2010', yval: 21515750 },
    { x: 6, xval: '2011', yval: 21766710 },
    { x: 7, xval: '2012', yval: 22015580 },
    { x: 8, xval: '2013', yval: 22262500 },
    { x: 9, xval: '2014', yval: 22507620 },
  ];
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Axis customization in Angular Sparkline component

You can customize axis value types and min and max values of the sparkline.

Change value type of the sparkline

You can change the sparkline value type by setting the `[valueType]` property to `[Numeric]`, `[Category]`, or `[DateTime]`.

<!-- markdownlint-disable MD036 -->

DateTime

You can assign date-time values to the sparkline by setting the `[valueType]` property to `[DateTime]`.

`[app.component.ts]`

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='130px' height='150px'
type='Column' valueType= 'DateTime' [dataSource]="data" xName="xDate"
yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { xDate: new Date(2018, 0, 1), x: 0, yval: 4 },
    { xDate: new Date(2018, 0, 2), x: 1, yval: 4.5 },
    { xDate: new Date(2018, 0, 3), x: 2, yval: 8 },
    { xDate: new Date(2018, 0, 4), x: 3, yval: 7 },
    { xDate: new Date(2018, 0, 5), x: 4, yval: 6 },
    { xDate: new Date(2018, 0, 8), x: 5, yval: 8 },
    { xDate: new Date(2018, 0, 9), x: 6, yval: 8 },
    { xDate: new Date(2018, 0, 10), x: 7, yval: 6.5 },
    { xDate: new Date(2018, 0, 11), x: 8, yval: 4 },
    { xDate: new Date(2018, 0, 12), x: 9, yval: 5.5 }
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Category

You can assign category values to the sparkline by setting [valueType] to [Category].

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='130px' height='150px'
type='Column' valueType='Category' [dataSource]="data" xName="xval"
yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 0, xval: 'Robert', yval: 60 },
    { x: 1, xval: 'Andrew', yval: 65 },
    { x: 2, xval: 'Suyama', yval: 70 },
    { x: 3, xval: 'Michael', yval: 80 },
    { x: 4, xval: 'Janet', yval: 55 },
    { x: 5, xval: 'Davolio', yval: 90 },
    { x: 6, xval: 'Fuller', yval: 75 },
    { x: 7, xval: 'Nancy', yval: 85 }
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Numeric

You can assign numeric values to the sparkline by setting the [valueType] to [Numeric].

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='130px' height='150px'
type='Column' valueType= 'Numeric'[dataSource]="data" xName="xval"
yName="yval">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [
    { x: 1, xval: 2010, yval: 190 },
    { x: 2, xval: 2011, yval: 165 },
    { x: 3, xval: 2012, yval: 158 },
    { x: 4, xval: 2013, yval: 175 },
    { x: 5, xval: 2014, yval: 200 },
    { x: 6, xval: 2015, yval: 180 },
    { x: 7, xval: 2016, yval: 210 }
  ];
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

<!-- markdownlint-disable MD036 -->

Change min and max values of axis

You can change the min and max values of x-axis by setting the [minX] and [maxX] values to the [axisSettings] property. You can also change the min and max values of y-axis by setting the [minY] and [maxY] values to the [axisSettings] property.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
```

```

        SparklineModule
    ],
    providers: [SparklineTooltipService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-sparkline id='container'width='130px' height='150px'
type='Column' valueType= 'Numeric' [axisSettings] = 'axisSettings'
[dataSource]="data" xName="x" yName="yval">
    </ejs-sparkline>`
  })
  export class AppComponent {
    public data: object[] = [
      { x: 0, yval: 50 },
      { x: 1, yval: 30 },
      { x: 2, yval: 20 },
      { x: 3, yval: 30 },
      { x: 4, yval: 50 },
      { x: 5, yval: 40 },
      { x: 6, yval: 20 },
      { x: 7, yval: 10 },
      { x: 8, yval: 30 },
      { x: 9, yval: 10 },
      { x: 10, yval: 40 }
    ];
    public axisSettings: object = {
      minY: 0, maxY: 150
    };
  };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change value of axis

You can set horizontal axis line value of the sparkline by setting `[value]` to the `[axisSettings]` property. The following code example shows this.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,

```



```

    selector: 'app-container',
    template: `<ejs-sparkline id='container' width='130px' height='150px'
type='Column' valueType= 'Numeric' [axisSettings] = 'axisSettings'
[dataSource]="data" xName="xval" yName="yval">
    </ejs-sparkline>`
  })
  export class AppComponent {
    public data: object[] = [
      { x: 0, yval: 50 },
      { x: 1, yval: 30 },
      { x: 2, yval: 20 },
      { x: 3, yval: 30 },
      { x: 4, yval: 50 },
      { x: 5, yval: 40 },
      { x: 6, yval: 20 },
      { x: 7, yval: 10 },
      { x: 8, yval: 30 },
      { x: 9, yval: 10 },
      { x: 10, yval: 40 }
    ];
    public axisSettings: object = {
      value: 25
    };
  };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Axis line customization

Axis of the sparkline can be collapsed using the `[visible]` property in `[lineSettings]`; this is not applicable for win-loss. You can customize the `[color]`, `[width]`, `[opacity]`, and `[dashArray]` of axis line.

`[app.component.ts]`

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='130px' height='150px'
type='Column' valueType= 'Numeric' [axisSettings] = 'axisSettings'
[dataSource]="data" xName="x" yName="yval">

```

```

    </ejs-sparkline>`
  })
  export class AppComponent {
    public data: object[] = [
      { x: 0, yval: 50 },
      { x: 1, yval: 30 },
      { x: 2, yval: 20 },
      { x: 3, yval: 30 },
      { x: 4, yval: 50 },
      { x: 5, yval: 40 },
      { x: 6, yval: 20 },
      { x: 7, yval: 10 },
      { x: 8, yval: 30 },
      { x: 9, yval: 10 },
      { x: 10, yval: 40 }
    ];
    public axisSettings: object = {
      // To configure axis line settings
      lineSettings: {
        visible: true,
        color: "#ff14ae",
        dashArray: 5
      }
    };
  };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Special points customization in Angular Sparkline component

You can customize the points by initializing the point colors. The customization options allows to differentiate the [start], [end], [positive], [negative], and [low] points. This customization is only applicable for line, column, and area type sparkline.

<!-- markdownlint-disable MD036 -->

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,

```

```

        selector: 'app-container',
        template: `<ejs-sparkline id='container' width='130px' height='150px'
endPointColor='green' negativePointColor='red' startPointColor='green'
lowPointColor='orange' highPointColor='blue' valueType= 'Category'
type='Column' [dataSource]="data" xName="xval" yName="yval">
        </ejs-sparkline>`
    })
    export class AppComponent {
        public data: object[] = [
            { x: 0, xval: 'AUDI', yval: 1 },
            { x: 1, xval: 'BMW', yval: 5 },
            { x: 2, xval: 'BUICK', yval: -1 },
            { x: 3, xval: 'CETROEN', yval: -6 },
            { x: 4, xval: 'CHEVROLET', yval: 0 },
            { x: 5, xval: 'FIAT', yval: 1 },
            { x: 6, xval: 'FORD', yval: -2 },
            { x: 7, xval: 'HONDA', yval: 7 },
            { x: 8, xval: 'HYUNDAI', yval: -9 },
            { x: 9, xval: 'JEEP', yval: 0 },
            { x: 10, xval: 'KIA', yval: -10 },
            { x: 11, xval: 'MAZDA', yval: 3 }
        ];
    };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tie point color

Tie point color is used to configure the win-loss series type sparkline's y-value point color. The following code sample shows the tie point color of sparkline series.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='130px' height='150px'
type='WinLoss' valueType= 'Numeric' tiePointColor='blue'
[dataSource]="data" >
    </ejs-sparkline>`
})

```

```

    })
    export class AppComponent {
        public data: object[] = [
            12, 15, -10, 13, 15, 6, -12, 17, 13, 0, 8, -10
        ] as any;
    };

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Range band in Angular Sparkline component

This section explains how to customize the sparkline with multiple range bands.

Range band customization

The range band feature is used to highlight a particular range along with the y-axis using the [startRange] and [endRange] properties. You can also customize the [color] and [opacity] of the range band.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
@Component({
    imports: [
        SparklineModule
    ],
    providers: [SparklineTooltipService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-sparkline id='container' width='150px' height='150px'
lineWidth= 2 fill = '#0d3c9b' [rangeBandSettings] = 'rangeBandSettings'
[dataSource]="data">
    </ejs-sparkline>`
})
export class AppComponent {
    public data: object[] = [ 0, 6, 4, 1, 3, 2, 5 ] as any;
    public rangeBandSettings: object[] = [{
        startRange: 1,
        endRange: 3,
        color: '#bfd4fc',
        opacity: 0.4
    }];
};

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Multiple range band customization

You can define multiple range bands to a sparkline as shown in the following code sample.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='150px' height='150px'
lineWidth= 2 fill= '#0d3c9b' [rangeBandSettings] ='rangeBandSettings'
[dataSource]="data" >
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [0, 6, 4, 1, 3, 2, 5] as any;
  public rangeBandSettings: object[] = [
    {
      startRange: 1,
      endRange: 2,
      color: '#bfd4fc',
      opacity:0.4
    },
    {
      startRange: 4,
      endRange: 5,
      color: 'red',
      opacity:0.4
    }
  ]
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Marker in Angular Sparkline component

This section explains how to add markers to the sparkline.

Adding marker to the sparkline

To add marker to the sparkline, specify the **visible** of **markerSettings** as following values. The **visible** will accept multiple values too.

- All - Enables markers for all points.
- Start - Enables marker for the start point.
- End - Enables marker for the end point.
- High - Enables marker for the high point.
- Low - Enables marker for the low point.
- Negative - Enables markers for the negative points.

The following code example shows enabling markers for all points.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[markerSettings] = 'markerSettings' [axisSettings] = 'axisSettings'
[dataSource]="data" >
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [0, 6, 4, 1, 3, 2, 5] as any;
  public markerSettings: object = {
    visible: ['All']
  };
  public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 7, minY: -1
  };
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Adding marker to special point

In sparkline, markers can be enabled for particular points such as the start, end, low, high, or negative. The following code examples shows enabling markers for the high and low points.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[axisSettings]='axisSettings' [markerSettings]='markerSettings'
lowPointColor= 'red' highPointColor= 'blue' [dataSource]="data" >
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
  public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 7, minY: -1
  };
  public markerSettings: object = {
    visible: ['High', 'Low']
  };
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customizing markers

Sparkline markers can be customized in terms of fill color, border color, width, opacity, and size. The following code example shows customizing marker's fill, border, and size.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
```

```
imports: [
    SparklineModule
],
providers: [SparklineTooltipService],
standalone: true,
selector: 'app-container',
template: `<ejs-sparkline id='container' width='350px' height='200px'
[axisSettings]='axisSettings' [markerSettings]='markerSettings'
[dataSource]="data" >
    </ejs-sparkline>`
})
export class AppComponent {
    public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
    public markerSettings: object = {
        visible: ['All'],
        size: 5, fill: 'white', border: { color: 'blue', width: 2}
    };
    public axisSettings: object = {
        minX: -1, maxX: 7, maxY: 7, minY: -1
    };
};
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Data labels in Angular Sparkline component

Data labels are used to display values of data points to improve the readability.

Enable data label

To enable data label for sparkline series, provide **visible** of the **dataLabelSettings** as following possible values:

- All - Enables data label of all points.
- Start - Enables data label of the start point.
- End - Enables data label of the end point.
- High - Enables data label of the high point.
- Low - Enables data label of the low point.
- Negative - Enables data labels of the negative points.

The following example shows enabling sparkline data label for all points.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
```



```
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
    fill= 'blue' [axisSettings] = 'axisSettings' [dataLabelSettings] =
    'dataLabelSettings' [dataSource]="data" >
    </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [0, 6, 4, 1, 3, 2, 5] as any;
  public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 7, minY: -1
  };
  public dataLabelSettings: object = {
    visible: ['All']
  };
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize data label

Data labels can be customized using the fill, border, opacity, and textStyle. The following code example shows customizing data label border, text color, and fill color.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
    [dataLabelSettings]='dataLabelSettings' [axisSettings] = 'axisSettings'
    fill= 'blue'[dataSource]="data" >
    </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [0, 6, 4, 1, 3, 2, 5] as any;
```

```

public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 8, minY: -1
};
// To customize data label fill, border, and text color
public dataLabelSettings: object = {
    visible: ['All'],
    border: { color: 'blue', width: 1},
    fill: 'blue', opacity: 0.4,
    textStyle: {
        color: 'white'
    }
};
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Format data label text

The text of data labels can be formatted using the `format` API in the sparkline `dataLabelSettings`. By default, data label shows the y-value of point. The following code example shows how to display x and y-values for points.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='500px' height='200px'
[axisSettings]='axisSettings' fill= 'blue'
[dataLabelSettings]='dataLabelSettings' valueType= 'Category'
[dataSource]="data" xName= 'x' yName= 'y'>
    </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed',
y: 2 }, {x: 'Thu', y: 4 }, {x: 'Fri', y: 6 },];
  public dataLabelSettings: object = {
    format: '$ {x} : $ {y}',
    visible: ['All'],
    border: { color: 'blue', width: 1},

```

```

        fill: 'blue', opacity: 0.4,
        textStyle: {
            color: 'white'
        }
    };
    public axisSettings: object = {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

User interaction in Angular Sparkline component

Sparkline has two user interaction features: tooltip and tracker line.

Tooltip

The sparkline provides options to display details about values of data points through tooltips when hovering the mouse over data point. To use tooltip in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

The following code example shows enabling tooltip for sparkline with format.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
    imports: [
        SparklineModule
    ],
    providers: [SparklineTooltipService],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-sparkline id='container' width='500px' height='200px'
[axisSettings]='axisSettings' [tooltipSettings]='tooltipSettings' fill=
'blue' valueType= 'Category' [dataSource]='data' xName='x' yName='y'>
</ejs-sparkline>`
})
export class AppComponent {
    public data: object[] = [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed',
y: 2 }, {x: 'Thu', y: 4 }, {x: 'Fri', y: 6 }];
    public axisSettings: object = {
        minX: -1, maxX: 7, maxY: 8, minY: -1
    };
};

```

```

    public tooltipSettings: object = {
        visible: true,
        // formatting tooltip text
        format: '${x} : ${y}'
    };
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tooltip customization

The fill color, text styles, format, and border of the tooltip can be customized. The following code example shows customization of tooltip's fill color and text style.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='500px' height='200px'
[axisSettings]='axisSettings' [tooltipSettings]='tooltipSettings' fill=
'#033e96' valueType= 'Category' [dataSource]="data" xName="x" yName="y">
</ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed',
y: 2 }, {x: 'Thu', y: 4 }, {x: 'Fri', y: 6 }];
  public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 8, minY: -1
  };
  public tooltipSettings: object= {
    visible: true,
    format: '${x} : ${y}',
    fill: '#033e96',
    textStyle: {
      color: 'white'
    },
  },
}
};

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tooltip template

Sparkline tooltip has template support. By using tooltip template, you can customize tooltips. The following code example shows more customization options provided to `sparktooltip` css class that is used in tooltip template div. Using this template, images also can be added to tooltip.

```
`css

.sparktooltip {
border-radius: 5px;
background: #008cff;
color: #FFFFFF !important;
font-size: 16px;
font-style: italic;
padding: 8px;
}
`
```

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
imports: [
SparklineModule
],
providers: [SparklineTooltipService],
standalone: true,
selector: 'app-container',
template: `<ejs-sparkline id='container' width='500px' height='200px'
[axisSettings]='axisSettings' [tooltipSettings]='tooltipSettings' fill=
'#033e96' valueType= 'Category' [dataSource]="data" xName="x" yName="y">
</ejs-sparkline>`
})
export class AppComponent {
public data: object[] = [{x: 'Mon', y: 3 }, {x: 'Tue', y: 5 }, {x: 'Wed',
y: 2 }, {x: 'Thu', y: 4 }, {x: 'Fri', y: 6 }];
```

```

    public axisSettings: object = {
      minX: -1, maxX: 7, maxY: 8, minY: -1
    };
    public tooltipSettings: object = {
      visible: true,
      template: '<div style=" border-radius: 5px; background: #008cff;
padding: 8px; color: #FFFFFF !important; font-size: 16px; font-style:
italic;">${x} : ${y}<div>'
    }
  };

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Track line

The track line tracks data points that are closer to the mouse position or touch contact.

To enable track lines for sparkline, specify the `visible` option of `trackLineSettings` to true. Based on theme, tracker color will be changed. The default value of tracker color is black.

To use track line in sparkline, inject the `SparklineTooltip` module to sparkline using the inject method.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container'width='500px' height='200px'
[axisSettings]='axisSettings' [tooltipSettings]='tooltipSettings' fill=
'#033e96' valueType= 'Category' [dataSource]="data" xName="x" yName="y">
</ejs-sparkline>`
})
export class AppComponent { public data: object[] = [{x: 'Mon', y: 3 }, {x:
'Tue', y: 5 }, {x: 'Wed', y: 2 }, {x: 'Thu', y: 4 }, {x: 'Fri', y: 6 }];
  public axisSettings: object = {
    minX: -1, maxX: 7, maxY: 8, minY: -1
  };
  public tooltipSettings: object = {
    visible: true,
    trackLineSettings: {

```

```

        visible: true,
        color: '#033e96',
        width: 1
    };
};
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

[Appearance in Angular Sparkline component](#)

The appearance of the sparkline can be customized using margin, container Area border, and container Area background.

[Sparkline border](#)

The `containerArea` border of the sparkline is used to render border to cover sparkline area.

The following code example shows the sparkline with overall border.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts';
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[border]= 'border' [containerArea]='containerArea' [dataSource]="data" fill=
'#b2cfff' type="Area">
    </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
  public containerArea: object = {
    border: { color: '#033e96', width: 2 }
  };
  public border: object = { color: '#033e96', width: 1 };
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sparkline padding

Padding is used to specify padding value between container and sparkline. By default, padding value of the sparkline is 5. Sparkline **padding** values are specified by the left, right, top, and bottom.

The following code example shows the sparkline with overall padding is set to 20.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[padding] = 'padding' [border]= 'border' [containerArea]='containerArea'
[dataSource]="data" fill= '#b2cfff' type="Area">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
  public containerArea: object= {
    border: { color: '#033e96', width: 2 }
  };
  public padding: object= { left: 20, right: 20, bottom: 20, top: 20};
  public border: object= { color: '#033e96', width: 1 };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sparkline area customization

The background color of the sparkline area can be customized using the **containerArea** background color. By default, the sparkline background color is **transparent**.

[app.component.ts]

APP.COMPONENT.TS


```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[padding]='padding' [border]='border' [containerArea]='containerArea'
[dataSource]="data" fill= '#b2cfff' type="Area">
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
  public containerArea: object= {
    border: { color: '#033e96', width: 2 },
    background: '#eff1f4',
  };
  public padding: object= { left: 20, right: 20, bottom: 20, top: 20};
  public border: object= { color: '#033e96', width: 1 };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Sparkline theme

Datalabel and track line colors of the sparkline will be changed based on theme. For example, for dark theme, the color of datalabel and track line should be white; for light theme, their value should be black. The possible values for sparkline theme are **Material**, **Fabric**, **Bootstrap**, and **Highcontrast**.

The following code example shows the color for datalabel and track line is set to white for dark theme.

[app.component.ts]

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='350px' height='200px'
[axisSettings]= 'axisSettings' theme= 'Highcontrast' [dataLabelSettings]=
'dataLabelSettings' [tooltipSettings]= 'tooltipSettings' [border]= 'border'
[dataSource]="data" fill= '#007dd1' type="Line">
    </ejs-sparkline>`
  })
export class AppComponent {
  public data: object[] = [3, 6, 4, 1, 3, 2, 5] as any;
  public dataLabelSettings: object= { visible: ['All']};
  public tooltipSettings: object= {
    trackLineSettings: { visible: true }
  };
  public axisSettings: object= {
    minX: -1, maxX: 7
  };
  public border: object = { color: 'transparent', width: 2 };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Localization in Angular Sparkline component

The sparkline control supports localization. The default culture for localization is **en-US**. You can change the culture using the **setCulture** method.

<!-- markdownlint-disable MD009 -->

Tooltip format

Sparkline tooltip supports localization. The following code sample shows tooltip text with currency format based on culture.

[app.component.ts]

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-angular-charts'
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-sparkline id='container' width='350px' height='200px'
[containerArea]= 'containerArea' lineWidth= 3 format= 'c0'
useGroupingSeparator= 'true' [padding]='padding' [dataSource]="data" fill=
'#b2cfff' type="Area"
    [tooltipSettings]="tooltipSettings">
    </ejs-sparkline>`
  })
export class AppComponent {
  public data: object[] = [30000, 60000, 40000, 10000, 30000, 20000,
50000] as any;
  public containerArea: object= {
    border: { color: '#033e96', width: 2 }
  };
  public tooltipSettings: object= {
    visible: true
  };
  public padding: object = { left: 20, right: 20, bottom: 20, top: 20};
  public border: object = { color: '#033e96', width: 2 };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Rtl

If you set the `enableRtl` property to true, then the sparkline will be rendered from right-to-left direction.

The following example shows the sparkline is render from "Right-to-left".

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SparklineModule, SparklineTooltipService } from '@syncfusion/ej2-
angular-charts'
import { Component } from '@angular/core';
import { TooltipSettingsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    SparklineModule
  ],
  providers: [SparklineTooltipService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-sparkline id='container' width='150px' height='150px'
[dataSource]="data" enableRtl= true
  >
  </ejs-sparkline>`
})
export class AppComponent {
  public data: object[] = [0, 6, 4, 1, 3, 2, 5] as any;
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Accessibility in Angular Sparkline component

The Sparkline 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 Sparkline 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 Sparkline component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Sparkline component:

- `img` (role)
- `aria-label` (attribute)
- `aria-hidden` (attribute)

Keyboard interaction

The Sparkline 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 Sparkline component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + P** | Prints the Sparkline. |

Ensuring accessibility

The Sparkline component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Sparkline component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Sparkline component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

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

```
<!-- markdownlint-disable MD038 -->
```

Ej1 api migration in Angular Sparkline component

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

Sparkline Types

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

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

|Type| **Property:** `type`

 <ej-sparkline id="container" type='line'></ej-sparkline>|
Property: `type`

 <ejs-sparkline id='container' type="Column"></ejs-sparkline> |

Databinding

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

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

| Datasource | **Property:** `dataSource`
[`dataSource`]=`"dataSource"` | **Property:** `dataSource`
`id="sparkline"` [`dataSource`]=`"dataSource"` |

| Binding X values with datasource | **Property:** `xName`
`xName="xValue"` | **Property:** `xName`
`xName="xValue"` |

| Binding Y values with datasource | **Property:** `yName`
`yName="yValue"` | **Property:** `yName`
`yName="yValue"` |

Markers

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

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

| Enable markers | **Property:** `markerSettings.visible`
[`markerSettings.visible`]=`"true"` | **Property:** `markerSettings.visible`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` | **TS:** `public markerSettings: Object = { visible: ["All"] } |`

| Color | **Property:** `markerSettings.fill`
`markerSettings.fill="red"` | **Property:** `markerSettings.fill`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` | **TS:** `public markerSettings: Object = { fill: "green" } |`

| Size | **Property:** `markerSettings.width`
[`markerSettings.width`]=`10` | **Property:** `markerSettings.size`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` | **TS:** `public markerSettings: Object = { size: 5 } |`

| Opacity | **Property:** `markerSettings.opacity`
[`markerSettings.opacity`]=`0.5` | **Property:** `markerSettings.opacity`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` | **TS:** `public markerSettings: Object = { opacity: 0.5 } |`

| Border color | **Property:** `markerSettings.border.color`
`markerSettings.border.color="green"` | **Property:** `markerSettings.border.color`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` | **TS:** `public markerSettings: Object = { border: { color: "red" } } |`

| Border width | **Property:** `markerSettings.border.width`
[`markerSettings.border.width`]=`1` | **Property:** `markerSettings.border.width`
`id="sparkline"` [`markerSettings`] =`'markerSettings'` |

= 'markerSettings'></ejsparkline>
 TS:
 public markerSettings: Object = { border: { width: 2 } } |

| Border opacity | **Property:** *markerSettings.border.opacity*

 <ejsparkline id="sparkline" [markerSettings.border.opacity]=0.5></ejsparkline> | Not applicable |

Data labels

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

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

| Enable data labels | Not applicable | **Property:** *dataLabelSettings.visible*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { visible: ["All"] } |

| Color | Not applicable | **Property:** *dataLabelSettings.fill*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { fill: "red" } |

| Opacity | Not applicable | **Property:** *dataLabelSettings.opacity*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { opacity: 0.5 } |

| Border color | Not applicable | **Property:** *dataLabelSettings.border.color*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { border: { color: "green" } } |

| Border width | Not applicable | **Property:** *dataLabelSettings.border.width*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { border: { width: 1 } } |

| Format | Not applicable | **Property:** *dataLabelSettings.format*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { format: "{\$xval}: {\$yval}" } |

| Horizontal Offset | Not applicable | **Property:** *dataLabelSettings.offset.x*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { offset: { x: 100 } } |

| Vertical Offset | Not applicable | **Property:** *dataLabelSettings.offset.y*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { offset: { y: 100 } } |

| Font color | Not applicable | **Property:** *dataLabelSettings.textStyle.color*

<ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { textStyle: { color: "green" } } |

| Font family | Not applicable | **Property:** *dataLabelSettings.textStyle.fontFamily*

 <ejsparkline id="sparkline" [dataLabelSettings] = 'dataLabelSettings'></ejsparkline>
 TS:
 public dataLabelSettings: Object = { textStyle: { fontFamily: "Arial" } } |

| Font style | Not applicable | **Property:** *dataLabelSettings.textStyle.fontStyle*
 id="sparkline" [dataLabelSettings] = 'dataLabelSettings' TS: public
 dataLabelSettings: Object = { textStyle: { fontStyle: "normal" } }

| Font weight | Not applicable | **Property:** *dataLabelSettings.textStyle.fontWeight*
 id="sparkline" [dataLabelSettings] = 'dataLabelSettings' TS:
 public dataLabelSettings: Object = { textStyle: { fontWeight: "Bold" } }

| Font opacity | Not applicable | **Property:** *dataLabelSettings.textStyle.opacity*
 id="sparkline" [dataLabelSettings] = 'dataLabelSettings' TS: public
 dataLabelSettings: Object = { textStyle: { opacity: 0.5 } }

| Font size | Not applicable | **Property:** *dataLabelSettings.textStyle.fontSize*
 id="sparkline" [dataLabelSettings] = 'dataLabelSettings' TS: public
 dataLabelSettings: Object = { textStyle: { fontSize: "12px" } }

Range band

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

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

| Color | **Property:** *rangeBandSettings.color*
 id="sparkline" [rangeBandSettings] = "rangeBandSettings" TS: this.rangeBandSettings
 = { color: "#ff14ae" } | **Property:** *rangeBandSettings.color*
 id="sparkline" [rangeBandSettings] = 'rangeBandSettings' TS: public
 rangeBandSettings: Object = { color: "red" } |

| Opacity | **Property:** *rangeBandSettings.opacity*
 id="sparkline" [rangeBandSettings] = "rangeBandSettings" TS: this.rangeBandSettings
 = { opacity: 0.5 } | **Property:** *rangeBandSettings.opacity*
 id="sparkline" [rangeBandSettings] = 'rangeBandSettings' TS: public
 rangeBandSettings: Object = { opacity: 0.5 } |

| Start range | **Property:** *rangeBandSettings.startRange*
 id="sparkline" [rangeBandSettings] = "rangeBandSettings" TS: this.rangeBandSettings
 = { startRange: 0 } | **Property:** *rangeBandSettings.startRange*
 id="sparkline" [rangeBandSettings] = 'rangeBandSettings' TS: public
 rangeBandSettings: Object = { startRange: 0 } |

| End range | **Property:** *rangeBandSettings.endRange*
 id="sparkline" [rangeBandSettings] = "rangeBandSettings" TS: this.rangeBandSettings
 = { endRange: 50 } | **Property:** *rangeBandSettings.endRange*
 id="sparkline" [rangeBandSettings] = 'rangeBandSettings' TS: public
 rangeBandSettings: Object = { endRange: 30 } |

Special points customization

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

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

| High point color | **Property:** *highPointColor*

 <ej-sparkline id="sparkline" highPointColor="blue"></ej-sparkline> | **Property:** *highPointColor*

 <ejs-sparkline id="sparkline" highPointColor="red"></ejs-sparkline> |

| Low point color | **Property:** *lowPointColor*

 <ej-sparkline id="sparkline" lowPointColor="orange"></ej-sparkline> | **Property:** *lowPointColor*

 <ejs-sparkline id="sparkline" lowPointColor="red"></ejs-sparkline> |

| Negative point color | **Property:** *negativePointColor*

 <ej-sparkline id="sparkline" negativePointColor="red"></ej-sparkline> | **Property:** *negativePointColor*

 <ejs-sparkline id="sparkline" negativePointColor="red"></ejs-sparkline> | **Property:** *startPointColor*

 <ejs-sparkline id="sparkline" startPointColor="red"></ejs-sparkline> |

| End point color | **Property:** *endPointColor*

 <ej-sparkline id="sparkline" endPointColor="green"></ej-sparkline> | **Property:** *endPointColor*

 <ejs-sparkline id="sparkline" endPointColor="red"></ejs-sparkline> |

| Tie point color | **Property:** *tiePointColor*

 Not Applicable | **Property:** *tiePointColor*

 <ejs-sparkline id="sparkline" tiePointColor="red"></ejs-sparkline> |

Axis customization

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

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

| Show axis line | **Property:** *axisSettings.visible*

 <ej-sparkline id="sparkline" [axisLineSettings.visible]="true"></ej-sparkline> | **Property:** *axisSettings.lineSettings.visible*

 <ejs-sparkline id="sparkline" [axisSettings]='axisSettings'></ejs-sparkline>
 TS:
 public axisSettings: Object = { visible: true } |

| Line color | **Property:** *axisSettings.color*

 <ej-sparkline id="sparkline" axisLineSettings.color="#ff14ae"></ej-sparkline> | **Property:** *axisSettings.lineSettings.color*

 <ejs-sparkline id="sparkline" [axisSettings]='axisSettings'></ejs-sparkline>
 TS:
 public axisSettings: Object = { color: "red" } |

| Line width | **Property:** *axisSettings.width*

 <ej-sparkline id="sparkline" [axisLineSettings.width]=2></ej-sparkline> | **Property:** *axisSettings.lineSettings.width*

 <ejs-sparkline id="sparkline" [axisSettings]='axisSettings'></ejs-sparkline>
 TS:
 public axisSettings: Object = { width: 2 } |

| Dash array | **Property:** *axisSettings.dashArray*

 <ej-sparkline id="sparkline" axisLineSettings.dashArray="5,3"></ej-sparkline> | **Property:** *axisSettings.lineSettings.dashArray*

 <ejs-sparkline id="sparkline" [axisSettings]='axisSettings'></ejs-sparkline>
 TS:
 public axisSettings: Object = { dashArray: "5,3" } |

| X axis minimum value | Not applicable | **Property:** *axisSettings.minX*

 <ejs-sparkline id="sparkline" [axisSettings]='axisSettings'></ejs-sparkline>
 TS:
 public axisSettings: Object = { minX: 0 } |

|X axis maximum value| Not applicable | **Property:** *axisSettings.maxX*

<ej-sparkline id="sparkline" [axisSettings] ='axisSettings'></ej-sparkline>
 TS:
 public axisSettings: Object = { maxX: 100 }|

|Y axis minimum value| Not applicable | **Property:** *axisSettings.minY*

<ej-sparkline id="sparkline" [axisSettings] ='axisSettings'></ej-sparkline>
 TS:
 public axisSettings: Object = { minY: 0 }|

|Y axis maximum value| Not applicable | **Property:** *axisSettings.maxY*

<ej-sparkline id="sparkline" [axisSettings] ='axisSettings'></ej-sparkline>
 TS:
 public axisSettings: Object = { maxY: 100 }|

|Horizontal axis line position| Not applicable | **Property:** *axisSettings.value*

<ej-sparkline id="sparkline" [axisSettings] ='axisSettings'></ej-sparkline>
 TS:
 public axisSettings: Object = { value: 10 }|

Appearance customization

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

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

|Background color| **Property:** *background*

<ej-sparkline id="sparkline" background="grey"></ej-sparkline> | **Property:** *containerArea.background*

<ej-sparkline id="sparkline" [containerArea]='containerArea'></ej-sparkline>
 TS:
 public containerArea: Object = { background: '#eff1f4' }|

|Border color | Not applicable | **Property:** *containerArea.border.color*

<ej-sparkline id="sparkline" [containerArea]='containerArea'></ej-sparkline>
 TS:
 public containerArea: Object = { border: { color: '#033e96' } }|

|Border width | Not applicable | **Property:** *containerArea.border.width*

<ej-sparkline id="sparkline" [containerArea]='containerArea'></ej-sparkline>
 TS:
 public containerArea: Object = { border: { width: 2 } }|

|Series color| **Property:** *fill*

<ej-sparkline id="sparkline" fill="grey"></ej-sparkline> | **Property:** *fill*

<ej-sparkline id="sparkline" fill="lime"></ej-sparkline>|

|Series opacity| **Property:** *opacity*

<ej-sparkline id="sparkline" [opacity]=0.5></ej-sparkline> | **Property:** *opacity*

<ej-sparkline id="sparkline" opacity=0.5></ej-sparkline>|

|Line series width| **Property:** *width*

<ej-sparkline id="sparkline" [width]=3></ej-sparkline> | **Property:** *lineWidth*

<ej-sparkline id="sparkline" lineWidth=4></ej-sparkline>|

|Series border color| **Property:** *border.color*

<ej-sparkline id="sparkline" ><e-border color="red"></e-border></ej-sparkline> | **Property:** *border.color*

<ej-sparkline id="sparkline" [border]="border"></ej-sparkline> **TS:**
 public: Object = [{ color: "red" }]|

|Series border width| **Property:** *border.width*

<ej-sparkline id="sparkline" ><e-border [width]="1"></e-border></ej-sparkline> | **Property:** *border.width*

<ej-sparkline id="sparkline" [border]="border"></ej-sparkline> **TS:**
 public: Object = [{ width: 2 } |

|Series palette| **Property:** *palette*
`<ej-sparkline id="sparkline" [palette]="palettes"></ej-sparkline>`
TS: `palettes: string[] = ["red", "green", "orange", "blue"]` | **Property:** *palette*
`<ejs-sparkline id="sparkline" [palette]="palettes"></ejs-sparkline>`
TS: `public palettes: string[] = ["red", "green", "orange", "blue"]`

|Theme| **Property:** *theme*
`<ej-sparkline id="sparkline" theme="flatdark"></ej-sparkline>` | **Property:** *theme*
`<ejs-sparkline id="sparkline" theme="Material"></ejs-sparkline>`

|Width| **Property:** *size.width*
`<ej-sparkline id="sparkline" size.width="170px"></ej-sparkline>` | **Property:** *width*
`<ejs-sparkline id="sparkline" width="400px"></ejs-sparkline>`

|Height| **Property:** *size.height*
`<ej-sparkline id="sparkline" size.height="170px"></ej-sparkline>` | **Property:** *height*
`<ejs-sparkline id="sparkline" height="200px"></ejs-sparkline>`

Tooltip

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

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

|Show tooltip| **Property:** *tooltip.visible*
`<ej-sparkline id="sparkline" [tooltip.visible]="true"></ej-sparkline>` | **Property:** *tooltipSettings.visible*
`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>`
TS: `public tooltipSettings: Object = { visible: true }`

|Background| **Property:** *tooltip.fill*
`<ej-sparkline id="sparkline" tooltip.fill="red"></ej-sparkline>` | **Property:** *tooltipSettings.fill*
`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>`
TS: `public tooltipSettings: Object = { fill: "white" }`

|Format| Not applicable | **Property:** *tooltipSettings.format*
`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>`
TS: `public tooltipSettings: Object = { format: "{$xval}: {$yval}" }`

|Template| **Property:** *tooltip.template*
`<ej-sparkline id="sparkline" tooltip.template="tooltip"></ej-sparkline>`
`<div id="tooltip"></div>`
`<div>#point.x</div>`
`<div>#point.y</div>` | **Property:** *tooltipSettings.template*
`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>`
`<div id="tooltip">${x} : ${y}</div>`
TS: `public tooltipSettings: Object = { template: "tooltip" }`

|Font color| **Property:** *tooltip.font.color*
`<ej-sparkline id="sparkline" tooltip.font.color="red"></ej-sparkline>` | **Property:** *tooltipSettings.textStyle.color*
`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>`
TS: `public tooltipSettings: Object = { textStyle: { color: "grey" } }`

|Font opacity| **Property:** *tooltip.font.opacity*
`<ej-sparkline id="sparkline" [tooltip.font.opacity]=0.5></ej-sparkline>` | **Property:** *tooltipSettings.textStyle.opacity*

`<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
`
 public tooltipSettings: Object = { textStyle: { opacity: 0.5 } }

| Font size | **Property:** `tooltip.font.size

` `<ej-sparkline id="sparkline" tooltip.font.size="12px"></ej-sparkline>` | **Property:** `tooltipSettings.textStyle.size

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { textStyle: { size: "12px" } }

| Font family | **Property:** `tooltip.font.fontFamily

` `<ej-sparkline id="sparkline" tooltip.font.fontFamily="Arial"></ej-sparkline>` | **Property:** `tooltipSettings.textStyle.fontFamily

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { textStyle: { fontFamily: "Arial" } }

| Font style | **Property:** `tooltip.font.fontStyle

` `<ej-sparkline id="sparkline" tooltip.font.fontStyle="normal"></ej-sparkline>` | **Property:** `tooltipSettings.textStyle.fontStyle

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { textStyle: { fontStyle: "normal" } }

| Font weight | **Property:** `tooltip.font.fontWeight

` `<ej-sparkline id="sparkline" tooltip.font.fontWeight="bold"></ej-sparkline>` | **Property:** `tooltipSettings.textStyle.fontWeight

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { textStyle: { fontWeight: "bold" } }

| Enable track line | Not applicable | **Property:** `tooltipSettings.trackLineSettings.visible

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { trackLineSettings: { visible: true } }

| Track line color | Not applicable | **Property:** `tooltipSettings.trackLineSettings.color

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { trackLineSettings: { color: "red" } }

| Track line width | Not applicable | **Property:** `tooltipSettings.trackLineSettings.width

` `<ejs-sparkline id="sparkline" [tooltipSettings]='tooltipSettings'></ejs-sparkline>
` **TS:** `
` public tooltipSettings: Object = { trackLineSettings: { width: width } }

Rendering

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

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

| Enable canvas rendering | **Property:** `enableCanvasRendering

` Not Applicable | Not applicable |

Localization

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

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

| Localization | **Property:** *locale*

 <ej-sparkline id="sparkline" locale="en-US"></ej-sparkline> | **Property:** *type*

 <ejs-sparkline id="sparkline" locale="en-US"></ejs-sparkline> |

Methods

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

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

| Dynamically updating sparkline | **Method:** *redraw*

 TS:
 @ViewChild("sparkline") sparkline: EJComponents<any, any>;
 this.sparkline.widget.redraw() | **Method:** *refresh*

 TS:
 @ViewChild("sparkline") public sparkline: SparklineComponent;
 this.sparkline.refresh() |

Events

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

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

| Load | **Event:** *load*

 <ej-sparkline id="sparkline" (load)="load(\$event)"> </ej-sparkline>
 TS:
 public load(args) { } | **Event:** *load*

 <ejs-sparkline id="sparkline" (load)="load(\$event)"></ejs-sparkline>
 TS:
 public load(args) { } |

| Load completed | **Event:** *loaded*

 <ej-sparkline id="sparkline" (loaded)="loaded(\$event)"></ej-sparkline>
 TS:
 public loaded(args) { } | **Event:** *loaded*

 <ejs-sparkline id="sparkline" (loaded)="loaded(\$event)"></ejs-sparkline>
 TS:
 public loaded(args) { } |

| Initialize tooltip | **Event:** *tooltipInitialize*

 <ej-sparkline id="sparkline" (tooltipInitialize)="tooltipInitialize(\$event)"></ej-sparkline>
 TS:
 public tooltipInitialize(args) { } | **Event:** *tooltipInitialize*

 <ejs-sparkline id="sparkline" (tooltipInitialize)="tooltipInitialize(\$event)"></ejs-sparkline>
 TS:
 public tooltipInitialize(args) { } |

| Series rendering | **Event:** *seriesRendering*

 <ej-sparkline id="sparkline" (seriesRendering)="seriesRendering(\$event)"></ej-sparkline>
 TS:
 public seriesRendering(args) { } | **Event:** *seriesRendering*

 <ejs-sparkline id="sparkline" (seriesRendering)="seriesRendering(\$event)"></ejs-sparkline>
 TS:
 public seriesRendering(args) { } |

| Region mouse move | **Event:** *pointRegionMouseMove*

 <ej-sparkline id="sparkline" (pointRegionMouseMove)="pointRegionMove(\$event)"></ej-sparkline>
 TS:
 public pointRegionMove(args) { } | **Event:** *pointRegionMouseMove*

 <ejs-sparkline id="sparkline" (pointRegionMouseMove)="pointRegionMouseMove(\$event)"></ejs-sparkline>
 TS:
 public pointRegionMouseMove(args) { } |

| Region click | **Event:** *pointRegionMouseClick*

 <ej-sparkline id="sparkline" (pointRegionMouseClick)="pointRegionClick(\$event)"></ej-sparkline>
 TS:
 public pointRegionClick(args) { } | **Event:** *pointRegionMouseClick*

 <ejs-sparkline id="sparkline" (pointRegionMouseClick)="pointRegionMouseClick(\$event)"></ejs-sparkline>
 TS:
 public pointRegionMouseClick(args) { } |

| Mouse move | **Event:** *sparklineMouseMove*

 <ej-sparkline id="sparkline"
(sparklineMouseMove)="mouseMove(\$event)"></ej-sparkline>
 TS:
public
mouseMove(args) { } | **Event:** *sparklineMouseMove*

 <ejs-sparkline id="sparkline"
(sparklineMouseMove)="sparklineMouseMove(\$event)"></ejs-sparkline>
 TS:
public
sparklineMouseMove(args) { } |

| Mouse leave | **Event:** *sparklineMouseLeave*

 <ej-sparkline id="sparkline"
(sparklineMouseLeave)="mouseLeave(\$event)"></ej-sparkline>
 TS:
public
mouseLeave(args) { } | Not applicable |

| Click | **Event:** *click*

 <ej-sparkline id="sparkline"
(sparklineMouseClicked)="sparklineMouseClicked(\$event)"></ej-sparkline>
 TS:
public
sparklineMouseClicked(args) { } | **Event:** *sparklineMouseClicked*

 <ejs-sparkline id="sparkline"
(sparklineMouseClicked)="sparklineMouseClicked(\$event)"></ejs-sparkline>
 TS:
public
sparklineMouseClicked(args) { } |

| doubleClick | **Event:** *doubleClick*

 <ej-sparkline id="sparkline"
(doubleClick)="doubleClick(\$event)"></ej-sparkline>
 TS:
public load(args) { } | Not
applicable |

| rightClick | **Event:** *rightClick*

 <ej-sparkline id="sparkline"
(rightClick)="rightClick(\$event)"></ej-sparkline>
 TS:
public rightClick(args) { } | Not
applicable |

| axisRendering | Not applicable | **Event:** *axisRendering*

 <ejs-sparkline id="sparkline"
(axisRendering)="axisRendering(\$event)"></ejs-sparkline>
 TS:
public
axisRendering(args) { } |

| dataLabelRendering | Not applicable | **Event:** *dataLabelRendering*

 <ejs-sparkline
id="sparkline" (dataLabelRendering)="dataLabelRendering(\$event)"></ejs-sparkline>
 TS:

public dataLabelRendering(args) { } |

| markerRendering | Not applicable | **Event:** *markerRendering*

 <ejs-sparkline id="sparkline"
(markerRendering)="markerRendering(\$event)"></ejs-sparkline>
 TS:
public
markerRendering(args) { } |

| pointRendering | Not applicable | **Event:** *pointRendering*

 <ejs-sparkline id="sparkline"
(pointRendering)="pointRendering(\$event)"></ejs-sparkline>
 TS:
public load(args) { } |

| resize | Not applicable | **Event:** [Link to the Video](#)

 <ej-sparkline id="sparkline"
(resize)="resize(\$event)"></ej-sparkline>
 TS:
public resize(args) { } |

Speed Dial

Getting started with Angular Speed dial component

This section explains how to create a simple SpeedDial and demonstrate the basic usage of the SpeedDial component in an Angular environment.

To get started quickly with Angular Speed Dial component, you can check out this video:

Dependencies

The list of dependencies required to use the SpeedDial component in your application is given as follows:

```
`js
|-- @syncfusion/ej2-angular-buttons
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
`,`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`
npm install -g @angular/cli
`,`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`
ng new my-app
cd my-app
`,`
```

Installing Syncfusion Buttons package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#).
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-buttons](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons --save
`,`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the ngcc package use the below.

Add [@syncfusion/ej2-angular-buttons@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-buttons@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-buttons:"20.3.0.47-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding SpeedDial module

Import SpeedDial module into Angular application(`app.module.ts`) from the package

`@syncfusion/ej2-angular-buttons`.

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Imported Syncfusion SpeedDial module from buttons package.
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons';
import { AppComponent } from './app.component';
@NgModule({
  imports: [ BrowserModule, SpeedDialModule ], // Registering EJ2 SpeedDial Module.
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Adding Syncfusion SpeedDial component

Modify the template in `app.component.ts` file to render the SpeedDial component and define the action items using [items](#) property.

```
`typescript
import { Component } from '@angular/core';
```



```

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
  <button ej-speeddial id='element' content='Edit' [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
}
`

```

Adding CSS reference

Add SpeedDial component's styles as given below in `style.css`.

```

`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
`

```

Running the application

Run the application in the browser using the following command:

```

`
ng serve
`

```

The below example shows a basic SpeedDial component,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    SpeedDialModule // Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',

```

```

    template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
      <!-- To Render SpeedDial component. -->
      <button ejs-speeddial id="speeddial" content='Edit'
target='#targetElement' [items]='items'></button>`
  })
  export class AppComponent {
    public items: SpeedDialItemModel[] = [
      { text: 'Cut' },
      { text: 'Copy' },
      { text: 'Paste' }
    ];
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: You can refer to our [Angular Speed Dial](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Speed Dial example](#) that shows you how to render the Speed Dial in Angular.

Items in Angular Speed dial component

The Angular Speed Dial action items can be added by using [items](#) property.

| Fields | Type | Description |

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

| [text](#) | string | Defines the text content of SpeedDialItem. |

| [iconCss](#) | string | Defines one or more CSS classes to include an icon or image in Speed Dial item. |

| [disabled](#) | boolean | Defines whether to enable or disable the SpeedDialItem. |

| [id](#) | string | Defines a unique value for the SpeedDialItem which can be used to identify the item in event args. |

| [title](#) | string | Defines the title of SpeedDialItem to display tooltip. |

Icons in Speed Dial items

You can customize the icon and text of Speed Dial action items using [iconCss](#) and [text](#) properties.

Icon only

You can show icon only in SpeedDial items by setting [iconCss](#) property. You can show tooltip on hover to show additional details to end-user by setting [title](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';

```

```
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement'
[items]='items'></button>`
  })
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss:'e-icons e-cut'},
    { iconCss:'e-icons e-copy'},
    { iconCss:'e-icons e-paste'}
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Text only

You can show only text in Speed Dial items by setting [text](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" content='Edit'
target='#targetElement' [items]='items'></button>`
  })
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut' },
```

```

    { text: 'Copy' },
    { text: 'Paste' }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Icon with text

You can show icon along with text in Speed Dial items by setting [iconCss](#) and [text](#) properties.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" content='Edit' openIconCss='e-icons e-edit' closeIconCss='e-icons e-close' target='#targetElement' [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut', iconCss: 'e-icons e-cut' },
    { text: 'Copy', iconCss: 'e-icons e-copy' },
    { text: 'Paste', iconCss: 'e-icons e-paste' }
  ];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Disabled

You can disable Speed Dial items by setting [disabled](#) property as `true`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" content='Edit'
target='#targetElement' [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut', disabled: true },
    { text: 'Copy' },
    { text: 'Paste' }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Animation

The Speed Dial items can be animated during the opening and closing of the popup action items. You can customize the animation's effect, delay, and duration by setting [animation](#) property. By default, Speed Dial animates with a [fade](#) effect and supports all [speeddialanimation](#) effects.

Below example demonstrates the Speed Dial items with applied Zoom effect.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialAnimationSettingsModel, SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
```

```

    selector: 'app-root',
    template: `<div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit'
content='Edit' closeIconCss='e-icons e-close' target='#targetElement'
[items]='items' [animation]= 'animation'></button>`
  })
  export class AppComponent {
    public items: SpeedDialItemModel[] = [
      { text: 'Cut', iconCss: 'e-icons e-cut' },
      { text: 'Copy', iconCss: 'e-icons e-copy' },
      { text: 'Paste', iconCss: 'e-icons e-paste' }
    ];
    public animation: SpeedDialAnimationSettingsModel =
      { effect: 'Zoom' };
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template

The Speed Dial supports to customize the action items and entire pop-up container by setting [itemTemplate](#) and [popupTemplate](#) properties. For more details about templates, check out the link [here](#).

Positions in Angular Speed dial component

The Speed dial control can be positioned anywhere on the [target](#) using the [position](#) property. If the [target](#) is not defined, then Speed Dial is positioned based on the browser viewport.

The position values of Speed Dial are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft
- BottomCenter
- BottomRight

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';

```

```
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.

  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" content='Add' position='BottomLeft'
    target='#targetElement'></button>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

INDEX.CSS

```
/* Represents the style for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
```

Opens items on hover

You can open the Speed Dial action items on mouse hover by setting the [opensOnHover](#) property.

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
  <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit' closeIconCss='e-icons e-close'
  target='#targetElement' [items]='items' [opensOnHover]='true'></button>`
})

export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
```

```
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' }
];
}
```

Programmatically show/hide Speed Dial items

You can open/close the Speed Dial action items programmatically using [show](#) and [hide](#) methods.

Below example demonstrates open/close action items on button click.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, HostListener } from '@angular/core';
import { SpeedDialComponent, SpeedDialItemModel } from '@syncfusion/ej2-
angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-
height:340px;border:1px solid;padding:10px;">
  <button id="show" (click)="show()">Show</button>
  <button id="hide" (click)="hide()">Hide</button><div>
  <!-- To Render SpeedDial component. -->
  <button ejs-speeddial #speeddial openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' [opensOnHover]=
'true' [items]='items'></button>`
})
export class AppComponent {
  @ViewChild('speeddial')
  public speeddialObj :SpeedDialComponent | any;
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
  public show() {
    this.speeddialObj.show();
  }
  public hide() {
    this.speeddialObj.hide();
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```



```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

INDEX.CSS

```
/* Represents the styles for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
/* Represents the styles for button */
#hide{
  float: right;
}
```

Programmatically refresh the position

You can refresh the position of the Speed Dial using [refreshPosition](#) method when the **target** position is changed.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, HostListener } from '@angular/core';
import { SpeedDialComponent, SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';

@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:340px;border:1px solid;padding:10px;">
    <button id="refresh" (click)="refresh()">Refresh</button><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial #speeddial openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' position='MiddleRight'
target='#targetElement' [items]='items'></button>`
})
export class AppComponent {
  @ViewChild('speeddial')
  public speeddialObj:SpeedDialComponent | any;
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
};
```

```

public refresh() {
    document.getElementById("targetElement")!.style.minHeight = "300px";
    this.speeddialObj.refreshPosition();
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

/* Represents the styles for loader */
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}

```

Display modes in Angular Speed dial component

The action items in [Angular Speed Dial](#) can be displayed in **Linear** and **Radial** display modes by setting [mode](#) property.

Linear display mode

In **Linear** display mode, Speed Dial action items are displayed in a list-like format either horizontally or vertically. By default, Speed Dial items are displayed in **Linear** mode.

Direction

You can open the action items on the top, left, up, and down side of the Speed Dial button by setting [direction](#) property. The default value is **Auto** where the action items are displayed based on the [position](#) of the Speed Dial.

The **Linear** directions of Speed Dial are as follows:

- Left - Action items are displayed on the left side of the button.
- Right - Action items are displayed on the right side of the button.
- Up - Action items are displayed on the top of the button.
- Down - Action items are displayed on the bottom of the button.
- Auto - Action items display direction auto calculated based on **position** of the Speed Dial. If Speed Dial is position at bottom right, then action items displayed at top.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons';

```

```
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ej-speeddial id="speeddial" openIconCss='e-icons e-edit'
    closeIconCss='e-icons e-close' target='#targetElement' [items]='items' mode=
    'Linear' direction='Left'></button>`
  })
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Radial display mode (Radial Menu)

In **Radial** mode, Speed Dial action items are displayed in a circular pattern like a radial menu. For more details about radial mode, check out the link [here](#).

Radial menu in Angular Speed dial component

The Angular Speed Dial action items can be displayed in a circular pattern like a radial menu by setting [mode](#) property. You can customize the [direction](#), [startAngle](#), [endAngle](#) and [offset](#) by setting [radialSettings](#) property.

Radial menu direction

You can open the action items in either clockwise or anticlockwise by setting [direction](#) property. The default value is [Auto](#) where the action items are displayed based on the [position](#) property of the Speed Dial.

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
```

```

<button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit' closeIconCss='e-icons e-close'
target='#targetElement' [items]='items' mode= 'Radial' [radialSettings]= 'radialSettings'></button>`
})
export class AppComponent {
public items: SpeedDialItemModel[] = [
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' },
{ iconCss: 'e-icons e-edit' },
{ iconCss: 'e-icons e-save' }
];
public radialSettings: RadialSettingsModel = { direction: 'AntiClockwise' };
}
`

```

Radial menu start and end angle

You can modify the start and end angle of action items by setting [startAngle](#) and [endAngle](#) properties. If the angle is not defined, the action items are displayed based on the **position** property of the Speed Dial.

```

`typescript
import { Component } from '@angular/core';

@Component({
selector: 'app-root',
template: `<!-- To Render SpeedDial component. -->
<button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit' closeIconCss='e-icons e-close'
target='#targetElement' [items]='items' mode= 'Radial' position='MiddleCenter' [radialSettings]=
'radialSettings'></button>`
})
export class AppComponent {
public items: SpeedDialItemModel[] = [
{ iconCss: 'e-icons e-cut' },
{ iconCss: 'e-icons e-copy' },
{ iconCss: 'e-icons e-paste' },
{ iconCss: 'e-icons e-edit' },
{ iconCss: 'e-icons e-save' }
];
}
`

```

```
public radialSettings: RadialSettingsModel = { direction: 'AntiClockwise', startAngle: 180, endAngle: 360
};
}
,
```

Offset

You can modify the offset distance between action items and Speed Dial button using [offset](#) property.

`typescript

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
<button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit' closeIconCss='e-icons e-close'
target='#targetElement' [items]='items' mode= 'Radial' [radialSettings]= 'radialSettings'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
  public radialSettings: RadialSettingsModel = { offset: '80px' };
}
```

Below example demonstrates the radial menu settings of the Speed Dial.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { RadialSettingsModel, SpeedDialItemModel } from '@syncfusion/ej2-
angular-buttons';
@Component({
  imports: [
    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"><div>
```

```

    <!-- To Render SpeedDial component. -->
    <button ej-speeddial id="speeddial" openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' [items]='items' mode=
'Radial' position= 'MiddleRight' [radialSettings]=
'radialSettings'></button>`
  })
  export class AppComponent {
    public items: SpeedDialItemModel[] = [
      { iconCss: 'e-icons e-cut' },
      { iconCss: 'e-icons e-copy' },
      { iconCss: 'e-icons e-paste' },
      { iconCss: 'e-icons e-edit' },
      { iconCss: 'e-icons e-save' }
    ];
    public radialSettings: RadialSettingsModel = { offset: '80px', direction:
'AntiClockwise', startAngle: 90, endAngle: 270 };
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template in Angular Speed dial component

This section explains available templates in SpeedDial component and its usage.

Item template

You can use the [itemTemplate](#) property to set a template content for the SpeedDial items. The template content is defined as a child content of `itemTemplate` tag directive.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <!-- To Render SpeedDial component with icon and text -->
    <button ej-speeddial id="speeddial" content='Edit' openIconCss='e-icons
e-edit' target='#targetElement' [items]=items [itemTemplate]="itemTemplate">
    <ng-template #itemTemplate let-items="">
      <div class="itemlist">
        <span class="icon" {{items.iconCss}} style="padding:3px"></span>
        <span class="text">{{items.text}}</span>
      </div>
    </ng-template>
  `
})

```

```

        </div>
    </ng-template></button>`
    })
    export class AppComponent {
        public items: SpeedDialItemModel[] = [
            { text: 'Cut', iconCss: 'e-icons e-cut' },
            { text: 'Copy', iconCss: 'e-icons e-copy' },
            { text: 'Paste', iconCss: 'e-icons e-paste' }
        ];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

/* Represents the styles for loader */
#loader {
    color: #008cff;
    height: 40px;
    left: 45%;
    position: absolute;
    top: 45%;
    width: 30%;
}
/* Represents the styles for speeddial list items */
.itemlist .text {
    padding: 0 5px;
}
.e-speeddial-li .itemlist {
    display: inherit;
    width: 100%;
    border: 1px solid transparent;
    align-items: center;
    padding: 5px;
    border-radius: 500px;
    background-color: rgba(104, 99, 104, 0.1);
    box-shadow: 0 0 4px grey;
}
.e-speeddial-li .itemlist:hover {
    background-color: rgb(224, 224, 224);
}

```

Popup template

You can use the [popupTemplate](#) property to set a template content for popup of SpeedDial component. The template content is defined as a child content of `popupTemplate` tag directive.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

```

```

import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
<!-- To Render SpeedDial component with icon and text -->
<button ej-speeddial id="speeddial" content='Edit' openIconCss='e-icons e-edit' target='#targetElement' [popupTemplate]="popupTemplate">
  <ng-template #popupTemplate>
    <div>
      <div class="speeddial-form">
        <p>Here you can customize your code.</p>
      </div>
    </div>
  </ng-template>
</button>`
})
export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

/* Represents the styles for loader */
#loader {
  color: #008cff;
  height: 40px;
  left: 45%;
  position: absolute;
  top: 45%;
  width: 30%;
}
/* Represents the styles for speeddial-form */
.speeddial-form {
  width: 200px;
  height: 80px;
  text-align: center;
  border-radius: 15px;
  box-shadow: rgb(0 0 0 / 10%) 0px 10px 15px -3px, rgb(0 0 0 / 5%) 0px 4px 6px -2px;
  background: #f5f5f5;
  padding: 15px;
}

```


Styles in Angular Speed dial component

This section briefs different ways to style SpeedDial component.

SpeedDial button

You can customize the icon and text of Angular Speed Dial button using [openIconCss](#), [closeIconCss](#) and [content](#) properties.

Icon only

You can use the [openIconCss](#) and [closeIconCss](#) properties to show icons in speed dial button. You can also show tooltip on hover to show additional details to end-user by setting `title` attribute.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
    <!-- To Render SpeedDial component with icon only -->
    <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit' target="#targetElement"></button>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Text only

You can show only text in Speed Dial button by setting [content](#) property without setting icon properties..

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
```

```

    selector: 'app-root',
    template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
      <!-- To Render SpeedDial component with text only -->
      <button ejs-speeddial id="speeddial" content='Edit'
target='#targetElement'></button>`
  })
  export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Icon with text

You show icon and text in SpeedDial button using [openIconCss](#), [closeIconCss](#) and [content](#) properties together.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
    <!-- To Render SpeedDial component with icon and text -->
    <button ejs-speeddial id="speeddial" content='Edit' openIconCss='e-
icons e-edit' target='#targetElement'></button>`
  })
  export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Disabled

You can enable or disable the SpeedDial component using [disabled](#) property.

```
`typescript
```

```
import { Component } from '@angular/core';
```

```

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component in disabled state -->
  <button ej-speeddial id="speeddial" content='Edit' [disabled]='true'></button>`
})
export class AppComponent { }
`

```

cssClass

The Angular Speed Dial supports the following predefined styles that can be defined using the [cssClass](#) property. You can customize by setting 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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
  <!-- To Render SpeedDial component with applied warning style -->
  <button ej-speeddial id="speeddial" content='Edit' cssClass='e-warning' target='#targetElement'></button>`
})
export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Visible

You can set the Speed Dial button to visible/hidden state using [visible](#) property.

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component in hidden state -->
<button ej-speeddial id="speeddial" content='Edit' [visible]='false'></button>`
})
export class AppComponent { }
`
```

Tooltip

You can show tooltip on hover to show additional details to end-user by setting [title](#) to Speed Dial button.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    SpeedDialModule // Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
  <!-- To Render SpeedDial component -->
  <button ej-speeddial id="speeddial" openIconCss='e-icons e-edit'
  [items]='items' target='#targetElement'></button>`
})
export class AppComponent {
  // Initialize action items with tooltip
  public items: SpeedDialItemModel[] = [
    { iconCss:'e-icons e-cut', title:'Cut' },
    { iconCss:'e-icons e-copy', title:'Copy' },
    { iconCss:'e-icons e-paste', title:'Paste' }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Opens on hover

You can use [opensOnHover](#) property to open actions items on hover itself. By default action items displayed only when clicking the speed dial button.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.

  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"><div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit'
closeIconCss='e-icons e-close' target='#targetElement' [items]='items'
[opensOnHover]= 'true'></button>`
  })
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss: 'e-icons e-cut' },
    { iconCss: 'e-icons e-copy' },
    { iconCss: 'e-icons e-paste' }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Modal in Angular Speed dial component

You can use the [modal](#) property to set the Speed Dial as modal which adds an overlay to prevent the background interaction.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
```

```
import { SpeedDialItemModel } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    SpeedDialModule // Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-height:350px;border:1px solid;"></div>
    <!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" openIconCss='e-icons e-edit'
[modal]='true' target='#targetElement' [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { iconCss:'e-icons e-cut'},
    { iconCss:'e-icons e-copy'},
    { iconCss:'e-icons e-paste'}
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Event in Angular Speed dial component

This section explains the [Angular Speed Dial](#) events that will be triggered when appropriate actions are performed.

clicked

The SpeedDial component triggers the [clicked](#) event with [SpeedDialItemEventArgs](#) argument when an action item is clicked. You can use this event to perform the required action.

`typescript

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
    <button ejs-speeddial id="speeddial" content='Edit' (clicked)="clicked($event)"
[items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
```

```

{ text: 'Copy'},
{ text: 'Paste'}
];
public clicked(args: SpeedDialItemEventArgs) {
//Your required action here
}
}
`

```

created

The Speed Dial component triggers the [created](#) event when SpeedDial component rendering is completed.

`typescript

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
<button ejs-speeddial id="speeddial" content='Edit' (created)='created()' [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
  public created() {
//Your required action here
}
}
`

```

beforeOpen

The SpeedDial component triggers the [beforeOpen](#) event with [SpeedDialBeforeOpenCloseEventArgs](#) argument before the SpeedDial popup is opened.

`typescript

```

import { Component } from '@angular/core';

```

```

@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
  <button ejs-speeddial id="speeddial" content='Edit' (beforeOpen)='beforeOpen($event)'
  [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
  public beforeOpen(args: SpeedDialBeforeOpenCloseEventArgs) {
    //Your required action here
  }
}
`

```

onOpen

The SpeedDial component triggers the [onOpen](#) event with [SpeedDialOpenCloseEventArgs](#) argument when SpeedDial popup is opened.

`typescript

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
  <button ejs-speeddial id="speeddial" content='Edit' (onOpen)='onOpen($event)'
  [items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
}

```



```
public onOpen(args: SpeedDialOpenCloseEventArgs) {
//Your required action here
}
}
```

beforeClose

The SpeedDial component triggers the [beforeClose](#) event with [SpeedDialBeforeOpenCloseEventArgs](#) argument before the SpeedDial popup is closed.

`typescript

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
<button ejs-speeddial id="speeddial" content='Edit' (beforeClose)='beforeClose($event)'
[items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
  public beforeOpen(args: SpeedDialBeforeOpenCloseEventArgs) {
//Your required action here
}
}
```

onClose

The SpeedDial component triggers the [onClose](#) event with [SpeedDialOpenCloseEventArgs](#) argument when SpeedDial popup is closed.

`typescript

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render SpeedDial component. -->
```

```

<button ejs-speeddial id="speeddial" content='Edit' (onClose)='onClose($event)'
[items]='items'></button>`
})
export class AppComponent {
public items: SpeedDialItemModel[] = [
{ text: 'Cut'},
{ text: 'Copy'},
{ text: 'Paste'}
];
public onClose(args: SpeedDialOpenCloseEventArgs) {
//Your required action here
}
}
`

```

beforeItemRender

The SpeedDial component triggers the [beforeItemRender](#) event with [SpeedDialItemEventArgs](#) argument for each **SpeedDialItem** once it is rendered.

```

`typescript
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
template: `<!-- To Render SpeedDial component. -->
<button ejs-speeddial id="speeddial" content='Edit' (beforeItemRender)='beforeItemRender($event)'
[items]='items'></button>`
})
export class AppComponent {
public items: SpeedDialItemModel[] = [
{ text: 'Cut'},
{ text: 'Copy'},
{ text: 'Paste'}
];
public beforeItemRender(args: SpeedDialItemEventArgs) {
//Your required action here
}
}
`

```

```
}
,
```

Below example demonstrates the clicked event of the Speed Dial component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpeedDialModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
import { SpeedDialItemEventArgs, SpeedDialItemModel } from '@syncfusion/ej2-
angular-buttons';
@Component({
  imports: [

    SpeedDialModule// Registering EJ2 SpeedDial Module.
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div id="targetElement" style="position:relative;min-
height:350px;border:1px solid;"></div>
    <!-- To Render SpeedDial component with bind clicked event. -->
    <button ej2-speeddial id="speeddial" content='Edit'
(clicked)='clicked($event)' target='#targetElement'
[items]='items'></button>`
})
export class AppComponent {
  public items: SpeedDialItemModel[] = [
    { text: 'Cut' },
    { text: 'Copy' },
    { text: 'Paste' }
  ];
  public clicked(args: SpeedDialItemEventArgs) {
    alert(args.item.text + ' is clicked');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Accessibility in Angular Speed dial component

The Speed Dial 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 Speed Dial 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 Speed Dial component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Speed Dial component:

| Attributes | Purpose |

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

| `role=menu` | Specifies that the Speed Dial item has a submenu. |

| `role=menuitem` | Indicates an actionable item within the Speed Dial submenu. |

- | `aria-label` | Indicates the Speed Dial Popup item text. |
- | `aria-expanded` | It indicates whether the Speed Dial current state is expanded or collapsed. |
- | `aria-haspopup` | It indicates whether the Speed Dial has popup items or not. |
- | `aria-controls` | Attribute is set to the Speed Dial button and it points to the corresponding content. |
- | `aria-disabled` | Indicates the state of menu item whether it is disabled. |

Keyboard interaction

The Speed Dial 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 Speed Dial component.

- | **Press** | **To do this** |
- |-----|-----|
- | **Enter** | **Open/close the menu.** |
- | **Up-arrow** | **Focuses the next menu item.** |
- | **Left-Arrow** | **Focuses the previous menu item.** |
- | **Down-Arrow** | **Focuses the previous menu item.** |
- | **Right-Arrow** | **Focuses the next menu item.** |
- | **Home** | **Focuses the first menu item.** |
- | **End** | **Focuses the last menu item.** |
- | **Esc** | **Closes the menu.** |

Ensuring accessibility

The Speed Dial component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Speed Dial component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Speed Dial component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Spinner

Getting started with Angular Spinner component

This section explains the steps to create a simple **Spinner** and configure its functionalities in Angular.

Getting Started with Angular CLI

The following section explains the steps required to create a simple angular-cli application and how to configure the 'Spinner'.

Prerequisites

To get started with Syncfusion Angular UI Components, make sure that you have compatible versions of Angular and TypeScript.

- Angular : 6+
- TypeScript : 2.6+

Setting up an Angular project

Angular provides an easiest way to setup project using Angular CLI [Angular CLI](#) tool.

Install the CLI application globally in your machine.

```
`javascript
npm install -g @angular/cli
`
```

Create a new application

```
`javascript
ng new syncfusion-angular-app
`
```

Once you have executed the above command you may ask for following options,

- Would you like to add Angular routing?
- Which stylesheet format would you like to use?

By default, it install the CSS style base application. To setup with SCSS, pass `--style=SCSS` argument on create project.

Example code snippet.

```
`javascript
ng new syncfusion-angular-app --style=SCSS
`
```

Use below command to Navigate into the created project folder.

```
`javascript
cd syncfusion-angular-app
`
```

Installing Syncfusion Popups package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-popups](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-popups --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-popups@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-popups@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-popups:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding CSS reference

The following CSS files can be referenced in `[src/styles.css]` file.

```
`css
@import '../node_modules/@syncfusion/ej2-angular-popups/styles/material.css';
`
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Adding Spinner

Initialize the Spinner using "createSpinner" method and show/hide the spinner using "showSpinner" and "hideSpinner" methods accordingly. Set the target to the spinner to render it based on specific target.

The following steps explains you on how to create and how to show/hide your Spinner.

- Import the "createSpinner" method from "ej2-popups" library into your file as shown in below.

```
`typescript
```

```
import { createSpinner } from '@syncfusion/ej2-angular-popups';
```

```
,
```

- Show and hide this spinner by using "showSpinner" and "hideSpinner" methods for loading in your page and import them in your file as shown in below.

```
`typescript
```

```
import { showSpinner, hideSpinner } from '@syncfusion/ej2-popups';
```

```
,
```

Create the Spinner globally

The Spinner can be render globally in a page using public exported functions of the "ej2-popups" package.

```
`typescript
```

```
import { showSpinner, hideSpinner } from '@syncfusion/ej2-angular-popups';
```

```
,
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
import { createSpinner, showSpinner, hideSpinner } from '@syncfusion/ej2-
angular-popups';
@Component({
  imports: [

    FormsModule,
  ],
  providers: [],
  standalone: true,
  selector: 'app-container',
  template: `<div id="container"></div>`
})
export class AppComponent {
  constructor() {}
  ngAfterViewInit() {
    //createSpinner() method is used to create spinner
    createSpinner({
      // Specify the target for the spinner to show
      target: document.getElementById('container') as any
    });
    // showSpinner() will make the spinner visible
    showSpinner((document as any).getElementById('container'));
    setInterval(function() {
      // hideSpinner() method used hide spinner
      hideSpinner((document as any).getElementById('container'));
    }, 10000);
  }
}
```



```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template in Angular Spinner component

You can use custom templates on the Spinner instead of the default Spinner by specifying the template in the `setSpinner` method.

The following steps explain you on how to define template for Spinner.

- Import the `setSpinner` method from `ej2-angular-popups` library into your `app.component.ts` as shown in below.

```
`typescript

```

```
import { setSpinner } from '@syncfusion/ej2-angular-popups';

```

- Pass your custom template to the `setSpinner` method like as below.

```
`typescript

```

```
// Specify the template content to be displayed in the Spinner

```

```
setSpinner({ template: '<div style="width:100%;height:100%" class="custom-rolling"><div></div></div>'});

```

You should set the template to the Spinner before creating the respective Essential JS 2 component.

Also, until we replace `setSpinner` template, the further Essential JS 2 component rendering is created with given template only.

- Now, render the Essential JS 2 component. It's render the Spinner with the template specified in the `setSpinner` method.

In the below sample, we have rendered the Grid component with custom Spinner using `setSpinner` method.

You have to define the styles for the template in `index.css`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'

```

```

import { GridModule, PageService, SortService, FilterService, GroupService }
  from '@syncfusion/ej2-angular-grids';
import { Component, ViewChild, OnInit } from '@angular/core';
import { GridComponent } from '@syncfusion/ej2-angular-grids';
import { setSpinner } from '@syncfusion/ej2-angular-popups';
import { data } from './datasource';
@Component({
  imports: [

    FormsModule,
    GridModule
  ],
  providers: [
    PageService,
    SortService,
    FilterService,
    GroupService
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-grid #grid1 [dataSource]='gridData'
    (created)='onFirstGridCreated()'>
      <e-columns>
        <e-column field='OrderID' headerText='Order ID'
        textAlign='right' width=120 type='number'></e-column>
        <e-column field='CustomerID' headerText='Customer ID'
        width=140 type='string'></e-column>
        <e-column field='Freight' headerText='Freight'
        textAlign='right' format='C' width=120></e-column>
        <e-column field='OrderDate' headerText='Order Date'
        width=120 format='yMd' width=140></e-column>
      </e-columns>
    </ejs-grid>
    <br/>
    <br/>
    <ejs-grid #grid2 [dataSource]='gridData'
    (created)='onSecondGridCreated()'>
      <e-columns>
        <e-column field='OrderID' headerText='Order ID'
        textAlign='right' width=120 type='number'></e-column>
        <e-column field='CustomerID' headerText='Customer ID'
        width=140 type='string'></e-column>
        <e-column field='Freight' headerText='Freight'
        textAlign='right' format='C' width=120></e-column>
        <e-column field='OrderDate' headerText='Order Date'
        width=120 format='yMd' width=140></e-column>
      </e-columns>
    </ejs-grid>
  `
})
export class AppComponent implements OnInit {
  @ViewChild('grid1') grid_1: GridComponent | any;
  @ViewChild('grid2') grid_2: GridComponent | any;
  public gridData?: Object[];
  constructor() {}
  public ngOnInit(): void {

```

```

        this.gridData = data.slice(0, 7);
    }
    public onFirstGridCreated(): void {
        this.grid_1.hideSpinner = () => true;
        setSpinner({ template: '<div style="width:100%;height:100%"
class="custom-rolling"><div></div></div>' });
    }
    public onSecondGridCreated(): void {
        this.grid_2.hideSpinner = () => true;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

INDEX.CSS

```

#container {
    visibility: hidden;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
#wrapper {
    margin: 0 auto;
    padding-top: 20px;
}
/* csslint ignore:start */
@keyframes custom-rolling {
    0% {
        -webkit-transform: translate(-50%, -50%) rotate(0deg);
        transform: translate(-50%, -50%) rotate(0deg);
    }
    100% {
        -webkit-transform: translate(-50%, -50%) rotate(360deg);
        transform: translate(-50%, -50%) rotate(360deg);
    }
}
@-webkit-keyframes custom-rolling {
    0% {
        -webkit-transform: translate(-50%, -50%) rotate(0deg);
        transform: translate(-50%, -50%) rotate(0deg);
    }
    100% {
        -webkit-transform: translate(-50%, -50%) rotate(360deg);
        transform: translate(-50%, -50%) rotate(360deg);
    }
}

```

```

}
.custom-rolling {
  position: relative;
}
.custom-rolling div,
.custom-rolling div:after {
  border: 16px solid #51CACC;
  border-radius: 50%;
  border-top-color: transparent;
  height: 160px;
  position: absolute;
  width: 160px;
}
.custom-rolling div {
  -webkit-animation: custom-rolling 1.3s linear infinite;
  animation: custom-rolling 1.3s linear infinite;
  top: 100px;
  left: 100px;
}
.custom-rolling div:after {
  -ms-transform: rotate(90deg);
  -webkit-transform: rotate(90deg);
  transform: rotate(90deg);
}
.custom-rolling {
  -webkit-transform: translate(-31px, -31px) scale(0.31) translate(31px, 31px);
  height: 62px !important;
  transform: translate(-31px, -31px) scale(0.31) translate(31px, 31px);
  width: 62px !important;
}
/* csslint ignore:end */

```

Types in Angular Spinner component

By default, the Spinner is loaded in the applicable Essential JS 2 component based on the theme imported into the page. Based on the theme, the type is set to the Spinner.

The available types are:

- Material
- Fabric
- Bootstrap

You can change the Essential JS 2 component spinner type by passing the type of the spinner as parameter to the `setSpinner` method like as below.

`typescript

// Specify the type of the Spinner to be displayed

setSpinner({ type: 'Bootstrap'});

`

After Essential JS 2 component creation only, you can change the Essential JS 2 component spinner type.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { GridModule, PageService, SortService, FilterService, GroupService }
from '@syncfusion/ej2-angular-grids'
import { Component, ViewChild, OnInit } from '@angular/core';
import { GridComponent } from '@syncfusion/ej2-angular-grids';
import { setSpinner } from '@syncfusion/ej2-angular-popups';
import { data } from '../datasource';
@Component({
  imports: [

    FormsModule,
    GridModule
  ],
  providers: [
    PageService,
    SortService,
    FilterService,
    GroupService
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <h5> Grid is rendered with Bootstrap type Spinner </h5>
    <ejs-grid #grid [dataSource]='gridData' (created)='gridCreated()'>
      <e-columns>
        <e-column field='OrderID' headerText='Order ID'
textAlign='right' width=120 type='number'></e-column>
        <e-column field='CustomerID' headerText='Customer ID'
width=140 type='string'></e-column>
        <e-column field='Freight' headerText='Freight'
textAlign='right' format='C' width=120></e-column>
        <e-column field='OrderDate' headerText='Order Date'
width=120 format='yMd' width=140></e-column>
      </e-columns>
    </ejs-grid>
  `
})
export class AppComponent implements OnInit {
  @ViewChild('grid') grid: GridComponent | any;
  public gridData?: Object[];
  constructor() {}
  public ngOnInit(): void {
    this.gridData = data.slice(0, 7);
  }
  public gridCreated(): void {
    this.grid.hideSpinner = () => true;
    setSpinner({ type: 'Bootstrap' });
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Style in Angular Spinner component

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 spinner

Use the following CSS to customize the spinner stroke color.

Material theme

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-material {
stroke: green;
}
```

,

Fabric theme

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-fabric {
stroke: green;
}
```

,

Bootstrap theme

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap {
fill: green;
stroke: green;
}
```

,

Bootstrap4 theme

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap4 {
stroke: green;
}
```

,

High Contrast theme

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-high-contrast .e-path-arc {
stroke: green;
}
,
```

SplitButton

Getting started with Angular Split button component

This section explains how to create a simple SplitButton and demonstrate the basic usage of the SplitButton component in an Angular environment.

Dependencies

The list of dependencies required to use the SplitButton component in your application is given as follows:

```
`js
|-- @syncfusion/ej2-angular-splitbuttons
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
,
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
npm install -g @angular/cli
,
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
ng new my-app
cd my-app
,
```

Installing Syncfusion SplitButton package

Syncfusion packages are distributed in npm as **@syncfusion** scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-splitbuttons](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-splitbuttons --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-splitbuttons@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-splitbuttons@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-splitbuttons:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding SplitButton module

Import SplitButton module into Angular application(app.module.ts) from the package

[@syncfusion/ej2-angular-splitbuttons](#).

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Imported Syncfusion splitbutton module from splitbuttons package.
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';
import { AppComponent } from './app.component';
@NgModule({
imports: [ BrowserModule, SplitButtonModule ], // Registering EJ2 SplitButton Module.
```



```

declarations: [ AppComponent ],
bootstrap: [ AppComponent ]
})
export class AppModule { }
`

```

Adding Syncfusion SplitButton component

Modify the template in `app.component.ts` file to render the SplitButton component.

```

`typescript
import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  selector: 'app-root',
  template: `<!-- To Render splitbutton. -->
<ejs-splitbutton content="Paste" [items]='items'></ejs-splitbutton>`
})
export class AppComponent {
  public items: ItemModel[] = [
    { text: 'Cut'},
    { text: 'Copy'},
    { text: 'Paste'}
  ];
}
`

```

Adding CSS reference

Add SplitButton component's styles as given below in `style.css`.

```

`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
`

```

Running the application

Run the application in the browser using the following command:

```

`

```

ng serve

The below example shows a basic splitbutton component,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Paste" [items]='items'></ejs-
splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    { text: 'Cut' },
    { text: 'Copy' },
    { text: 'Paste' }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Icons and separator in Angular Split button component

SplitButton Icons

SplitButton can have an icon to provide the visual representation of the action. To place the icon on a SplitButton, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the SplitButton. You can customize the icon's position by using the [iconPosition](#) property

The following example illustrates how to place icon in SplitButton component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'

import { Component } from '@angular/core';
```

```
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Paste" [items]='items' iconCss="e-
sb-icons e-paste"></ejs-splitbutton>
    <ejs-splitbutton content="Paste" [items]='items' iconCss="e-
sb-icons e-paste" iconPosition="Top"></ejs-splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

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 splitBtn using the **iconCss** property.

Vertical Button

Vertical Button in SplitButton can be achieved by adding **e-vertical** class using **cssClass** property.

The following example illustrates how to vertical support in SplitButton component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';

import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
```

```

        SplitButtonModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="e-section-control">
        <!-- To Render splitbutton. -->
        <ejs-splitbutton content="Paste" [items]='items' iconCss="e-
sb-icons e-paste" cssClass="e-vertical" iconPosition="Top"></ejs-
splitbutton></div>`
    })
    export class AppComponent {
        public items: ItemModel[] = [
            {
                text: 'Cut'
            },
            {
                text: 'Copy'
            },
            {
                text: 'Paste'
            }
        ];
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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 SplitButton using the **iconCss** property.

Separator

SplitButton component has Separator support. This can be achieved by setting **separator** as **true**.

The following example illustrates how to enable separator support in SplitButton component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';

import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
    imports: [
        SplitButtonModule
    ],
    standalone: true,

```

```

    selector: 'app-root',
    template: `<div class="e-section-control">
        <!-- To Render splitbutton. -->
        <ejs-splitbutton content="Paste" [items]='items' iconCss="e-
sb-icons e-paste"></ejs-splitbutton></div>`
  })
  export class AppComponent {
    public items: ItemModel[] = [
      {
        text: 'Cut',
        iconCss: 'e-sb-icons e-cut'
      },
      {
        text: 'Copy',
        iconCss: 'e-icons e-copy'
      },
      {
        text: 'Paste',
        iconCss: 'e-sb-icons e-paste'
      },
      {
        separator: true
      },
      {
        text: 'Font',
        iconCss: 'e-sb-icons e-font'
      },
      {
        text: 'Paragraph',
        iconCss: 'e-sb-icons e-paragraph'
      }
    ];
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [SplitButton popup with icons](#)

Popup items in Angular Split button component

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 Cut, Copy and Paste menu items are added using the iconCss property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'

import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ejs-splitbutton content="Paste" iconCss ="e-sb-icons e-
paste"[items]='items'></ejs-splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    {
      text: 'Cut',
      iconCss: 'e-sb-icons e-cut'
    },
    {
      text: 'Copy',
      iconCss: 'e-icons e-copy'
    },
    {
      text: 'Paste',
      iconCss: 'e-sb-icons e-paste'
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Template

Item Templating

Popup items can be customized by 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 it based on the requirement.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { Component } from '@angular/core';
```

```
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton iconCss="e-sb-icons e-paste"
[items]='items' (beforeItemRender)= 'addClass($event)'></ejs-splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    {
      text: 'Cut',
    },
    {
      text: 'Copy',
    },
    {
      text: 'Paste',
    }
  ];
  public addClass(args: MenuEventArgs) {
    let shortCutSpan: HTMLElement = createElement('span');
    let text: string | any = args.item.text;
    args.element.appendChild(shortCutSpan);
    shortCutSpan.setAttribute('class', 'shortcut');
    let clsName: string = (text == 'Copy') ? 'e-icons' : 'e-sb-icons';
    shortCutSpan.classList.add(clsName);
    (text === 'Cut') ? shortCutSpan.classList.add('e-cut') : (text === 'Paste') ? shortCutSpan.classList.add('e-paste') : shortCutSpan.classList.add('e-copy');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Popup Templating

The whole popup can be customized as per the requirement. In the following example, the popup can be customized by handling it in [target](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```

import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <div id='dropdowntarget'>
      <div id= "first">
        <div id='black'></div>
        <div id='red'></div>
        <div id='green'></div>
        <div id='gray'></div>
        <div id='blue'></div>
        <div id='violet'></div>
        <div id='brown'></div>
        <div id='darkgoldenrod'></div>
        <div id='aquamarine'></div>
      </div>
    </div>
    <ejs-splitbutton iconCss ="e-sb-icons e-color"
    target="#dropdowntarget"></ejs-splitbutton></div>`
  })
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Popup items grouping](#)
- [SplitButton popup with separator](#)

Accessibility in Angular Split button component

The Split 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 Split 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 Split button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Split button component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Split button component as **button**, Split button popup as **menu**, and the Split button popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability and type of interactive Split button popup element. |

| **aria-expanded** | Indicates whether the Split button 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 Split 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 Split button component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the opened popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Spacer** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| **Down** | Navigates down or to the next action item. |

| **Alt + Up Arrow** | Closes the popup. |

| **Alt + Down Arrow** | Opens the popup. |

Ensuring accessibility

The Split button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Split button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Split button component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Create right to left splitbutton in Angular Split button component

SplitButton 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 SplitButton component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
```

```
import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Autosum" [items]='items'
iconCss="e-sb e-sigma" enableRtl=true></ejs-splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    {
      text: 'Autosum'
    },
    {
      text: 'Average'
    },
    {
      text: 'Count numbers',
    },
    {
      text: 'Min'
    },
    {
      text: 'Max'
    }
  ];
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Group items in popup in Angular Split button component

Grouped items are possible in SplitButton by templating entire popup with ListView. Check ListView [grouping](#) and create such items. Create ListView with id `listview` and provide element of the ListView as target of SplitButton to render it in popup area.

In this following example, ListView is created and its element is set as [target](#) for SplitButton.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';
import { ListViewModule } from '@syncfusion/ej2-angular-lists';
import { Component } from '@angular/core';
```

```

@Component({
  imports: [
    SplitButtonModule,
    ListViewModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To render ListView. -->
    <ejs-listview id='listview' [dataSource]='listItems'
[fields]='field' sortOrder='Descending'></ejs-listview>
    <!-- To render splitbutton. -->
    <ejs-splitbutton content="Clipboard"
target='#listview'></ejs-splitbutton></div>`
})
export class AppComponent {
  // Datasource for listview.
  public listItems: { [key: string]: Object }[] = [
    {
      'text': 'Cut',
      'category': 'Basic'
    },
    {
      'text': 'Copy',
      'category': 'Basic'
    },
    {
      'text': 'Paste',
      'category': 'Basic'
    },
    {
      'text': 'Paste as Formula',
      'category': 'Advanced'
    },
    {
      'text': 'Paste as Hyperlink',
      'category': 'Advanced'
    }
  ];
  public field: Object = { groupBy: 'category' };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Open a dialog on popup item click in Angular Split button component

This section explains about how to open a dialog on SplitButton popup item click. This can be achieved by handling dialog open in [select](#) event of the SplitButton.

In the following example, Dialog will open while selecting **Update...** item:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { DialogModule } from '@syncfusion/ej2-angular-popups'
import { Component, ViewChild } from '@angular/core';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
@Component({
  imports: [

    SplitButtonModule,
    DialogModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To render Dialog. -->
    <ejs-dialog #dialog [buttons]='alertDlgButtons'
[visible]='false' content='Are you sure want to update?' width='250px'
header='Software Update'>
    </ejs-dialog>
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Help" [items]='items'
(select)='select($event)'></ejs-splitbutton></div>`
})
export class AppComponent {
  @ViewChild('dialog')
  public alertDialog?: DialogComponent;
  public items: ItemModel[] = [
    {
      text: 'Help'
    },
    {
      text: 'About'
    },
    {
      text: 'Update...'
    }
  ];
  public alertDlgButtons: Object[] = [{
    buttonModel: {
      isPrimary: true,
      content: 'Ok',
    },
    click: function () {
      (this as any).hide();
    }
  },
  {
    buttonModel: {
      isPrimary: true,
      content: 'Cancel',
    },
    click: function () {
      (this as any).hide();
    }
  }
  ]
}

```

```

    }
  }];
  public select (args: MenuEventArgs) {
    if (args.item.text === 'Update...') {
      this.alertDialog!.show();
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Set the disabled state in Angular Split button component

SplitButton component can be enabled or disabled by [disabled](#) property.

To disable SplitButton component, set the disabled property as true.

The following example illustrates how to set the disable state in SplitButton component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';

import { Component } from '@angular/core';
import { ItemModel } from '@syncfusion/ej2-angular-splitbuttons';
@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Autosum" [items]='items'
    iconCss="e-sb e-sigma" disabled= true></ejs-splitbutton></div>`
})
export class AppComponent {
  public items: ItemModel[] = [
    {
      text: 'Autosum'
    },
    {
      text: 'Average'
    },
    {
      text: 'Count numbers',
    },
    {
      text: 'Min'
    }
  ]
}

```

```

    },
    {
        text: 'Max'
    }
  ]];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Underline a character in a text in Angular Split button component

To underline a particular character in a text, it 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**:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SplitButtonModule } from '@syncfusion/ej2-angular-splitbuttons';
import { Component } from '@angular/core';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';

@Component({
  imports: [
    SplitButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render splitbutton. -->
    <ejs-splitbutton content="Paste" [items]='items'
    (beforeItemRender)="itemRender($event)"></ejs-splitbutton></div>`
})
export class AppComponent {
  // Initialize action items.
  public items: ItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];

  public itemRender(args: MenuEventArgs) {
    if (args.item.text === 'Copy') {
      // To underline a particular text.
      args.element.innerHTML = '<u>C</u>opy';
    }
  }
}

```

```

    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ej1 api migration in Angular Split button component

This article describes the API migration process of SplitButton 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 splitbutton | **Property:** *text*

 <ej-splitbutton id="splitbutton" text="login"></ej-splitbutton> | **Property:** *content*

 <ejs-splitbutton id="splitbutton" content="Paste"></ejs-splitbutton> |

| Popup content | **Property:** *target*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton> | **Property:** *target*

 <ejs-splitbutton id="splitbutton" target="#target" content="SplitButton"></ejs-splitbutton> |

| Popup items | Not applicable | **Property:** *items*

 <ej-splitbutton id="splitbutton" content="Paste" [items]="items"></ej-splitbutton>
 items: ItemModel[] = [
 {
 text: 'Cut'
 },
 {
 text: 'Copy'
 },
 {
 text: 'Paste'
 }
]; |

| Arrow position | **Property:** *arrowPosition*

 <ej-splitbutton id="splitbutton" text="login" target="#target" arrowPosition="Left"></ej-splitbutton> | Not applicable |

| Button mode | **Property:** *buttonMode*

 <ej-splitbutton id="splitbutton" text="login" target="#target" buttonMode="Dropdown"></ej-splitbutton> | Not applicable |

| Content type | **Property:** *contentType*

 <ej-splitbutton id="splitbutton" text="login" contentType="TextOnly" target="#target"></ej-splitbutton> | Not applicable |

| Icons | **Property:** *prefixIcon*

 <ej-splitbutton id="splitbutton" text="login" contentType="TextAndImage" prefixIcon="e-icon e-handup" target="#target"></ej-splitbutton> | **Property:** *iconCss*

 <ejs-splitbutton id="splitbutton" content="Paste" [items]="items" iconCss="e-icons e-paste"></ejs-splitbutton> |

| Icon position | **Property:** *imagePosition*

 <ej-splitbutton id="splitbutton" text="login" contentType="TextAndImage" prefixIcon="e-icon e-handup" target="#target" imagePosition="ImageTop"></ej-splitbutton> | **Property:** *iconPosition*

 <ejs-splitbutton id="splitbutton" content="Paste" [items]="items" iconCss="e-icons e-paste" iconPosition="Top"></ejs-splitbutton> |

| Secondary icon | **Property:** *suffixIcon*

 <ej-splitbutton id="splitbutton" text="login" contentType="ImageTextImage" prefixIcon="e-icon e-handup" suffixIcon="e-icon e-palette" target="#target"></ej-splitbutton> | Not applicable |

| Adding custom class | **Property:** *cssClass*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" cssClass="custom-class"></ej-splitbutton> | **Property:** *cssClass*

 <ejs-splitbutton id="splitbutton" [items]="items" cssClass="custom-class"></ejs-splitbutton> |

| Disabled state | **Property:** *enabled*

 <ej-splitbutton id="splitbutton" text="login" target="#target" [enabled]="false"></ej-splitbutton> | **Property:** *disabled*

 <ejs-splitbutton id="splitbutton" (items)="items" [disabled]="true"></ejs-splitbutton> |

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

 <ej-splitbutton id="splitbutton" text="login" target="#target" contentType="TextAndImage" prefixIcon="e-icon e-handup" [enableRTL]="true"></ej-splitbutton> | **Property:** *enableRtl*

 <ejs-splitbutton id="splitbutton" [items]="items" content="Paste" [enableRtl]="true" iconCss="e-icons e-paste"></ejs-splitbutton> |

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

 <ej-splitbutton id="splitbutton" text="login" target="#target" height="30px"></ej-splitbutton> | Not applicable |

| Width | **Property:** *width*

 <ej-splitbutton id="splitbutton" text="login" target="#target" width="100px"></ej-splitbutton> | Not applicable |

| HTML attributes | **Property:** *htmlAttributes*

 <ej-splitbutton id="splitbutton" text="login" target="#target" [htmlAttributes]="attributes"></ej-splitbutton> | Not applicable |

| Show rounded corner | **Property:** *showRoundedCorner*

 <ej-splitbutton id="splitbutton" text="login" target="#target" [showRoundedCorner]="true"></ej-splitbutton> | Not applicable |

| Size | **Property:** *size*

 <ej-splitbutton id="splitbutton" size="small" text="Small" target="#target"></ej-splitbutton> | **Property:** *cssClass*

 <ejs-splitbutton id="splitbutton" [items]="items" content="Small" cssClass="e-small"></ejs-splitbutton> |

Methods

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

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

| Destroy method | **Method:** *destroy*

 <ej-splitbutton #split id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.destroy(); | **Method:** *destroy*

 <ejs-splitbutton #split id="splitbutton" [items]="items" content="SplitButton"></ejs-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.destroy(); |

| Disable method | **Method:** *disable*

 <ej-splitbutton #split id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.disable(); | Not applicable |

| Enable method | **Method:** *enable*

 <ej-splitbutton #split id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.enable(); | Not applicable |

| Hide popup | **Method:** *hide*

 <ej-splitbutton #split id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.hide(); | **Method:** *toggle*

 <ej-splitbutton #split id="splitbutton" [items]="items" content="SplitButton"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.toggle(); |

| Show popup | **Method:** *show*

 <ej-splitbutton #split id="splitbutton" text="SplitButton" target="#target"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.show(); | **Method:** *toggle*

 <ej-splitbutton #split id="splitbutton" [items]="items" content="SplitButton"></ej-splitbutton>
 @ViewChild('split')
 public splitButton: SplitButtonComponent;
 this.splitButton.toggle(); |

Events

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

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

| BeforeOpen event | **Event:** *beforeOpen*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (beforeOpen)="beforeOpen(\$event)"></ej-splitbutton>
 beforeOpen(args) {
 / code block /
} | **Event:** *beforeOpen*

 <ej-splitbutton id="splitbutton" [items]="items" content="SplitButton" (beforeOpen)="beforeOpen(\$event)"></ej-splitbutton>
 beforeOpen(args) {
 / code block /
} |

| Click event | **Event:** *click*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (click)="click(\$event)"></ej-splitbutton>
 click(args) {
 / code block /
} | **Event:** *click*

 <ej-splitbutton id="splitbutton" [items]="items" content="SplitButton" (click)="click(\$event)"></ej-splitbutton>
 click(args) {
 / code block /
} |

| Close event | **Event:** *close*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (close)="close(\$event)"></ej-splitbutton>
 close(args) {
 / code block /
} | **Event:** *close*

 <ej-splitbutton id="splitbutton" [items]="items" content="SplitButton" (close)="close(\$event)"></ej-splitbutton>
 close(args) {
 / code block /
} |

| Create event | **Event:** *create*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (create)="create(\$event)"></ej-splitbutton>
 create(args) {
 / code block /
} | **Event:** *created*

 <ej-splitbutton id="splitbutton" [items]="items" content="SplitButton" (created)="created()"></ej-splitbutton>
 created() {
 / code block /
} |

| Destroy event | **Event:** *destroy*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (destroy)="destroy(\$event)"></ej-splitbutton>
 destroy(args) {
 / code block * /
} | Not applicable |

| ItemMouseOut event | **Event:** *itemMouseOut*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (itemMouseOut)="itemMouseOut(\$event)"></ej-splitbutton>
 itemMouseOut(args) {
 / code block */
} | Not applicable |

| ItemMouseOver event | **Event:** *itemMouseOver*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (itemMouseOver)="itemMouseOver(\$event)"></ej-splitbutton>
 itemMouseOver(args) {
 / code block */
} | Not applicable |

| Item select event | **Event:** *itemSelected*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (itemSelected)="itemSelected(\$event)"></ej-splitbutton>
 itemSelected(args) {
 / code block /
} | **Event:** *select*

 <ejs-splitbutton id="splitbutton" [items]="items" content="SplitButton" (select)="select(\$event)"></ejs-splitbutton>
 select(args) {
 / code block /
} |

| Open event | **Event:** *open*

 <ej-splitbutton id="splitbutton" text="SplitButton" target="#target" (open)="open(\$event)"></ej-splitbutton>
 open(args) {
 / code block /
} | **Event:** *open*

 <ejs-splitbutton id="splitbutton" [items]="items" content="SplitButton" (open)="open(\$event)"></ejs-splitbutton>
 open(args) {
 / code block /
} |

| BeforeClose event | Not applicable | **Event:** *beforeClose*

 <ejs-splitbutton id="splitbutton" [items]="items" content="SplitButton" (beforeClose)="beforeClose(\$event)"></ejs-splitbutton>
 beforeClose(args) {
 / code block */
} |

| BeforeItemRender event | Not applicable | **Event:** *beforeItemRender*

 <ejs-splitbutton id="splitbutton" [items]="items" content="SplitButton" (beforeItemRender)="beforeItemRender(\$event)"></ejs-splitbutton>
 beforeItemRender(args) {
 / code block */
} |

Splitter

Getting started with Angular Splitter component

The following section explains the steps required to create the Syncfusion's Angular Splitter component.

The Splitter component will make splittable layouts by placing separator in-between two panes. Based on the position of the separator you can adjust size of the splitter panes in the dynamic manner.

Getting Started with Angular CLI

The following section explains the steps required to create and configure basic angular-cli application.

Prerequisites

To get started with Syncfusion Angular UI Components, make sure that you have compatible versions of Angular and TypeScript.

- Angular : 6+
- TypeScript : 2.6+

Setting up an Angular project

Angular provides an easiest way to setup project using Angular CLI [Angular CLI](#) tool.

Install the CLI application globally in your machine.

```
`javascript
npm install -g @angular/cli
`
```

Create a new application

```
`javascript
ng new syncfusion-angular-app
`
```

Once you have executed the above command you may ask for following options,

- Would you like to add Angular routing?
- Which stylesheet format would you like to use?

By default, it install the CSS style base application. To setup with SCSS, pass `--style=SCSS` argument on create project.

Example code snippet.

```
`javascript
ng new syncfusion-angular-app --style=SCSS
`
```

Navigate to the created project folder.

```
`javascript
cd syncfusion-angular-app
`
```

Installing Syncfusion Splitter package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-layouts](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-layouts@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-layouts:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Splitter module

Once you have successfully installed the layouts package, corresponding component modules are ready to configure in your application from the installed location. Syncfusion Angular package provides two different types of ngModules.

Refer to [Ng-Module](#) to learn about ngModules.

Refer to the following snippet to import the `SplitterModule` in `app.module.ts` from the `@syncfusion/ej2-angular-layouts`.

```
`javascript
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
// Imported syncfusion SplitterModule from layouts package
import { SplitterModule } from '@syncfusion/ej2-angular-layouts';
@NgModule({
  declarations: [
    AppComponent
  ],
```

```

imports: [
  BrowserModule,
  AppRoutingModule,
  // Registering EJ2 Splitter Module
  SplitterModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
`

```

Adding Splitter component

Add the Splitter component snippet in `app.component.ts` as follows.

```

`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `
<div id='container'>
<ejs-splitter #horizontal height='110px' width='100%' >
<e-panes>
<e-pane></e-pane>
<e-pane></e-pane>
</e-panes>
</ejs-splitter>
</div>`
})
export class AppComponent {
  constructor() {
  }
}
`

```

Add following styles in corresponding css file. The below example contains styles in styles.css file,

```
`css
container {
margin: 50px auto;
}
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder. This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-icons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-layouts/styles/material.css';
`
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Load content to the pane

You can load the pane contents either as HTML element or string type using [content](#) property.

For detailed information, refer to the [Pane Content](#) section.

Running the application

Run the `ng serve` command in command window, it will serve your application and you can open the browser window. Output will appear as follows.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
imports: [
FormsModule, SplitterModule
],
standalone: true,
selector: 'app-root',
template: `
<div id='container'>
  <ejs-splitter #horizontal height='250px' width='600px'>
    <e-panes>
      <e-pane></e-pane>
      <e-pane></e-pane>
      <e-pane></e-pane>
    </e-panes>
  </ejs-splitter>
</div>
`
})
export class AppComponent {}
```

```

        </e-panes>
      </ejs-splitter>
    </div>`
  })
  export class AppComponent {
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Construct different layouts using Splitter](#)

Split panes in Angular Splitter component

This section explain about split panes behaviours.

Horizontal layout

By default, Splitter will render in horizontal orientation. Splitter container will be split as panes in horizontal flow direction with vertical separator.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #horizontal height='250px' width='600px'>
        <e-panes>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
                feature-rich control used to display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>

```



```

        <ng-template #content>
            <div class="content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
                facilitates almost all calendar features, thus allowing users to manage
                their time efficiently
            </div>
        </ng-template>
    </e-pane>
    <e-pane size='200px'>
        <ng-template #content>
            <div class="content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
                charting package, is used to add beautiful charts in web and mobile
                applications
            </div>
        </ng-template>
    </e-pane>
</e-panes>
</ejs-splitter>
</div>`
    })
    export class AppComponent {
        constructor() {
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Vertical layout

By setting [orientation](#) property as **Vertical**, Splitter will render in vertical orientation. Splitter container will be split as panes in vertical flow direction with horizontal separator.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
    imports: [
        FormsModule, SplitterModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <div id='container'>

```

```

    <ejs-splitter #vertical orientation='Vertical' height='305px'
width='600px'>
      <e-panes>
        <e-pane size='100px'>
          <ng-template #content>
            <div class="content">
              <h3 class="h3">Grid </h3>
              The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to display data in a tabular format.
            </div>
          </ng-template>
        </e-pane>
        <e-pane size='100px'>
          <ng-template #content>
            <div class="content">
              <h3 class="h3">Schedule </h3>
              The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
            </div>
          </ng-template>
        </e-pane>
        <e-pane size='100px'>
          <ng-template #content>
            <div class="content">
              <h3 class="h3">Chart </h3>
              ASP.NET charts, a well-crafted easy-to-use
charting package, is used to add beautiful charts in web and mobile
applications
            </div>
          </ng-template>
        </e-pane>
      </e-panes>
    </ejs-splitter>
  </div>`
  })
  export class AppComponent {
    constructor() {
  }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple panes

You can render the multiple panes with both **Horizontal** and **Vertical** orientations.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #multiple height='300px' width='600px'>
        <e-panes>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">PARIS </h3>
                Paris, the city of lights and love - this short
guide is full of ideas for how to make the most of the romanticism...
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">CAMEMBERT </h3>
                The village in the Orne département of Normandy
where the famous French cheese is originated from.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">GRENOBLE </h3>
                The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Australia </h3>
                Australia is a country and continent surrounded
by the Indian and Pacific oceans. Its major cities - Sydney, Brisbane,
Melbourne, Perth, Adelaide - are coastal. Its capital, Canberra, is inland.
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
  })
export class AppComponent {
  constructor() {

```

```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Separator

By default, pane separator will be render with **1px** width/height. You can customize the separator size by using [separatorSize](#) property.

For Horizontal orientation, it will be considered as separator width.

For Vertical orientation, it will be considered as separator height.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #separator height='250px' separatorSize='5'
width='600px'>
        <e-panes>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </div>
```

```

        <ng-template #content>
            <div class="content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
                charting package, is used to add beautiful charts in web and mobile
                applications
            </div>
        </ng-template>
    </e-pane>
</e-panes>
</ejs-splitter>
</div>`
    })
    export class AppComponent {
        constructor() {
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. You can use the same `<div>` element for splitter pane and nested splitter.

Also you can render the nested splitter using direct child of the splitter pane. For this, nested splitter should have **100%** width and height to match with the parent pane dimensions.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
import { SplitterComponent } from '@syncfusion/ej2-angular-layouts';
import { Splitter } from '@syncfusion/ej2-layouts';
@Component({
    imports: [
        FormsModule, SplitterModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <div id='container'>
            <ejs-splitter #splitterInstance id="nested-splitter"
            (created)='onCreated()' height='320px' width='580px'>
                <e-panes>
                    <e-pane size='200px'>
                        <ng-template #content>
                            <div class="content">

```

```

        <h3 class="h3">PARIS </h3>
        Paris, the city of lights and love - this short
guide is full of ideas for how to make the most of the romanticism...
    </div>
</ng-template>
</e-pane>
<e-pane size= '200px'>
    <ng-template #content>
        <div class="content">
            <h3 class="h3">CAMEMBERT </h3>
            The village in the Orne département of Normandy
where the famous French cheese is originated from.
        </div>
    </ng-template>
</e-pane>
<e-pane>
    <ng-template #content>
        <div id ='vertical_splitter'
class="vertical_splitter">
            <div>
                <div class="content">
                    <h3 class="h3">GRENOBLE </h3>
                    The capital city of the French Alps and
a major scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.
                </div>
            </div>
            <div>
                <div class="content">
                    <h3 class="h3">Australia </h3>
                    Australia is a country and continent
surrounded by the Indian and Pacific oceans
                </div>
            </div>
        </div>
    </ng-template>
</e-pane>
</e-panes>
</ejs-splitter>
</div>`
}))
export class AppComponent {
    constructor() {
    }
    @ViewChild('splitterInstance') splitterObj?: SplitterComponent;
    public onCreate () {
        let splitterObj1 = new Splitter({
            height: '310px',
            paneSettings: [
                { size: '150px', min: '20%' },
                { size: '100px', min: '20%' }
            ],
            orientation: 'Vertical'
        });
        splitterObj1.appendTo('#vertical_splitter');
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add or remove pane

You can add or remove panes programmatically to the splitter, By using [addPane](#) and [removePane](#) methods.

Add pane

You can add the panes dynamically in the splitter by passing [pane properties](#) along with index to the addPane method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
import { SplitterComponent, PanePropertiesModel } from '@syncfusion/ej2-
angular-layouts';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #add height='200px' width='600px'>
        <e-panes>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">Pane 1</div>
            </ng-template>
          </e-pane>
          <e-pane size='150px'>
            <ng-template #content>
              <div class="content">Pane 2</div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
      <div id='addButton'>
        <button class='e-control e-btn' id='add' (click)='addPane()'>Add
pane</button>
      </div>
    </div>`
})
export class AppComponent {
  @ViewChild('add') splitterObj?: SplitterComponent;
```

```

    constructor() {
    }
    public paneDetails: PanePropertiesModel = {
        size: '190px',
        content: 'New Pane',
        min: '30px',
        max: '250px',
    }
    addPane(): void {
        if ((this.splitterObj as any).allPanels.length >= 1) {
            this.splitterObj!.addPane(this.paneDetails, 1);
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Remove pane

You can remove the split panes dynamically by passing the pane index to [removePane](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
import { SplitterComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #remove height='200px' width='600px'>
        <e-panes>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">Pane 1</div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">Pane 2</div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">Pane 3</div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>
  `
})

```



```

        </ng-template>
      </e-pane>
    </e-panes>
  </ejs-splitter>
  <div id='removeButton'>
    <button class='e-control e-btn' id='remove'
    (click)='removePane()'>Remove pane</button>
  </div>
</div>`
  ))
}
export class AppComponent {
  @ViewChild('remove') splitterObj?: SplitterComponent;
  constructor() {
  }
  removePane(): void {
    if ((this.splitterObj as any).allPanes.length > 1) {
      this.splitterObj?.removePane(1);
    }
  }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Resizable split panes](#)
- [Collapsible panes](#)
- [Define size to a panes](#)
- [Specify content to a panes](#)

Resize in Angular Splitter component

By default, resizing will be enable for split panes. Resizing gripper element will be add to the separator to makes the resize easy.

Horizontal Splitter will allows to resize in horizontal directions.

Vertical Splitter will allows to resize in vertical directions.

While resizing, previous and next panes will be adjust its dimensions automatically.

Min and Max validation

Splitter allows you to set the minimum and maximum sizes for each pane. Resizing will not be occur over the minimum and maximum values.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'

```

```
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #validate height='200px' width='600'>
        <e-panes>
          <e-pane size='200px' min='20%' max='40%'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to
                display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px' min='30%' max='60%'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
charting package, is used to add beautiful charts in web and mobile
applications
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Prevent resizing

You can disable the resizing for the pane by setting `false` to the [resizable](#) property within paneSettings.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #resize height='200px' width='600'>
        <e-panes>
          <e-pane size='200px' min='20%' max='40%'
[resizable]='false'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to
                display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px' min='30%' max='60%'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
charting package, is used to add beautiful charts in web and mobile
applications
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
```

```

    })
    export class AppComponent {
        constructor() {
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: Splitter resizing will be enabled only when the target of the adjacent pane's `resizable` api is also in `true` state.

[Refresh content on resizing](#)

While resizing the panes, you can refresh the pane contents by using either [resizeStart](#), [resizing](#) or [resizeStop](#) events.

[Customize Resize-gripper and Cursor](#)

You can customize the resize gripper icon and cursor in CSS level.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #resizegrip id='resizegrip' height='200px'
width='600'>
        <e-panes>
          <e-pane size='200px' min='20%' max='40%'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">PARIS </h3>
                Paris, the city of lights and love - this short
guide is full of ideas for how to make the most of the romanticism...
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px' min='30%' max='60%'>
            <ng-template #content>
              <div class="content">
                <h3 class="h3">CAMEMBERT </h3>

```

```

        The village in the Orne département of Normandy
where the famous French cheese is originated from.
    </div>
  </ng-template>
</e-pane>
<e-pane size='200px'>
  <ng-template #content>
    <div class="content">
      <h3 class="h3">GRENOBLE </h3>
      The capital city of the French Alps and a major
      scientific center surrounded by many ski resorts, host of the Winter
      Olympics in 1968.
    </div>
  </ng-template>
</e-pane>
</e-panes>
</ejs-splitter>
</div>`
  })
  export class AppComponent {
    constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Collapsible panes](#)

Pane content in Angular Splitter component

This section explains how to provide plain text content or HTML markup or integrate other Angular UI controls as content of Splitter.

Template

You can render the HTML element directly to the Splitter pane content using ng-template.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,

```

```

selector: 'app-root',
template: `
  <div id='template_container'>
    <ejs-splitter #template height='250px' width='580px' >
      <e-panes>
        <e-pane size='200px'>
          <ng-template #content>
            <div class="template">
              <h3>PARIS </h3>Paris, the city of lights and
love - this short guide is full of ideas for how to make the most of the
romanticism...
            </div>
          </ng-template>
        </e-pane>
        <e-pane size='200px'>
          <ng-template #content>
            <div class="template">
              <h3>CAMEMBERT </h3>The village in the Orne
département of Normandy where the famous French cheese is originated from.
            </div>
          </ng-template>
        </e-pane>
        <e-pane size='200px'>
          <ng-template #content>
            <div class="template">
              <h3>GRENOBLE </h3>The capital city of the French
Alps and a major scientific center surrounded by many ski resorts, host of
the Winter Olympics in 1968.
            </div>
          </ng-template>
        </e-pane>
      </e-panes>
    </ejs-splitter>
  </div>`
})
export class AppComponent {
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Angular UI Components

You can render any Angular UI components along with their native and control events within Splitter as pane content.

You can refer [Accordion within splitter](#) and [Listview within splitter](#) samples.

Plain content

You can add the plain text as a pane contents using either inner HTML or [content](#) API

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #plain height='200px' width='600px'>
        <e-panes>
          <e-pane size='200px' content='<div class="content">Left
pane</div>'>
            </e-pane>
          <e-pane size='200px' content='<div class="content">Middle
pane</div>'>
            </e-pane>
          <e-pane size='200px' content='<div class="content">Right
pane</div>'>
            </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

HTML Markup

Splitter is a layout based container component. You can render the pane contents from existing HTML markup. Converting HTML markup as Splitter pane is easy way to add the panes content dynamically.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
```

```

    ],
    standalone: true,
    selector: 'app-root',
    template: `
      <div id='container'>
        <ejs-splitter #markup height='200px' width='600px'>
          <e-panes>
            <e-pane size='200px' content='<div class="content"><h3
class="h3">PARIS </h3>Paris, the city of lights and love - this short guide
is full of ideas for how to make the most of the romanticism...</div>'>
          </e-pane>
            <e-pane size='200px' content='<div class="content"><h3
class="h3">CAMEMBERT </h3>The village in the Orne département of Normandy
where the famous French cheese is originated from.</div>'>
          </e-pane>
            <e-pane size='200px' content='<div class="content"><h3
class="h3">GRENOBLE </h3>The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.</div>'>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
  })
  export class AppComponent {
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Pane content using selector

You can set HTML element as pane [content](#) using the query selectors such as ID or CSS class selectors.

The following code demonstrates how to fetch an element from the document and load it to pane using its ID.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
import { SplitterComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',

```



```

    template: `
      <div id='container'>
        <!-- render splitter component -->
        <ejs-splitter #horizontal height='200px' separatorSize=4
width='100%'></ejs-splitter>
        <!-- pane contents -->
        <div id="left-pane-content" style="display: none;">
        <div>Left pane<div id='panetext'>size: 25%</div>
        <div id='panetext'>min: 60px</div>
        </div>
        </div>
        <div id="middle-pane-content" style="display: none;">
        <span>Middle pane<div id="panetext">size: 50%</div>
        <div id="panetext">min: 60px</div>
        </span>
        </div>
        <div id="last-pane-content" style="display: none;">
        <span>Right pane<div id="panetext">size: 25%</div>
        <div id="panetext">min: 60px</div>
        </span>
        </div>
      </div>`
  })
  export class AppComponent {
    constructor() {
    }
    @ViewChild('horizontal') splitterObj?: SplitterComponent;
    ngAfterViewInit() {
      this.splitterObj!.paneSettings = [
        { size: '25%', min: '60px', content: '#left-pane-content' },
        { size: '50%', min: '60px', content: '#middle-pane-content' },
        { size: '25%', min: '60px', content: '#last-pane-content' } ]
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Expand collapse in Angular Splitter component

Collapsible panes

The Splitter panes can be configured with built-in expand and collapse functionalities. By default, the collapsible behavior is disabled. Enable the [collapsible](#) behavior in the paneSettings property to show or hide the expand or collapse icons in the panes. You can dynamically expand and collapse the panes by clicking on the corresponding icons.

The following code shows how to enable collapsible behavior.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='template_container'>
      <ejs-splitter #expand height='250px' width='580px' >
        <e-panes>
          <e-pane size='200px' [collapsible]='true'>
            <ng-template #content>
              <div class="template">
                <h3>PARIS </h3>Paris, the city of lights and
love - this short guide is full of ideas for how to make the most of the
romanticism...
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px' [collapsible]='true'>
            <ng-template #content>
              <div class="template">
                <h3>CAMEMBERT </h3>The village in the Orne
département of Normandy where the famous French cheese is originated from.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px' [collapsible]='true'>
            <ng-template #content>
              <div class="template">
                <h3>GRENOBLE </h3>The capital city of the French
Alps and a major scientific center surrounded by many ski resorts, host of
the Winter Olympics in 1968.
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
  })
export class AppComponent {
  constructor() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Programmatically control the expand and collapse action

The Splitter provides public method to control the expand and collapse behavior programmatically using the [expand](#) and [collapse](#) methods. Refer to the following code example.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
import { SplitterComponent } from '@syncfusion/ej2-angular-layouts';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #splitterInstance height='200px' width='600px'>
        <e-panes>
          <e-pane size='200px' [collapsible]='true' content='<div
class="content" >Left pane</div>'>
        </e-pane>
          <e-pane size='200px' [collapsible]='true' content='<div
class="content">Middle pane</div>'>
        </e-pane>
          <e-pane size='200px' [collapsible]='true' content='<div
class="content">Right pane</div>'>
        </e-pane>
        </e-panes>
      </ejs-splitter>
      <button ejs-button id='expand'
(click)="expandClick()">Expand</button>
      <button ejs-button id='collapse'
(click)="collapseClick()">Collapse</button>
    </div>`
})
export class AppComponent {
  constructor() {
  }
  @ViewChild('splitterInstance') splitterObj?: SplitterComponent;
  public expandClick: any = () => {
    this.splitterObj?.collapse(0);
  }
  public collapseClick: any = () => {
    this.splitterObj?.expand(0);
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Specify initial state to panes

You can render specific panes with collapsed state on page load. Specify a Boolean value to the [collapsed](#) property to control this behavior. The following code explains how to collapse panes on rendering (the second panes renders with collapsed state).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #collapsed height='200px' width='600px'>
        <e-panes>
          <e-pane size='200px' [collapsible]='true' content='<div
class="contents"><h3 class="h3">PARIS </h3>Paris, the city of lights and
love - this short guide is full of ideas for how to make the most of the
romanticism...</div>'>
            </e-pane>
            <e-pane [collapsible]='true' [collapsed]='true' size='200px'
content='<div class="contents"><h3 class="h3">CAMEMBERT </h3>The village in
the Orne département of Normandy where the famous French cheese is
originated from.</div>'>
            </e-pane>
            <e-pane [collapsible]='true' size='200px' content='<div
class="contents"><h3 class="h3">GRENOBLE </h3>The capital city of the French
Alps and a major scientific center surrounded by many ski resorts, host of
the Winter Olympics in 1968.</div>'>
            </e-pane>
          </e-panes>
        </ejs-splitter>
      </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can also explore our [Angular Splitter Expand and Collapse example](#) that shows how to render the expand and collapse feature in Angular Splitter.

See Also

- [Resizable split panes](#)

Pane sizing in Angular Splitter component

Splitter allows you to provide pane sizes either in pixel or percentage formats.

Pane size in pixel

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #plain height='200px' width='600'>
        <e-panes>
          <e-pane size='200px' content='<div class="content">Left
pane</div>'>
          </e-pane>
          <e-pane size='200px' content='<div class="content">Middle
pane</div>'>
          </e-pane>
          <e-pane size='200px' content='<div class="content">Right
pane</div>'>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Pane size in percentage

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #plain height='200px' width='600'>
        <e-panes>
          <e-pane size='30%' content='<div class="content">Left
pane</div>'>
          </e-pane>
          <e-pane size='40%' content='<div class="content">Middle
pane</div>'>
          </e-pane>
          <e-pane size='30%' content='<div class="content">Right
pane</div>'>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Auto-size panes

The splitter panes are automatically adjusted within its layout on resizing, while the size of panes are not specified. Because the panes are designed based on flex layout by default. When add/remove or show/hide the panes, the panes are auto aligned within its container.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
```

```
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #template height='200px'width='600px' >
        <e-panes>
          <e-pane>
            <ng-template #content>
              <div class="auto-size-content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane>
            <ng-template #content>
              <div class="auto-size-content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
              </div>
            </ng-template>
          </e-pane>
          <e-pane>
            <ng-template #content>
              <div class="auto-size-content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
charting package, is used to add beautiful charts in web and mobile
applications
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
  })
  export class AppComponent {
    constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Fixed pane

You can render the split panes with fixed size for both horizontal and vertical mode. Even you provide fixed sizes to all panes, the splitter will consider last pane as flexible pane. Because, the splitter needs at least one pane as flexible pane always to adjust its remaining layout space.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #resize height='200px' width='600'>
        <e-panes>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="fixed-pane-content">
                <h3 class="h3">Grid </h3>
                The ASP.NET DataGrid control, or DataTable is a
feature-rich control used to display data in a tabular format.
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="fixed-pane-content">
                <h3 class="h3">Schedule </h3>
                The ASP.NET Scheduler, a.k.a. event calendar,
facilitates almost all calendar features, thus allowing users to manage
their time efficiently
              </div>
            </ng-template>
          </e-pane>
          <e-pane size='200px'>
            <ng-template #content>
              <div class="fixed-pane-content">
                <h3 class="h3">Chart </h3>
                ASP.NET charts, a well-crafted easy-to-use
charting package, is used to add beautiful charts in web and mobile
applications
              </div>
            </ng-template>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
```



```
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Different layouts in Angular Splitter component

By using splitter control, you can create the different layouts with multiple and nested panes.

Code editor style layout

Step 1: Create the element with two child to render the outer splitter and create the inner splitter from first pane of vertical splitter.

```
`html
```

```
<ejs-splitter #splitterInstance id="outerSplitter" (created)=onCreated() orientation='Vertical'
height='400px' width='100%'>
```

```
<e-panes>
```

```
<e-pane size='53%' min='30%'></e-pane>
```

```
<e-pane>
```

```
<ng-template #content>
```

```
<div class="content">
```

```
<h3 class="h3">Preview of sample</h3>
```

```
<div class="splitter-image">
```

```

```

```
</div>
```

```
</div>
```

```
</ng-template>
```

```
</e-pane>
```

```
</e-panes>
```

```
</ejs-splitter>
```

```
,
```

```
`javascript
```

```
public onCreated () {
```

```
document.getElementById('outerSplitter').querySelector('.e-pane-vertical').setAttribute('id',
'Innersplitter');
```

```

let splitterObj1 = new Splitter({
height: '220px',
paneSettings: [
{ size: '29%', min: '23%', content: this.pane1Content },
{ size: '20%', min: '15%', content: this.pane2Content },
{ size: '35%', min: '35%', content: this.pane3Content }
],
width: '100%'
});
splitterObj1.appendTo('#Innersplitter');
}
`css
.code-editor #code-text {
margin-left: 5px;
}
.code-editor .code-preview {
margin-top: 15px;
font-size: 12px;
}
.code-editor#target {
margin: 20px auto;
max-width: 600px;
}
.code-editor.control-section {
min-height: 370px;
margin-bottom: 15px;
margin-top: 10px;
}
.code-editor .h3 {
font-size: 14px;
margin: 4px;
}

```

```
.code-editor .content {
padding: 12px;
}
.code-editor .splitter-image {
margin: 0 auto;
display: flex;
height: 115px;
margin-top: 10px;
}
,
```

Once the above configurations has been completed, you will get the output like [this](#)

Outlook style layout

Step 1: Create the element with three panes and place the elements within the pane to render **treeview**, **listview** and **RTE**.

```
`html
<ejs-splitter id="splitter1" #splitter1 height='493px' width='100%'>
<e-panes>
<e-pane size='28%' min='27%'>
<ng-template #content>
<div class="content">
<ejs-treeview id="template" [fields]="field">
<!-- Template to render tree node -->
<ng-template #nodeTemplate="" let-data="">
<div>
<div class="treeviewdiv">
<div style="float:left">
<span class="treeName">{{data.name}}</span>
</div>
<div style="margin-right: 5px; float:right">
<span class="treeCount e-badge e-badge-primary" *ngIf="data.count">{{
data.count }}</span>
</div>
</div>
</div>
</div>
```

```

</ng-template>
</ejs-treeview>
</div>
</ng-template>
</e-pane>
<e-pane size='33%' min='23%'>
  <ng-template #content>
    <div class="content">
      <ejs-listview [dataSource]='dataSource' cssClass='e-list-template'>
        <ng-template #template let-dataSource="">
          <div class="settings e-list-wrapper e-list-multi-line e-list-avatar">
            <span class="e-list-item-header">{{dataSource.Name}}</span>
            <span class="e-list-content">
              <div>
                <div class="status">
                  {{dataSource.content}}</div>
                <div id="list-message">
                  {{dataSource.content2}}</div>
                </div>
              </span>
            </div>
          </ng-template>
        </ejs-listview>
      <div id="groupedList" tabindex="1"></div>
    </div>
  </ng-template>
</e-pane>
<e-pane size='37%' min='30%'>
  <ng-template #content>
    <div class="content">
      <div style="width: 100%; padding: 15px;">

```

To...	
Cc...	

Subject	
---------	--

```

</div>
<div class="forum">
<div id="createpostholder">
<ejs-richtexteditor id='blogRTE' #blogRTE height= '262px'></ejs-richtexteditor>
<div id='buttonSection'>
<button ejs-button [isPrimary]="true" id="send">Send</button>
<button ejs-button id="discard">Discard</button>
</div>
</div>
</div>
</div>
</div>
</ng-template>
</e-pane>
</e-panes>
</ejs-splitter>
<!-- Template to render tree node -->
<script id="treeTemplate" type="text/x-template">
<div>
<div class="treeviewdiv">
<div style="float:left">
<span class="treeName">${name}</span>
</div>
${if(count)}
<div style="margin-right: 5px; float:right">
<span class="treeCount e-badge e-badge-primary">${count}</span>
</div>
${/if}
</div>
</div>
</script>
`

```

Splitter

groupedList.e-listview .e-list-group-item {

`css

.outlook-style #discard {

margin-left: 7px;

}

.outlook-style table {

width: 100%;

}

.outlook-style#target {

margin: 20px auto;

max-width: 820px;

}

.outlook-style td {

padding: 2px;

}

.control-section {

min-height: 370px;

}

.e-treeview .e-list-text {

width: 100%;

}

groupedList.e-listview .e-list-group-item {

height: 0;

}

splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {

padding: 15px;

}

buttonSection {

padding: 7px;

}

.outlook-style #createpostholder {

padding-left: 3px;

padding-right: 4px;

}

,

Once the above configurations has been completed, you will get the output like [this](#).

See Also

- [Multiple panes in Splitter](#)

Style in Angular Splitter component

The following content provides the exact CSS structure that can be used to modify the component's appearance based on the user preference.

Customizing the split bar

Use the following CSS to customize the split bar properties.

Horizontal split bar

`CSS

/ default split bar color /

```
.e-splitter .e-split-bar.e-split-bar-horizontal{
```

```
background: blue;
```

```
}
```

/ split bar color in hover and active state /

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover,
```

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active {
```

```
background: green;
```

```
}
```

,

Vertical split bar

`CSS

/ default split bar color /

```
.e-splitter .e-split-bar.e-split-bar-vertical {
```

```
background: blue;
```

```
}
```

/ split bar color in hover and active state /

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover,
```

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active {
```

```
background: green;
```

```
}
```

,

Customizing the split bar resize handle

Use the following CSS to customize the split bar resize handle.

Horizontal split bar resize handle``CSS`*/ default split bar resize handle color /*

```
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
```

/ default split bar resize handle color in hover and active state /

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-resize-handler {
color: green;
}
```

`,`

Vertical split bar resize handle``CSS`*/ default split bar resize handle color /*

```
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
```

/ default split bar resize handle color in hover and active state /

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-resize-handler {
color: green;
}
```

`,`

Customizing the split bar arrows

Use the following CSS to customize the split bar arrows.

Horizontal split bar resize arrows``CSS`*/ split bar arrows /*

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::before, .e-
```



```

splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::after, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::after {
background-color: green;
}

```

/ split bar arrows - circular border /

```

.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left, .e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right {
border-color: rgba(33, 227, 22, 0.5);
}

```

,

Vertical split bar resize arrows

`CSS

/ split bar arrows /

```

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::after {

```

```

background-color: green;

```

```

}

```

/ split bar arrows - circular border /

```

.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down {

```

```

border-color: rgba(33, 227, 22, 0.5);

```

```

}

```

,

To hide the resize handle in Splitter

Use the following CSS to hide the resize handler in the split bar

Hide the horizontal split bar resize arrow

`CSS

```

.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {

```

```

display: none;

```

```

}

```

,

Hide the vertical split bar resize arrow

```
`CSS

.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
display: none;
}
`
```

Globalization in Angular Splitter component

RTL

Specifies the direction of the Splitter component using the enableRtl property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

The following code shows how to enable RTL behavior.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { SplitterModule } from '@syncfusion/ej2-angular-layouts'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule, SplitterModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id='container'>
      <ejs-splitter #plain height='200px' width='600' enableRtl='true'>
        <e-panes>
          <e-pane size='200px' content='<div class="content">Left
pane</div>'>
          </e-pane>
          <e-pane size='200px' content='<div class="content">Middle
pane</div>'>
          </e-pane>
          <e-pane size='200px' content='<div class="content">Right
pane</div>'>
          </e-pane>
        </e-panes>
      </ejs-splitter>
    </div>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Construct different layouts using Splitter](#)

Accessibility in Angular Splitter component

The Splitter 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 Splitter 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

You can use the following key shortcuts to access the splitter without interruptions:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Tab | Helps in focusing the splitter on the page and switching between the consecutive splitter bars. |

| Shift + Tab | Helps in focusing the previous splitter bar element on the splitter. |

| Right arrow | Helps in moving the active horizontal orientated splitter bar to its Right side. |

| Left arrow | Helps in moving the active horizontal orientated splitter bar to its Left side. |

| Up arrow | Helps in moving the active vertical orientated splitter bar to its Up side. |

| Down arrow | Helps in moving the active vertical orientated splitter bar to its Down side. |

| Enter | Helps to toggle between expand and collapse actions of the splitter bar when it is active. |

Ensuring accessibility

The Splitter component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Splitter component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Splitter component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Ej1 api migration in Angular Splitter component

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

Common

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

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

| Adding custom class | **Property:** `cssClass` `

` `<ej-splitter id="splitter" cssClass="customClass"></ej-splitter>` | **Property:** `cssClass` `
<ejs-splitter id="splitter" cssClass="customClass"></ejs-splitter>` `
` |

| Adjusting Height | **Property:** `height` `

` `<ej-splitter id="splitter" height="100%"></ej-splitter>` | **Property:** `height` `
` `<ejs-splitter id="splitter" height="100%"></ejs-splitter>` |

| Adjusting Width | **Property:** `width` `

 <ej-splitter id="splitter" width="600"></ej-splitter>` | **Property:** `width` `

 <ejs-splitter id="splitter" width="100%"></ejs-splitter>` |

| Orientation | **Property:** `orientation` `

 <ej-splitter id="splitter" [orientation]="orientation"></ej-splitter>
` **TS:** `
 export class AppComponent {
 orientation: any;
 constructor() {
 this.orientation =
ej.Orientation.Horizontal;
}
}` | **Property:** `orientation` `

 <ejs-splitter id="splitter" orientation="Horizontal"></ejs-splitter>` |

| Separator Size | Not Available | **Property:** `separatorSize` `

 <ejs-splitter id="splitter" separatorSize=4></ejs-splitter>` |

| Adding HTML attributes | **Property:** `htmlAttributes` `

 <ej-splitter id="splitter" [htmlAttributes]="htmlAttributes"></ej-splitter>
` **TS:** `
 export class AppComponent {
 htmlAttributes: any;
 constructor() {
this.htmlAttributes = {
 class: "my-class",
style: "border: 1px solid red"}
 }}` | Not Available |

| Customize expand/collapse icons | **Property:** `
 expanderTemplate

 <ej-splitter id="splitter" expanderTemplate=''></ej-splitter>` | Not Available |

| Make control flexible for mobile view | **Property:** `isResponsive` `

 <ej-splitter id="splitter" isResponsive="true"></ej-splitter>` | By default, Splitter works with mobile mode. |

| Refresh the Splitter | **Method:** `refresh()` `

 <ej-splitter id="splitter" #splitter></ej-splitter>
` **TS:** `
 @ViewChild('splitter') public
splitterObj: SplitterComponent;
SplitterObj.refresh();
` | **Method:** `refresh()` `

 <ejs-splitter id="splitter" #splitter></ejs-splitter>
` **TS:** `
@ViewChild('splitter') public
splitterObj: SplitterComponent;
SplitterObj.refresh();
` |

| Destroy the Control | **Method:** `destroy()` `

 <ej-splitter id="splitter" #splitter></ej-splitter>
` **TS:** `
@ViewChild('splitter') public
splitterObj: SplitterComponent;
SplitterObj.destroy();
` | **Method:** `destroy()` `
 <ejs-splitter id="splitter" #splitter></ejs-splitter>
` **TS:** `
@ViewChild('splitter') public
splitterObj: SplitterComponent;
SplitterObj.destroy();
` | **Method:** `destroy()` `
` |

| Event triggers after the Splitter created successfully | **Event:** `create` `

 <ej-splitter id="splitter" #splitter (create)='onCreate($event)'></ej-splitter>
` **TS:** `
onCreate(event){
}
` | **Event:** `created` `

 <ejs-splitter id="splitter" #splitter (create)='onCreate($event)'></ejs-splitter>
` **TS:** `
onCreate(event){
}
` |

| Event triggers when Splitter has been destroyed | **Event:** `destroy` `

 <ej-splitter id="splitter" #splitter (destroy)='onDestroy($event)'></ej-splitter>
` **TS:** `
onDestroy(event){
}
` | Not Available |

Accessibility and Localization

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

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

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

 <ej-splitter id="splitter" #splitter allowKeyboardNavigation='true'></ej-splitter> | No separate property for enable/disable keyboard navigation. Its enabled by default. |

| Right to Left | **Property:** *enableRTL*

 <ej-splitter id="splitter" #splitter enableRTL='false'></ej-splitter> | **Property:** *enableRtl*

 <ej-splitter id="splitter" #splitter [enableRtl]='false'></ej-splitter> |

Control State

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

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

| Enable/Disable the control | Not Available | **Property:** *enabled*

 <ej-splitter id="splitter" #splitter [enabled]='true'></ej-splitter> |

State Maintenance

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

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

| Save the model value in local storage or cookies | Not Available | **Property:** *enablePersistence*

 <ej-splitter id="splitter" #splitter [enablePersistence]='true'></ej-splitter> |

Pane Properties

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

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

| Default | **Property:** *properties*

 <ej-splitter id="splitter" #splitter [properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper: any;
constructor() {
this.proper = [];
}
} | **Property:** *paneSettings*
 <ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-splitter>**TS**
public panes: object[] = [];
 |

| Pane Content | Not Available | **Property:** *content*
 <ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-splitter>
TS
public panes: object[] = [{
content: 'First Pane Content'}];
 |

| Change the size of the pane | **Property:** *paneSize*
 <ej-splitter id="splitter" #splitter [properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper: any;
constructor() {
this.proper = [{paneSize: "30px"}];
}
} | **Property:** *size*
 <ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-splitter>
TS
public panes: object[] = [{ size: '25%'}];
 |

| Minimum pane size | **Property:** *minSize*
<ej-splitter id="splitter" #splitter [properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper: any;
constructor() {
this.proper = [{minSize: 30}];
}
} | **Property:** *min*

 <ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-splitter>
TS
public panes: object[] = [{
min: '60px'}];
 |

| Maximum pane size | **Property:** *maxSize*

 <ej-splitter id="splitter" #splitter [properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper:

any;
constructor() {
this.proper = [{maxSize: 30}];
}
| **Property:** *max*

<ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-splitter>
TS
public
panes: object[] = [{max: '60px'}];
|

| Enable/Disable the Pane Resizable behavior | **Property:** *resizable*

<ej-splitter
id="splitter" #splitter [properties]="proper"></ej-splitter>
TS
export class
AppComponent{
proper: any;
constructor() {
this.proper = [{resizable:
false}];
}
| **Property:** *resizable*

<ej-splitter id="splitter" #splitter
[paneSettings]="panes"></ej-splitter>
TS
public panes: object[] = [{resizable:
false}];
|

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

<ej-splitter id="splitter" #splitter
[properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper:
any;
constructor() {
this.proper = [{collapsible: true}];
}
| **Property:** *collapsible*

<ej-splitter id="splitter" #splitter [paneSettings]="panes"></ej-
splitter>
TS
public panes: object[] = [{collapsible: true}];
|

| Expandable | **Property:** *expandable*

<ej-splitter id="splitter" #splitter
[properties]="proper"></ej-splitter>
TS
export class AppComponent{
proper:
any;
constructor() {
this.proper = [{expandable: true}];
}
| Not Available |

| Collapsed | Not Available | **Property:** *collapsed*

<ej-splitter id="splitter" #splitter
[paneSettings]="panes"></ej-splitter>
TS
public panes: object[] = [{collapsed:
true}];
|

| Add Pane | **Method:** *addItem()*

<ej-splitter id="splitter" #splitter></ej-
splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.addItem("New Pane 0", {paneSize: 20, minSize: 20,
maxSize: 100, 0});
| **Method:** *addPane()*

<ej-splitter id="splitter"
#splitter></ej-splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.addPane({size: "25%", content: "Pane"}, 0);
|

| Remove Pane | **Method:** *removeItem()*

<ej-splitter id="splitter" #splitter></ej-
splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.removeItem(0);
| **Method:** *removePane()*

<ej-
splitter id="splitter" #splitter></ej-splitter>
TS
@ViewChild('splitter')
public splitterObj: SplitterComponent;
splitterObj.removePane(0);
|

| Collapse Pane | **Method:** *collapse()*

<ej-splitter id="splitter" #splitter></ej-
splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.collapse(0);
| **Method:** *collapse()*

<ej-splitter
id="splitter" #splitter></ej-splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.collapse(0);
|

| Expand Pane | **Method:** *expand()*

<ej-splitter id="splitter" #splitter></ej-splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.expand(0);
| **Method:** *expand()*

<ej-splitter
id="splitter" #splitter></ej-splitter>
TS
@ViewChild('splitter') public splitterObj:
SplitterComponent;
splitterObj.expand(0);
|

| Event triggers when before panes get expanded/collapsed | **Event:** *beforeExpandCollapse* `

<ej-splitter id="splitter" #splitter (beforeExpandCollapse)= 'beforeExpandCollapse($event)'></ej-splitter>
TS
beforeExpandCollapse(event){ }
` | **Event:** *beforeExpand* `

<ej-splitter id="splitter" #splitter (beforeExpand)= 'beforeExpand($event)'></ej-splitter>
TS
beforeExpand(event){ }
` | **Event:** *beforeCollapse* `
<ej-splitter id="splitter" #splitter (beforeCollapse)= 'beforeCollapse($event)'></ej-splitter>
TS
beforeCollapse(event){ }
` |

| Event triggers when after panes get expanded/collapsed | **Event:** *expandCollapse* `

<ej-splitter id="splitter" #splitter (expandCollapse)= 'expandCollapse($event)'></ej-splitter>
TS
expandCollapse (event){ }
` | **Event:** *expand* `

<ej-splitter id="splitter" #splitter (expand)= 'expand($event)'></ej-splitter>
TS
expand(event){ }
` | **Event:** *collapse* `
<ej-splitter id="splitter" #splitter (collapse)= 'collapse ($event)'></ej-splitter>
TS
collapse (event){ }
` |

| Event triggers when Resizing the pane | **Event:** *resize* `

<ej-splitter id="splitter" #splitter (resize)= 'resize($event)'></ej-splitter>
TS
resize(event){ }
` | **Event:** *resizing* `

<ej-splitter id="splitter" #splitter (resizing)= 'resizing ($event)'></ej-splitter>
TS
resizing (event){ }
` |

| Event triggers when pane is started to resize | Not Available | **Event:** *resizeStart* `

<ej-splitter id="splitter" #splitter (resizeStart)= 'resizeStart ($event)'></ej-splitter>
TS
resizeStart (event){ }
` |

| Event triggers when pane is stopped to resize | Not Available | **Event:** *resizeStop* `

<ej-splitter id="splitter" #splitter (resizeStop)= 'resizeStop($event)'></ej-splitter>
TS
resizeStop (event){ }
` |

| Event triggers when click template icon | **Event:** *clickOnExpander* `

<ej-splitter id="splitter" #splitter (clickOnExpander)= 'clickOnExpander($event)'></ej-splitter>
TS
clickOnExpander(event){ }
` | Not Available |

Animation

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

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

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

<ej-splitter id="splitter" #splitter enableAnimation="true"></ej-splitter>
` | Not Available |

| AnimationSpeed | **Property:** *animationSpeed* `

<ej-splitter id="splitter" #splitter animationSpeed =150></ej-splitter>
` | Not Available |

Spreadsheet

Overview of the Angular Spreadsheet component

The Angular Spreadsheet is an user interactive component to organize and analyze data in tabular format with configuration options for customization. It will load data by importing an Excel/CSV file or from local and remote data sources such as JSON, RESTful services, OData services, and more. The populated data can be exported as Excel with xlsx, xls, CSV and PDF formats.

Key features

- [Data sources](#): Bind the Spreadsheet component with an array of objects or data from a web service using `DataManager`.
- [Virtualization](#): Provides the option to load large amount of data without performance degradation.
- [Selection](#): Provides the option to select a cell or range of cells.
- [Editing](#): Provides the option to dynamically edit a cell.
- [Formulas](#): Provides built-in calculation library with pre-defined formulas and named range support.
- [Clipboard](#): Provides the option to perform clipboard operations.
- [Cell formatting](#): Provides the option to customize the appearance of cells.
- [Number formatting](#): Provides the option to format the cell value.
- [Open](#): Provides the option to open Excel and CSV files in Spreadsheet.
- [Save](#): Provides the option to save Spreadsheet data as Excel, CSV, and PDF documents.
- [Sorting](#): Helps you to arrange the data to particular order in a selected range of cells.
- [Filtering](#): Helps you to view specific rows in the Spreadsheet by hiding the other rows.
- [Undo Redo](#): Provides the option to perform undo redo operations in Spreadsheet.
- [Collaborative editing](#): Provides the option for real time changes across multiple users in the Spreadsheet.
- [Hyperlink](#): Provides the option to navigate to web link or cell reference within the sheet or to other sheet in Spreadsheet.
- [Resize](#): Allows you to change the row height and column width. Auto fit the rows and columns based on its content.
- [Wrap text](#): Provides the option to display the large content as multiple lines in a single cell.
- [Data validation](#): Provides the option to validate edited values based on data validation rules defined for a cell or range of cells.
- [Find and replace](#): Provides the option to find the data and replace it across all sheets in Spreadsheet.
- [Protect sheet](#): Provides the option to restrict user actions like cell editing, row and column insertion, deletion, and resizing.
- [Borders](#): Provides the option to customize cell gridlines such as color and its style for enhanced UI.
- [Show/hide](#): Provides the option to show/hide rows, columns and sheets.
- [Insert/delete](#): Provides the option to insert/delete rows, columns and sheets.
- [Merge cells](#): Provides the option to combine two or more cells located in the same row or column into a single cell.
- [Conditional formatting](#): Provides the option to format a cell or range of cells based on conditions applied.
- [Autofill](#): Provides the option to fill or copy a series or pattern of values and formats into adjacent cells in any direction.
- [Clear](#): Provides the option to clear the content, formats, and hyperlinks applied to a cell or range of cells in a Spreadsheet.
- [Aggregates](#): Provides the option to check the sum, average, count, and more for the selected cells or range in the sheet.
- [Picture](#): Allows you to view, insert, and modify a picture in a Spreadsheet with customizing options.

- [Chart](#): Transforms your Spreadsheet data to an intuitive overview for better understanding and to make smart business decisions.
- [Freeze panes](#): Allows you to keep the specified rows and columns always visible at the top and left side of the sheet while scrolling through the sheet.
- [Password protection](#): Allows you to protect the workbook with a password.
- [Multi-line editing](#): Allows you to insert a line break between paragraphs of the text within a cell in a Spreadsheet.
- [Calculate range selection](#): Helps you to select a range or multiple ranges when editing a formula in a cell.
- [Right-to-left \(RTL\)](#): Aligns content in the Spreadsheet component from right to left.
- [Templates](#): Templates can be used to create custom user experiences in the Spreadsheet.
- [Globalization](#): Personalize the Spreadsheet component with different languages, as well as culture-specific number, date, and time formatting.
- [AccessibilityLink to the Video](#): Provides with built-in accessibility support which helps to access all the Spreadsheet component features through the keyboard, screen readers, or other assistive technology devices.

Getting started with Angular Spreadsheet component

This section explains the steps to create a simple Spreadsheet control with basic features in an Angular environment.

To get start quickly with Angular Spreadsheet using CLI, you can check on this video:

Dependencies

The following list of dependencies are required to use the Spreadsheet control in your application.

```
`js
|-- @syncfusion/ej2-angular-spreadsheet
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-spreadsheet
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-grids
`,`
```

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli
`,`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion Spreadsheet package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-spreadsheet](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-spreadsheet --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-spreadsheet@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-spreadsheet@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-spreadsheet:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering Spreadsheet Module

Import Spreadsheet module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-spreadsheet` [`src/app/app.module.ts`].

```
`typescript
```

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet';
import { AppComponent } from './app.component';

@NgModule({
  imports: [ BrowserModule, SpreadsheetAllModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import '../node_modules/@syncfusion/ej2-spreadsheet/styles/material.css';
@import '../node_modules/@syncfusion/ej2-grids/styles/material.css';
`
```

Add Spreadsheet control

Modify the template in `[src/app/app.component.ts]` file to render the spreadsheet component. Add the Angular Spreadsheet by using `<ejs-spreadsheet>` selector in template section of the `app.component.ts` file.

```
`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
```

// specifies the template string for the Spreadsheet control

```
template: <ejs-spreadsheet> </ejs-spreadsheet>
```

```
}}
```

```
export class AppComponent { }
```

```
,
```

[Run the application](#)

Use the following command to run the application in the web browser

```
,
```

```
ng serve
```

```
,
```

The following example shows a basic Spreadsheet component

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component } from '@angular/core';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-root',
  template: '<ejs-spreadsheet > </ejs-spreadsheet>'
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) that shows you how present and manipulate data, including editing, formulas, formatting, importing, and exporting.

[See Also](#)

- [Data Binding](#)
- [Open and Save](#)

Data binding in Angular Spreadsheet component

The Spreadsheet uses [DataManager](#), which supports both RESTful JSON data services and local JavaScript object array binding to a range. The `dataSource` property can be assigned either with the instance of [DataManager](#) or JavaScript object array collection.

To bind data to a cell, use `cell data binding` support.

Local data

To bind local data to the Spreadsheet, you can assign a JavaScript object array to the `dataSource` property.

Refer to the following code example for local data binding.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-column [width]=130></e-column><e-column [width]=120></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.data = data;
  }
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

The local data source can also be provided as an instance of the [DataManager](#). By default, [DataManager](#) uses [JsonAdaptor](#) for local data-binding.

Remote data

To bind remote data to the Spreadsheet control, assign service data as an instance of [DataManager](#) to the `dataSource` property. To interact with remote data source, provide the service endpoint `url`.

Refer to the following code example for remote data binding.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet><e-sheets><e-sheet
name='Shipment Details'> <e-ranges> <e-range [showFieldAsHeader]='false'
startCell='A2' [dataSource]='data' [query]='query'> </e-range> </e-
ranges><e-rows><e-row><e-cells><e-cell value='Order ID'></e-cell> <e-cell
value='Customer Name'></e-cell> <e-cell value='Freight'></e-cell> <e-cell
value='Ship Name'></e-cell><e-cell value='Ship City'></e-cell> <e-cell
value='Ship Country'></e-cell></e-cells> </e-row></e-rows><e-columns><e-
column [width]=100></e-column> <e-column [width]=130></e-column> <e-column
[width]=100></e-column> <e-column [width]=220></e-column> <e-column
[width]=150></e-column> <e-column [width]=180></e-column> </e-columns> </e-
sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  public query: Query = new Query().select(['OrderID', 'CustomerID',
'ShipName', 'ShipCity', 'ShipCountry', 'Freight']).take(200);
  public data: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/js/production/api/Orders',
    crossDomain: true
  });
  ngOnInit(): void {
    this.data = this.data;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

By default, `DataManager` uses `ODataAdaptor` for remote data-binding.

Binding with OData services

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { DataManager, ODataAdaptor } from '@syncfusion/ej2-data';
import { SpreadsheetComponent, CellModel, UsedRangeModel, SheetModel } from '@syncfusion/ej2-angular-spreadsheet';
import { getComponent, print } from '@syncfusion/ej2-base';
@Component({
  imports: [
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (created)='created()'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=280></e-column><e-column [width]=180></e-column><e-column [width]=80></e-column><e-column [width]=180></e-column><e-column [width]=180></e-column></e-columns> </e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  @ViewChild('spreadsheet')
  public spreadsheetObj: SpreadsheetComponent | undefined;
  public data?: DataManager;
  ngOnInit(): void {
    this.data = new DataManager({
      url:
        'https://services.syncfusion.com/angular/production/api/Orders',
      adaptor: new ODataAdaptor(),
      crossDomain: true
    });
  }
  created(){
    //Applies cell and number formatting to specified range of the active sheet
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle' },
      'A1:K1');
  };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```



```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Web API

You can use WebApiAdaptor to bind spreadsheet with Web API created using OData endpoint.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataManager, WebApiAdaptor } from '@syncfusion/ej2-data';
import { getComponent, print } from '@syncfusion/ej2-base';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (created)='created()'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=80></e-column><e-column [width]=280></e-column><e-column [width]=180></e-column><e-column [width]=80></e-column><e-column [width]=180></e-column><e-column [width]=180></e-column></e-columns> </e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  @ViewChild('spreadsheet')
  public spreadsheetObj: SpreadsheetComponent | undefined;
  public data?: DataManager;
  ngOnInit(): void {
    this.data = new DataManager({
      url:
        'https://services.syncfusion.com/angular/production/api/Orders',
      adaptor: new WebApiAdaptor(),
      crossDomain: true
    });
  }
  created() {
    //Applies cell and number formatting to specified range of the active sheet
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign: 'center', verticalAlign: 'middle' },
      'A1:K1');
  };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Cell data binding

The Spreadsheet control can bind the data to individual cell in a sheet . To achieve this you can use the **value** property.

Refer to the following code example for cell data binding.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet> <e-sheets> <e-sheet
selectedRange='D13'><e-rows> <e-row><e-cells> <e-cell value='Order ID'></e-
cell> <e-cell value='Customer ID'></e-cell> <e-cell value='Employee ID'></e-
cell> <e-cell value='Ship Name'></e-cell> <e-cell value='Ship City'></e-
cell> <e-cell value='Ship Address'></e-cell> </e-cells> </e-row> <e-row>
<e-cells> <e-cell value='10248'></e-cell> <e-cell value='VINET'></e-cell>
<e-cell value='5'></e-cell> <e-cell value='Vins et alcools Chevalier'></e-
cell> <e-cell value='Reims'></e-cell> <e-cell value='59 rue de
l'Abbaye'></e-cell> </e-cells> </e-row> <e-row> <e-cells> <e-cell
value='10249'></e-cell> <e-cell value='TOMSP'></e-cell> <e-cell
value='3'></e-cell> <e-cell value='Toms Spezialitäten'></e-cell> <e-cell
value='Münster'></e-cell> <e-cell value='Luisenstr. 48'></e-cell> </e-
cells> </e-row> <e-row> <e-cells> <e-cell value='10250'></e-cell> <e-
cell value='HANAR'></e-cell> <e-cell value='2'></e-cell> <e-cell
value='Hanari Carnes'></e-cell> <e-cell value='Rio de Janeiro'></e-cell>
<e-cell value='Rua do Paço, 67'></e-cell> </e-cells> </e-row> <e-row> <e-
cells> <e-cell value='10251'></e-cell> <e-cell value='VICTE'></e-cell> <e-
cell value='3'></e-cell> <e-cell value='Victuailles en stock'></e-cell> <e-
cell value='Lyon'></e-cell> <e-cell value='2, rue du Commerce'></e-cell>
</e-cells> </e-row> <e-row> <e-cells> <e-cell value='10252'></e-cell> <e-
cell value='SUPRD'></e-cell> <e-cell value='4'></e-cell> <e-cell
value='Suprêmes délices'></e-cell> <e-cell value='Charleroi'></e-cell> <e-cell
value='Boulevard Tirou, 255'></e-cell> </e-cells> </e-row> </e-rows> <e-
columns> <e-column [width]=110></e-column> <e-column [width]=115></e-
column> <e-column [width]=110></e-column> <e-column [width]=130></e-column>
<e-column [width]=140></e-column> <e-column [width]=150></e-column> </e-
columns></e-sheet></e-sheets><e-rows></e-rows></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

The cell data binding also supports formula, style, number format, and more.

Dynamic data binding and Datasource change event

You can dynamically change the datasource of the spreadsheet by changing the model which is bound to the `dataSource` property or by changing the `dataSource` property of the `range` object of the `sheet` directly. The `dataSourceChanged` event handler will be triggered when editing, inserting, and deleting a row in the datasource range. This event will be triggered with a parameter named `action` which indicates the `edit`, `add` and `delete` actions for the respective ones.

The following table defines the arguments of the `dataSourceChanged` event.

Property	Type	Description
action	string	Indicates the type of action such as <code>edit</code> , <code>add</code> , and <code>delete</code> performed in the datasource range.
data	object[]	Modified data for <code>edit</code> action; New data for <code>add</code> action; Deleted data for <code>delete</code> action.
rangeIndex	number	Specifies the range index of the datasource.
sheetIndex	number	Specifies the sheet index of the datasource.

For `add` action, the value for all the fields will be `null` in the data. In the case that you do not want the primary key field to be null which needs to be updated in the backend service, you can use `edit` action after updating the primary key field to update in the backend service.

For inserting a row at the end of the datasource range, you should insert a row below at the end of the range to trigger the `dataSourceChanged` event with action `add`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild, ViewEncapsulation } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataSourceChangedEventArgs } from '@syncfusion/ej2-spreadsheet';
import { data, itemData } from './datasource';
@Component({
  imports: [
    SpreadsheetAllModule
```

```

    ],
    standalone: true,
    selector: 'app-container',
    template: `<div>
      <div>
        <button class='e-btn' style="margin-bottom: 10px;"
        (click)='changeDataSource()'>Change Datasource</button>
        <ejs-spreadsheet #spreadsheet
        (dataSourceChanged)='dataSourceChanged($event)' [showRibbon]='false'>
          <e-sheets>
            <e-sheet>
              <e-ranges><e-range [dataSource]='data'></e-range></e-ranges>
              <e-columns><e-column [width]=90></e-column><e-column
[width]=100></e-column><e-column [width]=96></e-column><e-column
[width]=120></e-column><e-column [width]=130></e-column><e-column
[width]=120></e-column></e-columns>
            </e-sheet>
          </e-sheets>
        </ejs-spreadsheet>
      </div>
      <div>
        <h4><b>Event Trace</b></h4>
        <div id="evt">
          <div style="height:173px;overflow: auto;min-width: 250px;">
            <span #EventLog class="EventLog" id="EventLog"
style="word-break: normal;"></span>
          </div>
          <button class='e-btn' (click)='clearTrace()'>Clear</button>
        </div>
      </div>`,
    styles: [
      #EventLog b {
        color: #388e3c;
      }
      #evt {
        border: 1px solid #dcdcdc;
        padding: 10px;
      }
    ],
    encapsulation: ViewEncapsulation.None
  })
}

export class AppComponent {
  public data: object[] = data;
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  @ViewChild('EventLog') EventLogEle: any;
  dataSourceChanged(args: DataSourceChangedEventArgs): void {
    this.appendElement("Data source changed with" + "<b>#{160;" +
args.action + "</b> action<hr>");
  }
  changeDataSource(): void {
    this.data = itemData;
    // You can also change the datasource of the range by changing
    dataSource property of the range by using below line of code.
    // this.spreadsheetObj.sheets[0].ranges[0].dataSource = itemData;
  }
  clearTrace(): void {

```

```

        this.EventLogEle.nativeElement.innerHTML = "";
    }
    appendElement(html: string): void {
        let span: HTMLElement = document.createElement("span");
        span.innerHTML = html;
        let log: HTMLElement = this.EventLogEle.nativeElement;
        log.insertBefore(span, log.firstChild);
    }
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)
- [Collaborative Editing](#)

Open save in Angular Spreadsheet component

In import an excel file, it needs to be read and converted to client side Spreadsheet model. The converted client side Spreadsheet model is sent as JSON which is used to render Spreadsheet. Similarly, when you save the Spreadsheet, the client Spreadsheet model is sent to the server as JSON for processing and saved. Server configuration is used for this process.

Open

The Spreadsheet control opens an Excel document with its data, style, format, and more. To enable this feature, set [allowOpen](#) as `true` and assign service url to the [openUrl](#) property.

User Interface:

In user interface you can open an Excel document by clicking **File > Open** menu item in ribbon.

The following sample shows the **Open** option by using the [openUrl](#) property in the Spreadsheet control. You can also use the [beforeOpen](#) event to trigger before opening an Excel file.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons';
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet';
import { Component } from '@angular/core';

```

```
import { SpreadsheetComponent, BeforeSaveEventArgs, BeforeOpenEventArgs }
  from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule

  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet (beforeOpen)='beforeOpen($event)'
  openUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
  open' allowOpen='true'> </ejs-spreadsheet>"
})
export class AppComponent {
  beforeOpen (args: BeforeOpenEventArgs) {
    // your code snippets here
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Please find the below table for the beforeOpen event arguments.

Parameter	Type	Description
-----------	------	-------------

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

file	FileList or string or File	To get the file stream. FileList - contains length and item index.
------	----------------------------	---

File		specifies the file lastModified and file name.
-------------	--	--

cancel	boolean	To prevent the open operation.
--------	---------	--------------------------------

requestData	object	To provide the Form data.
-------------	--------	---------------------------

* Use **Ctrl + O** keyboard shortcut to open Excel documents.

* The default value of the [allowOpen](#) property is **true**. For demonstration purpose, we have showcased the [allowOpen](#) property in previous code snippet.

Open an external URL excel file while initial load

You can achieve to access the remote excel file by using the [created](#) event. In this event you can fetch the excel file and convert it to a blob. Convert this blob to a file and [open](#) this file by using Spreadsheet component open method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
```

```
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [
    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (created)='created()' "
  openUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
  open' allowOpen='true'> </ejs-spreadsheet>"
})
export class AppComponent {
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  created () {
    fetch("https://cdn.syncfusion.com/scripts/spreadsheet/Sample.xlsx")
    // fetch the remote url
    .then((response) => {
      response.blob().then((fileBlob) => { // convert the excel file
        to blob
        let file = new File([fileBlob], "Sample.xlsx"); //convert the
        blob into file
        this.spreadsheetObj!.open({ file: file }); // open the file into
        Spreadsheet
      })
    })
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

To add custom header during open

You can add your own custom header to the open action in the Spreadsheet. For processing the data, it has to be sent from server to client side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [beforeOpen](#) event, the custom header can be added to the request during open action.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, BeforeOpenEventArgs } from '@syncfusion/ej2-
angular-spreadsheet';
@Component({
  imports: [
```

```

        DropDownButtonModule,
        UploaderModule,
        SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template:
        "<ejs-spreadsheet #spreadsheet
        openUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
        open' allowOpen='true' (beforeOpen)='beforeOpen($event)'> </ejs-
        spreadsheet>",
    })
    export class AppComponent {
        @ViewChild('spreadsheet')
        public spreadsheetObj: SpreadsheetComponent;
        beforeOpen(args: BeforeOpenEventArgs) {
            args.requestData['headers'] = {
                Authorization: 'YOUR TEXT',
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Open excel file into a read-only mode

You can open excel file into a read-only mode by using the [openComplete](#) event. In this event, you must protect all the sheets and lock its used range cells by using [protectSheet](#) and [lockCells](#) methods.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import {
    SheetModel,
    ProtectSettingsModel,
    getRangeAddress,
} from '@syncfusion/ej2-spreadsheet';
@Component({
    imports: [

        DropDownButtonModule,
        SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',

```



```

    template: "<ejs-spreadsheet #spreadsheet
openUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
open' allowOpen='true' (openComplete)='openComplete($event)'> </ejs-
spreadsheet>"
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    public spreadsheetObj!: SpreadsheetComponent;
    openComplete(args: Object) {
      let sheets: SheetModel[] = this.spreadsheetObj.sheets;
      for (let index: number = 0; index < sheets.length; index++) {
        let name: string = this.spreadsheetObj.sheets[index].name!;
        let protectSetting: ProtectSettingsModel = {
          selectCells: true,
          formatCells: false,
        };
        //To protect the sheet using sheet name
        this.spreadsheetObj.protectSheet(name, protectSetting);
        let address: string = getRangeAddress([
          0,
          0,
          sheets[index].usedRange?.rowIndex!,
          sheets[index].usedRange?.colIndex!
        ]);
        //To lock the used range cells
        this.spreadsheetObj.lockCells(name + '!' + address, true);
      }
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

External workbook confirmation dialog

When you open an excel file that contains external workbook references, you will see a confirmation dialog. This dialog allows you to either continue with the file opening or cancel the operation. This confirmation dialog will appear only if you set the `AllowExternalWorkbook` property value to `false` during the open request, as shown below. This prevents the spreadsheet from displaying inconsistent data.

```
`csharp
```

```

public IActionResult Open(IFormCollection openRequest)
{
    OpenRequest open = new OpenRequest();
    open.AllowExternalWorkbook = false;
    open.File = openRequest.Files[0];
}

```

```
return Content(Workbook.Open(open));
}
,
```

This feature is only applicable when importing an Excel file and not when loading JSON data or binding cell data.

![External workbook confirmation dialog](./images/external-reference-dialog-alert%20.png)

Save

The Spreadsheet control saves its data, style, format, and more as Excel file document. To enable this feature, set [allowSave](#) as **true** and assign service url to the [saveUrl](#) property.

User Interface:

In user interface, you can save Spreadsheet data as Excel document by clicking **File > Save As** menu item in ribbon.

The following sample shows the **Save** option by using the [saveUrl](#) property in the Spreadsheet control. You can also use the [beforeSave](#) event to trigger before saving the Spreadsheet as an Excel file.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component } from '@angular/core';
import { SpreadsheetComponent, BeforeSaveEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet (beforeSave)='beforeSave($event)' saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/save' allowSave='true'> </ejs-spreadsheet>"
})
export class AppComponent {
  beforeSave (args: BeforeSaveEventArgs) {
    // your code snippets here
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Please find the below table for the beforeSave event arguments.

Parameter	Type	Description
-----	-----	-----
url	string	Specifies the save url.
fileName	string	Specifies the file name.
saveType	SaveType	Specifies the saveType like Xlsx, Xls, Csv and Pdf.
customParams	object	Passing the custom parameters from client to server while performing save operation.
isFullPost	boolean	It sends the form data from client to server, when set to true. It fetches the data from client to server and returns the data from server to client, when set to false.
needBlobData	boolean	You can get the blob data if set to true.
cancel	boolean	To prevent the save operations.

* Use **Ctrl + S** keyboard shortcut to save the Spreadsheet data as Excel file.

* The default value of [allowSave](#) property is **true**. For demonstration purpose, we have showcased the [allowSave](#) property in previous code snippet.

* Demo purpose only, we have used the online web service url link.

To send and receive custom params from client to server

Passing the custom parameters from client to server by using [beforeSave](#) event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit } from '@angular/core';
import { SpreadsheetComponent, BeforeSaveEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet
(beforeSave)='beforeSave($event)'
saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/save' allowSave='true'> <e-sheets> <e-sheet> <e-ranges> <e-range
[dataSource]='data'></e-range></e-ranges><e-columns><e-column
[width]=90></e-column><e-column [width]=100></e-column><e-column
[width]=96></e-column><e-column [width]=120></e-column><e-column
[width]=130></e-column><e-column [width]=120></e-column></e-columns></e-
sheet></e-sheets></ejs-spreadsheet>"
})
```

```
export class AppComponent implements OnInit {
    public data?: object[];
    ngOnInit(): void {
        this.data = data;
    }
    beforeSave (args: BeforeSaveEventArgs) {
        args.customParams = { customParams: 'you can pass custom params in server side'}; // you can pass the custom params
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Server side code snippets:

```
`csharp
```

```
public IActionResult Save(SaveSettings saveSettings, string customParams)
```

```
{
```

```
    Console.WriteLine(customParams); // you can get the custom params in controller side
```

```
    return Workbook.Save(saveSettings);
```

```
}
```

```
,
```

To add custom header during save

You can add your own custom header to the save action in the Spreadsheet. For processing the data, it has to be sent from client to server side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [fileMenuItemSelect](#) event, the custom header can be added to the request during save action.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { MenuSelectEventArgs, SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { createElement } from '@syncfusion/ej2-base';
import { data } from './datasource';
@Component({
    imports: [
        DropDownButtonModule,
        SpreadsheetAllModule
    ],
```

```

standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet
(fileMenuItemSelect)='onFileItemSelect($event)'
saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
save' allowSave='true'> <e-sheets> <e-sheet> <e-ranges> <e-range
[dataSource]='data'></e-range></e-ranges><e-columns><e-column
[width]=90></e-column><e-column [width]=100></e-column><e-column
[width]=96></e-column><e-column [width]=120></e-column><e-column
[width]=130></e-column><e-column [width]=120></e-column></e-columns></e-
sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public data: object[] | undefined;
  @ViewChild('spreadsheet') public spreadsheetObj!: SpreadsheetComponent;
  ngOnInit(): void {
    this.data = data;
  }
  onFileItemSelect(args: MenuSelectEventArgs) {
    if (args.item.text === 'Microsoft Excel') {
      args.cancel = true;
      this.spreadsheetObj
        .saveAsJson()
        .then((response: any) => {
          let formData: FormData = new FormData();
          formData.append(
            'JSONData',
            JSON.stringify(response.jsonObject.Workbook)
          );
          formData.append('fileName', 'Sample');
          formData.append('saveType', 'Xlsx');
          formData.append(
            'PdfLayoutSettings',
            JSON.stringify({ FitSheetOnOnePage: false })
          );
          fetch(
            'https://services.syncfusion.com/angular/production/api/spreadsheet/save',
            {
              method: 'POST',
              headers: { Authorization: 'YOUR TEXT' },
              body: formData,
            }
          ).then((response) => {
            response.blob().then((data) => {
              let anchor: HTMLAnchorElement = createElement('a', {
                attrs: { download: 'Sample.xlsx' },
              });
              let url = URL.createObjectURL(data);
              (anchor as HTMLAnchorElement).href = url;
              document.body.appendChild(anchor);
              anchor.click();
              URL.revokeObjectURL(url);
              document.body.removeChild(anchor);
            });
          });
        });
    }
  }
}

```

```

    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

To change the PDF orientation

By default, the PDF document is created in **Portrait** orientation. You can change the orientation of the PDF document by using the `args.pdfLayoutSettings.orientation` argument settings in the [beforeSave](#) event.

The possible values are:

- **Portrait** - Used to display content in a vertical layout.
- **Landscape** - Used to display content in a horizontal layout.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit } from '@angular/core';
import { SpreadsheetComponent, BeforeSaveEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
  imports: [
    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (beforeSave)='beforeSave($event)' saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/save' allowSave='true'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-column [width]=130></e-column><e-column [width]=120></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public data?: object[];
  ngOnInit(): void {
    this.data = data;
  }
}

```

```

        beforeSave (args: BeforeSaveEventArgs) {
            args.pdfLayoutSettings!.orientation = 'Landscape'; // You can change the orientation of the PDF document
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Methods

To save the Spreadsheet document as an `xlsx`, `xls`, `csv`, or `pdf` file, by using [save](#) method should be called with the `url`, `fileName` and `saveType` as parameters. The following code example shows to save the spreadsheet file as an `xlsx`, `xls`, `csv`, or `pdf` in the button click event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { data } from './datasource';
@Component({
    imports: [

        DropDownButtonModule,
        SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template: "<button ej2-dropdownbutton [items]='items' content='Save' (select)='itemSelect($event)'></button> <ejs-spreadsheet #spreadsheet > <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-column [width]=130></e-column><e-column [width]=120></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    ngOnInit(): void {
        this.data = data;
    }
    public items: ItemModel[] = [
        {
            text: "Save As xlsx"
        },
    ],

```

```

        {
            text: "Save As xls"
        },
        {
            text: "Save As csv"
        },
        {
            text: "Save As pdf"
        }
    ]];
    public itemSelect(args: MenuEventArgs) {
        if (args.item.text === 'Save As xlsx')
            this.spreadsheetObj!.save({url:
                'https://services.syncfusion.com/angular/production/api/spreadsheet/save',
                fileName: "Sample", saveType: "Xlsx"});
            if (args.item.text === 'Save As xls')
                this.spreadsheetObj!.save({url:
                    'https://services.syncfusion.com/angular/production/api/spreadsheet/save',
                    fileName: "Sample", saveType: "Xls"});
            if (args.item.text === 'Save As csv')
                this.spreadsheetObj!.save({url:
                    'https://services.syncfusion.com/angular/production/api/spreadsheet/save', fi
                    leName: "Sample", saveType: "Csv"});
            if (args.item.text === 'Save As pdf')
                this.spreadsheetObj!.save({url:
                    'https://services.syncfusion.com/angular/production/api/spreadsheet/save', fi
                    leName: "Sample", saveType: "Pdf"});
            }
    };

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Server Configuration

In Spreadsheet control, Excel import and export support processed in **server-side**, to use importing and exporting in your projects, it is required to create a server with any of the following web services.

- WebAPI
- WCF Service
- ASP.NET MVC Controller Action

The following code snippets shows server configuration using **WebAPI** service,

```
`csharp
```

```
[Route("api/[controller]")]
```

```
public class SpreadsheetController : Controller
```

```
{
```

```
//To open excel file
```



```

[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("Open")]
public IActionResult Open(IFormCollection openRequest)
{
    OpenRequest open = new OpenRequest();
    open.File = openRequest.Files[0];
    return Content(Workbook.Open(open));
}

//To save as excel file
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("Save")]
public IActionResult Save([FromForm]SaveSettings saveSettings)
{
    return Workbook.Save(saveSettings);
}

```

Server Dependencies

Open and save helper functions are shipped in the Syncfusion.EJ2.Spreadsheet package, which is available in Essential Studio and nuget.org. Following list of dependencies required for Spreadsheet open and save operations.

- Syncfusion.EJ2
- Syncfusion.EJ2.Spreadsheet
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

And also refer [this](#) for more information.

Supported File Formats

The following list of Excel file formats are supported in Spreadsheet:

- MS Excel (.xlsx)
- MS Excel 97-2003 (.xls)
- Comma Separated Values (.csv)

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

Worksheet in Angular Spreadsheet component

Worksheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data.

Add sheet

You can dynamically add or insert a sheet by one of the following ways,

- Click the **Add Sheet** button in the sheet tab. This will add a new empty sheet next to current active sheet.
- Right-click on the sheet tab, and then select **Insert** option from the context menu to insert a new empty sheet before the current active sheet.
- Using [insertSheet](#) method, you can insert one or more sheets at your desired index.

The following code example shows the insert sheet operation in spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { DataSource } from '../datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false" [showRibbon]="false">
    <e-sheets>
      <e-sheet name="Price Details">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <e-column [width]=110></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  `
})
```

```

        <e-column [width]=110></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
    </e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    created() {
        // Applies style formatting to the active sheet before inserting a
        new sheet
        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:H11');
        // inserting a new sheet with data at 1st index
        // You can also insert empty sheets by specifying the start and end
        sheet index instead of sheet model
        this.spreadsheetObj!.insertSheet([
            index: 1,
            name: 'Inserted Sheet',
            ranges: [{ dataSource: this.data }],
            columns: [{ width: 150 }, { width: 110 }, { width: 110 }, {
width: 85 }, { width: 85 }, { width: 85 }, { width: 85 },
                { width: 85 }
        ]]);
        // Applies style formatting for the inserted sheet
        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Inserted Sheet!A1:H1');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'Inserted
Sheet!D2:H11');
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Delete sheet

The Spreadsheet has support for removing an existing worksheet. You can dynamically delete the existing sheet by the following way,

- Right-click on the sheet tab, and then select **Delete** option from context menu.
- Using [delete](#) method to delete the sheets.

Rename sheet

You can dynamically rename an existing worksheet in the following way,

- Right-click on the sheet tab, and then select **Rename** option from the context menu.

Headers

By default, the row and column headers are visible in worksheets. You can dynamically show or hide worksheet headers by using one of the following ways,

- Switch to **View** tab, and then select **Hide Headers** option to hide both the row and column headers.
- Set **showHeaders** property in sheets as **true** or **false** to show or hide the headers at initial load. By default, the **showHeaders** property is enabled in each worksheet.

Gridlines

Gridlines act as a border like appearance of cells. They are used to distinguish cells on the worksheet. You can dynamically show or hide gridlines by using one of the following ways,

- Switch to **View** tab, and then select **Hide Gridlines** option to hide the gridlines in worksheet.
- Set **showGridLines** property in sheets as **true** or **false** to show or hide the gridlines at initial load. By default, the **showGridLines** property is enabled in each worksheet.

The following code example shows the headers and gridlines operation in spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false"
[showSheetTabs]="false">
<e-sheets>
  <!-- Hiding the headers and gridlines in 'Price Details'
sheet -->
  <e-sheet [showGridLines]="false" [showHeaders]="false">
    <e-ranges>
      <e-range [dataSource]="data"></e-range>
    </e-ranges>
  </e-sheet>
</e-sheets>`
})
```

```

        <e-column [width]=150></e-column>
        <e-column [width]=110></e-column>
        <e-column [width]=110></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:H11');
      // The gridlines have been removed to set border for the range of
cells
      this.spreadsheetObj!.setBorder({ border: '1px solid #e0e0e0' },
'A1:H11');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Sheet visibility

Hiding a worksheet can help prevent unauthorized or accidental changes to your file.

There are three visibility state as like Microsoft Excel,

State	Description
Visible	You can see the worksheet once the component is loaded.
Hidden	This worksheet is not visible, but you can unhide by selecting the sheet from List All Sheets dropdown menu.
VeryHidden	This worksheet is not visible and cannot be unhidden. Changing the state property to Visible is the only way to view this sheet.

The following code example shows the three types of sheet visibility state.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from '../datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[openUrl]="openUrl"
[saveUrl]="saveUrl" [showFormulaBar]="false"
[showRibbon]="false">
    <e-sheets>
      <!-- By default, state is set as 'visible'. We don't need
to said it in the sample. -->
      <e-sheet name="Visible Sheet" state="Visible">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
        </e-columns>
      </e-sheet>
      <!-- Sets sheet state as 'VeryHidden'. It can't be
unhidden. -->
      <e-sheet name="Very Hidden Sheet" state="VeryHidden">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
        </e-columns>
      </e-sheet>
      <!-- Sets sheet state as 'Hidden'. It can be unhidden
dynamically. -->
      <e-sheet name="Hidden Sheet" state="Hidden">
        <e-ranges>

```

```

        <e-range [dataSource]="data"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=150></e-column>
        <e-column [width]=110></e-column>
        <e-column [width]=110></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
        <e-column [width]=85></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    openUrl =
      'https://services.syncfusion.com/angular/production/api/spreadsheet/open';
    saveUrl =
      'https://services.syncfusion.com/angular/production/api/spreadsheet/save'
    created() {
      // Applies style formatting to active visible sheet
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:H11');
      // Applies style formatting to active hidden sheet
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Hidden Sheet!A1:H1');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'Hidden
Sheet!D2:H11');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sheet protection](#)
- [Rows and columns](#)
- [Cell range](#)

- [Formatting](#)

Cell range in Angular Spreadsheet component

A group of cells in a sheet is known as cell range.

Wrap text

Wrap text allows you to display large content as multiple lines in a single cell. By default, the wrap text support is enabled. Use the [allowWrap](#) property to enable or disable the wrap text support in spreadsheet.

Wrap text can be applied or removed to a cell or range of cells in the following ways,

- Using the `wrap` property in `cell`, you can enable or disable wrap text to a cell at initial load.
- Select or deselect wrap button from ribbon toolbar to apply or remove the wrap text to the selected range.
- Using the [wrap](#) method, you can apply or remove the wrap text once the component is loaded.

The following code example shows the wrap text functionality in spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataSource } from './datasource';

@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false">
    <e-sheets>
      <e-sheet name="Movie List">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [index]=1 [width]=100></e-column>
          <e-column [width]=140></e-column>
          <e-column [width]=90></e-column>
          <e-column [width]=150></e-column>
          <e-column [width]=120></e-column>
          <e-column [width]=90></e-column>
          <e-column [width]=180></e-column>
        </e-columns>
        <e-rows>
          <e-row [height]=30></e-row>
          <e-row>
            <e-cells>
              <e-cell [index]=7 [wrap]="true"></e-cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  `
})
export class AppContainerComponent {
  created() {
    console.log('Spreadsheet created');
  }
}
```



```

        </e-cells>
      </e-row>
    <e-row>
      <e-cells>
        <e-cell [index]=7 [wrap]="true"></e-cell>
      </e-cells>
    </e-row>
    <e-row>
      <e-cells>
        <e-cell [index]=7 [wrap]="true"></e-cell>
      </e-cells>
    </e-row>
    <e-row>
      <e-cells>
        <e-cell [index]=7 [wrap]="true"></e-cell>
      </e-cells>
    </e-row>
  </e-rows>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
      this.spreadsheetObj!.cellFormat({ verticalAlign: 'middle' },
'A1:H5');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'A2:B5');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:D5');
      // To wrap the cells from E2 to E5 range
      this.spreadsheetObj!.wrap('E2:E5');
      // To unwrap the H3 cell
      this.spreadsheetObj!.wrap('H3', false);
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Wrap text

The following features have some limitations in wrap text:

- Sorting with wrap text applied data.
- Merge with wrap text.

Merge cells

Merge cells allows users to span two or more cells in the same row or column into a single cell. When cells with multiple values are merged, top-left most cell data will be the data for the merged cell. By default, the merge cells option is enabled. Use [allowMerge](#) property to enable or disable the merge cells option in spreadsheet.

You can merge the range of cells in the following ways,

- Set the `rowSpan` and `colSpan` property in `cell` to merge the number of cells at initial load.
- Select the range of cells and apply merge by selecting the desired option from ribbon toolbar.
- Use [merge](#) method to merge the range of cells, once the component is loaded.

The available merge options in spreadsheet are,

Type	Action
----- -----	
Merge All	Combines all the cells in a range in to a single cell (default).
Merge Horizontally	Combines cells in a range as row-wise.
Merge Vertically	Combines cells in a range as column-wise.
UnMerge	Splits the merged cells into multiple cells.

The following code example shows the merge cells operation in spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataSource } from '../datasource';

@Component({
  imports: [
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false">
  <e-sheets>
    <e-sheet name="Merge Cells">
      <e-ranges>
        <e-range [dataSource]="data"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=90></e-column>
        <e-column [width]=150></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
})
export class AppContainerComponent {
  created() {
    console.log('Spreadsheet created');
  }
}
```

```

    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=120></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
  </e-columns>
  <e-rows>
    <e-row [height]=35></e-row>
    <e-row [height]=35>
      <e-cells>
        <!-- Merging the 2nd cells of rows 2 and 3 through
cell binding. -->
        <e-cell [index]=1 [rowSpan]=2></e-cell>
        <!-- Merging the 2nd row's 3rd and 4th cells
through cell binding. -->
        <e-cell [colSpan]=2></e-cell>
        <!-- Merging the 2nd row's 7th, 8th and 9th cells
through cell binding. -->
        <e-cell [index]=6 [colSpan]=3></e-cell>
        <!-- Merging the 2nd and 3rd rows 11th, 12th and
13th cells through cell binding. -->
        <e-cell [index]=10 [rowSpan]=2 [colSpan]=3></e-
cell>

        <e-cell [index]=13 [colSpan]=2></e-cell>
        <e-cell [index]=17 [colSpan]=2></e-cell>
      </e-cells>
    </e-row>
    <e-row [height]=35>
      <e-cells>
        <e-cell [index]=3 [colSpan]=3></e-cell>
        <e-cell [index]=6 [colSpan]=4></e-cell>
        <e-cell [index]=13 [colSpan]=3></e-cell>
        <e-cell [index]=17 [colSpan]=2></e-cell>
      </e-cells>
    </e-row>
    <e-row [height]=35>
      <e-cells>
        <e-cell [index]=2 [colSpan]=3></e-cell>
        <e-cell [index]=5 [colSpan]=2></e-cell>
        <e-cell [index]=7 [colSpan]=3></e-cell>
        <e-cell [index]=15 [colSpan]=2></e-cell>
        <e-cell [index]=17 [colSpan]=2></e-cell>
      </e-cells>
    </e-row>
    <e-row [height]=35>
      <e-cells>
        <e-cell [index]=2 [colSpan]=3></e-cell>
        <e-cell [index]=6 [colSpan]=4></e-cell>

```

```

        <e-cell [index]=16 [colSpan]=2></e-cell>
      </e-cells>
    </e-row>
    <e-row [height]=35>
      <e-cells>
        <e-cell [index]=2 [colSpan]=4></e-cell>
        <e-cell [index]=7 [colSpan]=3></e-cell>
        <e-cell [index]=15 [colSpan]=2></e-cell>
        <e-cell [index]=17 [colSpan]=2></e-cell>
      </e-cells>
    </e-row>
  </e-rows>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:S1');
      this.spreadsheetObj!.numberFormat('h:mm AM/PM', 'C1:S1');
      this.spreadsheetObj!.cellFormat({ verticalAlign: 'middle' },
'A1:S11');
      // Merging the `K4:M4` cells using method
      this.spreadsheetObj!.merge('K4:M4');
      // Merging the 5th and 6th row cells across 11th, 12th and 13th
column
      this.spreadsheetObj!.merge('K5:M6', 'Vertically');
      // Merging the 18th and 19th column cells across 2nd, 3rd and 4th
row
      this.spreadsheetObj!.merge('N4:O6', 'Horizontally');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Merge

The following features have some limitations in Merge:

- Merge with filter.
- Merge with wrap text.

Data Validation

Data Validation is used to restrict the user from entering the invalid data. You can use the [allowDataValidation](#) property to enable or disable data validation.

* The default value for `allowDataValidation` property is `true`.

Apply Validation

You can apply data validation to restrict the type of data or the values that users enter into a cell.

You can apply data validation by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Data Validation item.
- Use the [addDataValidation\(\)](#) method programmatically.

Clear Validation

Clear validation feature is used to remove data validations from the specified ranges or the whole worksheet.

You can clear data validation rule by one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Validation item.
- Use the [removeDataValidation\(\)](#) method programmatically.

Highlight Invalid Data

Highlight invalid data feature is used to highlight the previously entered invalid values.

You can highlight an invalid data by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Highlight Invalid Data item.
- Use the [addInvalidHighlight\(\)](#) method programmatically.

Clear Highlighted Invalid Data

Clear highlight feature is used to remove the highlight from invalid cells.

You can clear the highlighted invalid data by using the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Highlight item.
- Use the [removeInvalidHighlight\(\)](#) method programmatically.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { conditionalFormatData } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #default [openUrl]="openUrl"
[saveUrl]="saveUrl" (created)="created()" ">
    <e-sheets>
```

```

    <e-sheet name="PriceDetails" >
      <e-rows>
        <e-row>
          <e-cells>
            <e-cell value="Seller Name" [style]="style"></e-
cell>
            <e-cell value="Customer Id" [style]="style"></e-
cell>
            <e-cell value="Customer Name"
[style]="style"></e-cell>
            <e-cell value="Product Name"
[style]="style"></e-cell>
            <e-cell value="Product Price"
[style]="style"></e-cell>
            <e-cell value="Sales Date" [style]="style"></e-
cell>
            <e-cell value="Billing Time"
[style]="style"></e-cell>
            <e-cell value="Total Price" [style]="style"></e-
cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell value="John"></e-cell>
            <e-cell value="1" [validation]="validation"></e-
cell>
            <e-cell value="Nash"></e-cell>
            <e-cell value="Digger"
[validation]="listValidation"></e-cell>
            <e-cell value="50000"
[validation]="listValidation1"></e-cell>
            <e-cell value="04/11/2019"></e-cell>
            <e-cell value="11:34:32 AM"></e-cell>
            <e-cell value="1,45,000.00"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell value="Mike"></e-cell>
            <e-cell value="2" [validation]="validation"></e-
cell>
            <e-cell value="Jim" ></e-cell>
            <e-cell value="Cherry picker"
[validation]="listValidation2"></e-cell>
            <e-cell value="45000"
[validation]="validation"></e-cell>
            <e-cell value="04/11/2019"></e-cell>
            <e-cell value="10:15:00 AM"></e-cell>
            <e-cell value="1,40,040.00"></e-cell>
          </e-cells>
        </e-row>
        <e-row>
          <e-cells>
            <e-cell value="shane"></e-cell>
            <e-cell value="3" [validation]="validation"></e-
cell>

```

```

        <e-cell value="Sean"></e-cell>
        <e-cell value="Kango"
[validation]="validation3"></e-cell>
        <e-cell value="450"
[validation]="validation4"></e-cell>
        <e-cell value="06/25/2019"></e-cell>
        <e-cell value="01:30:11 PM"></e-cell>
        <e-cell value="545.00"></e-cell>
    </e-cells>
</e-row>
<e-row>
    <e-cells>
        <e-cell value="John"></e-cell>
        <e-cell value="1" [validation]="validation"></e-
cell>
        <e-cell value="Nash"></e-cell>
        <e-cell value="JCB"
[validation]="validation5"></e-cell>
        <e-cell value="90000"
[validation]="validation6"></e-cell>
        <e-cell value="09/22/2019"></e-cell>
        <e-cell value="12:30:02 PM"></e-cell>
        <e-cell value="1,00,095.00"></e-cell>
    </e-cells>
</e-row>
</e-rows>
<e-columns>
    <e-column [width]=88></e-column>
    <e-column [width]=88></e-column>
    <e-column [width]=106></e-column>
    <e-column [width]=98></e-column>
    <e-column [width]=88></e-column>
    <e-column [width]=86></e-column>
    <e-column [width]=81></e-column>
</e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent {
    @ViewChild('default')
    spreadsheetObj: SpreadsheetComponent | undefined;
    public style: any = { fontWeight: "bold", textAlign: "center" };
    public validation = { type: 'WholeNumber', operator: 'NotEqualTo',
    value1: '1' };
    public listValidation = { type: 'List', value1: 'Digger, Digger,
    Cherrypicker' };
    public listValidation1 = { type: 'List', value1: '50000,50000,45000' };
    public listValidation2 = { type: 'List', value1: 'Cherrypicker, JCB,
    Wheelbarrow' };
    public validation2 = { type: 'List', value1: '45000,90000,40' };
    public validation3 = { type: 'List', value1: 'Kango, Ropes' };
    public validation4 = { type: 'List', value1: '450, 95' };
    public validation5 = { type: 'List', value1: 'JCB, Ropes, scaffolding'
    };
    public validation6 = { type: 'List', value1: '90000, 95, 10000' };
    openUrl: any;

```

```

saveUrl: any;
created() {
  //Add Data validation to range.
  this.spreadsheetObj!.addDataValidation({ type: 'TextLength', operator:
'LessThanOrEqualTo', value1: '4' }, 'A2:A5');
  this.spreadsheetObj!.addDataValidation({ type: 'WholeNumber',
operator: 'NotEqualTo', value1: '1' }, 'B2:B5');
  this.spreadsheetObj!.addDataValidation({ type: 'Date', operator:
'NotEqualTo', value1: '04/11/2019' }, 'F2:F5');
  this.spreadsheetObj!.addDataValidation({ type: 'Time', operator:
'Between', value1: '10:00:00 AM', value2: '11:00:00 AM' }, 'G2:G5');
  this.spreadsheetObj!.addDataValidation({ type: 'Decimal', operator:
'LessThan', value1: '100000.00' }, 'H2:H5');
  //Highlight Invalid Data.
  this.spreadsheetObj!.addInvalidHighlight('A1:H5');
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Data validation

The following features have some limitations in Data Validation:

- Entire row data validation.
- Insert row between the data validation.
- Copy/paste with data validation.
- Delete cells between data validation applied range.

Auto Fill

Auto Fill is used to fill the cells with data based on adjacent cells. It also follows a pattern from adjacent cells if available. There is no need to enter the repeated data manually. You can use `allowAutoFill` property to enable/disable the auto fill support. You can also use `showFillOptions` property to enable/disable the fill option and `fillType` property to change the default auto fill option which is available in `autoFillSettings`.

You can do this by one of the following ways,

- Using “AutoFillOptions” menu which is open, while drag and drop the cell using fill handle element.
- Use the `autoFill()` method programmatically.

The available parameters in `autoFill()` method are,

Parameter	Type	Description
-----------	------	-------------

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

fillRange	string	Specifies the fill range.
dataRange	string	Specifies the data range.
direction	AutoFillDirection	Specifies the direction("Up","Right","Down","Left")to be filled.
fillType	AutoFillType	Specifies the fill type("CopyCells","FillSeries","FillFormattingOnly","FillWithoutFormatting") for autofill action.

In Auto Fill we have following options,

- Copy Cells
- Fill Series
- Fill Formatting Only
- Fill Without Formatting

* The default auto fill option is “FillSeries” which can be referred from fillType property.

Copy Cells

To copy the selected cell content to the adjacent cells. You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Copy Cells” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “CopyCells” as fill type in autoFill method to fill the adjacent cells.

Fill Series

To fill the series of numbers, characters, or dates based on selected cell content to the adjacent cells with their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Series” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillSeries” as fill type in autoFill method to fill the adjacent cells.

Fill Formatting Only

To fill the cell style and number formatting based on the selected cell content to the adjacent cells without their content.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Formatting Only” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillFormattingOnly” as fill type in autoFill method to fill the adjacent cells.

Fill Without Formatting

To fill series of numbers, characters, or dates based on the selected cells to the adjacent cells without their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Without Formatting” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillWithoutFormatting” as fill type in `autoFill` method to fill the adjacent cells.

In the following sample, you can enable/disable the fill option on the button click event by using the `showFillOptions` property in `autoFillSettings`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { defaultData } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div>
    <button class='e-btn' (click)='onclick()'>Change
showFillOptions</button>
    <ejs-spreadsheet #spreadsheet (created)="created()" >
      <e-sheets>
        <e-sheet name="Price Details">
          <e-ranges>
            <e-range [dataSource]="budgetData"></e-range>
          </e-ranges>
          <e-columns>
            <e-column [width]=130></e-column>
            <e-column [width]=100></e-column>
            <e-column [width]=100></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet></div>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  budgetData: object[] = defaultData;
  onclick(): void {
    var showFillOptions =
this.spreadsheetObj!.autoFillSettings.showFillOptions;
    this.spreadsheetObj!.autoFillSettings.showFillOptions =
!showFillOptions; //To change whether fill options need to be shown or not.
  }
  created() {
    this.spreadsheetObj!.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A1:H1');
    this.spreadsheetObj!.autoFill('D4:D11', 'D2:D3', 'Down', 'CopyCells');
```

```

        this.spreadsheetObj!.autoFill('E4:E11','E2:E3','Down','FillSeries');

this.spreadsheetObj!.autoFill('B4:B11','B2:B3','Down','FillFormattingOnly');

this.spreadsheetObj!.autoFill('C4:C11','C2:C3','Down','FillWithoutFormatting');
    };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Autofill

The following features have some limitations in Autofill:

- Flash Fill option in Autofill feature.
- Fill with Conditional Formatting applied cells.

Clear

Clear feature helps you to clear the cell contents (formulas and data), formats (including number formats, conditional formats, and borders) in a spreadsheet. When you apply clear all, both the contents and the formats will be cleared simultaneously.

Apply Clear Feature

You can apply clear feature by using one of the following ways,

- Select the clear icon in the Ribbon toolbar under the Home Tab.
- Using the [clear\(\)](#) method to clear the values.

Clear has the following types in the spreadsheet,

| Options | Uses |

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

| **Clear All** | Used to clear all contents, formats, and hyperlinks. |

| **Clear Formats** | Used to clear the formats (including number formats, conditional formats, and borders) in a cell. |

| **Clear Contents** | Used to clear the contents (formulas and data) in a cell. |

| **Clear Hyperlinks** | Used to clear the hyperlink in a cell. |

Methods

Clear the cell contents and formats in the Spreadsheet document by using the [clear](#) method. The [clear](#) method has **type** and **range** as parameters. The following code example shows how to clear the cell contents and formats in the button click event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
import { data } from '../datasource';
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<button ej2-dropdownbutton [items]='items' content='Clear' (select)='itemSelect($event)'></button> <ej2-spreadsheet #spreadsheet (created)='created()'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-column [width]=130></e-column><e-column [width]=120></e-column></e-columns></e-sheet></e-sheets></ej2-spreadsheet>"
})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.data = data;
  }
  public items: ItemModel[] = [
    {
      text: "Clear All"
    },
    {
      text: "Clear Formats"
    },
    {
      text: "Clear Contents"
    },
    {
      text: "Clear Hyperlinks"
    }
  ];
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', fontSize: '12pt', 'A1:E1' });
    this.spreadsheetObj!.cellFormat({ color: '#10c469', 'B1:B10' });
  }
  public itemSelect(args: MenuEventArgs) {
    if (args.item.text === 'Clear All')
      this.spreadsheetObj!.clear({ type: 'Clear All', range: 'D1:D10' }); // Clear the content, formats and hyperlinks applied in the provided range.
    if (args.item.text === 'Clear Formats')

```

```
    this.spreadsheetObj!.clear({ type: 'Clear Formats', range: 'B1:B10'
}); // Clear the formats applied in the provided range
    if (args.item.text === 'Clear Contents')
        this.spreadsheetObj!.clear({ type: 'Clear Contents', range: 'A1:A10'
}); // Clear the content in the provided range
    if (args.item.text === 'Clear Hyperlinks')
        this.spreadsheetObj!.clear({ type: 'Clear Hyperlinks', range: 'F2:F6'
}); // Clear the hyperlinks applied in the provided range
    }
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Rows and columns](#)
- [Formatting](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

Editing in Angular Spreadsheet component

You can edit the contents of a cell directly in the cell or by typing in the formula bar. By default, the editing feature is enabled in the spreadsheet. Use the [allowEditing](#) property to enable or disable the editing feature.

Edit cell

You can start editing by one of the following ways,

- Double click a cell to start the edit mode.
- Press F2 key to edit the active cell.
- Use formula bar to perform editing.
- Use BACKSPACE or SPACE key to clear the cell content and start the edit mode.
- Using the [startEdit](#) method.

Save cell

If the cell is in editable state, you can save the edited cell by one of the following ways,

- Perform mouse click on any other cell rather than the current editing cell.
- Press Enter or Tab keys to save the edited cell content.

- Using the [endEdit](#) method.

Cancel editing

To cancel the editing without saving the changes, you can use one of the following ways,

- Press **ESCAPE** key, this will remove the editable state and update the unchanged cell content.
- Using the [closeEdit](#) method.

The following sample shows how to prevent the editing and cell save. Here **E** column prevent the editing by using cancel argument as true in [cellEdit](#) event. In **D** column, prevent saving the edited changes by using cancel argument as true in [beforeCellSave](#) and use [closeEdit](#) method in spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, CellEditEventArgs } from '@syncfusion/ej2-
angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
(cellEdit)="cellEdit($event)" (beforeCellSave)="beforeCellSave($event)"
[showSheetTabs]="false" [showRibbon]="false">
    <e-sheets>
      <e-sheet selectedRange="E11">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=120></e-column>
          <e-column [width]=180></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=120></e-column>
          <e-column [width]=120></e-column>
        </e-columns>
        <e-rows>
          <e-row [index]=10>
            <e-cells>
              <e-cell [index]=3 value="Total Amount:" [style]="{
fontWeight: 'bold' }"></e-cell>
              <e-cell formula="=SUM(E2:E10)"></e-cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
```

```

        </e-sheets>
    </ejs-spreadsheet>`
    })
    export class AppComponent {
        @ViewChild('spreadsheet')
        spreadsheetObj: SpreadsheetComponent | undefined;
        data: object[] = dataSource;
        created() {
            this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
            'center' }, 'A1:E1');
            this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'A2:A10');
            this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'C2:C10');
            this.spreadsheetObj!.numberFormat('$#,##0.00', 'D2:D10');
            this.spreadsheetObj!.numberFormat('$#,##0.00', 'E2:E11');
        }
        // Triggers before going to the editing mode.
        cellEdit(args: CellEditEventArgs): void {
            // Preventing the editing in 5th(Amount) column.
            if (args.address.includes('E')) { args.cancel = true; }
        }
        // Trigger before saving the edited cell content.
        beforeCellSave(args: CellEditEventArgs): void {
            // Prevent saving the edited changes in 4th(Rate) column.
            if (args.address.includes('D')) {
                args.cancel = true;
                // Manually removes the editable state without saving the
                changes. Use `endEdit` method if you want to save the changes.
                this.spreadsheetObj!.closeEdit();
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations

- Text overflow in cells is not supported in Editing.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Cell range](#)
- [Formatting](#)
- [Hyperlink](#)

- [Undo and Redo](#)
- [Unlock the particular cells in the protected sheet](#)

Formulas in Angular Spreadsheet component

Formulas are used for calculating the data in a worksheet. You can refer the cell reference from same sheet or from different sheets.

Usage

You can set formula for a cell in the following ways,

- Using the `formula` property from `cell`, you can set the formula or expression to each cell at initial load.
- Set the formula or expression through data binding.
- You can set formula for a cell by [editing](#).
- Using the [updateCell](#) method, you can set or update the cell formula.

Culture-Based Argument Separator

Previously, although you could import culture-based Excel files into the Spreadsheet component, the formulas wouldn't calculate correctly. This was due to the absence of culture-based argument separators and support for culture-based formatted numeric values as arguments. However, starting from version 25.1.35, you can now import culture-based Excel files into the Spreadsheet component.

Before importing culture-based Excel files, ensure that the Spreadsheet component is rendered with the corresponding culture. Additionally, launch the import/export services with the same culture to ensure compatibility.

When loading spreadsheet data with culture-based formula argument separators using cell data binding, local/remote data, or JSON, ensure to set the [listSeparator](#) property value as the culture-based list separator from your end. Additionally, note that when importing an Excel file, the [listSeparator](#) property will be updated based on the culture of the launched import/export service.

In the example below, the Spreadsheet component is rendered with the `German` culture (`de`). Additionally, you can find references on how to set the culture-based argument separator and culture-based formatted numeric value as arguments to the formulas.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { L10n, loadCldr, setCulture, setCurrencyCode } from
 '@syncfusion/ej2-base';
import { SpreadsheetComponent, getFormatFromType } from '@syncfusion/ej2-
angular-spreadsheet';
import { data } from './datasource';
import deDELocalization from './locale.json';
import cagregorian from './ca-gregorian.json';
import currencies from './currencies.json';
import numbers from './numbers.json';
import timeZoneNames from './timeZoneNames.json';
import numberingSystems from './numberingSystems.json';
L10n.load(deDELocalization);
```



```

setCulture('de');
setCurrencyCode('EUR');
loadCldr(cagregorian, currencies, numbers, timeZoneNames, numberingSystems);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet locale='de' listSeparator=';'
[showRibbon]='false' [showSheetTabs]='false' (created)= 'created()'>
    <e-sheets>
      <e-sheet selectedRange='E14'>
        <e-ranges>
          <e-range [dataSource]='dataSource'></e-
range>
        </e-ranges>
        <e-rows>
          <e-row [index]=12>
            <e-cells>
              <e-cell [index]=3
value='Subtotal:'></e-cell>
              <e-cell
formula='=SUBTOTAL(9;E2:E12) '></e-cell>
              </e-cells>
            </e-row>
            <e-row>
              <e-cells>
                <e-cell [index]=3 value='Discount
(8,5%):'></e-cell>
                <e-cell
formula='=PRODUCT(8,5;E13)/100'></e-cell>
                </e-cells>
              </e-row>
              <e-row>
                <e-cells>
                  <e-cell [index]=3 value='Total
Amount:'></e-cell>
                  <e-cell formula='=E13-E14'></e-cell>
                </e-cells>
              </e-row>
            </e-rows>
          <e-columns>
            <e-column [width]=120></e-column>
            <e-column [width]=180></e-column>
            <e-column [width]=100></e-column>
            <e-column [width]=120></e-column>
            <e-column [width]=120></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>`
  })
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj!: SpreadsheetComponent;

```

```

    dataSource: Object[] = data;
    created(): void {
        this.spreadsheetObj.cellFormat({ textAlign: 'center', fontWeight:
'bold' }, 'A1:E1');
        this.spreadsheetObj.numberFormat(getFormatFromType('Currency'),
'D2:E12');
        this.spreadsheetObj.numberFormat(getFormatFromType('Currency'),
'E13:E15');
    }
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Create User Defined Functions / Custom Functions

The Spreadsheet includes a number of built-in formulas. For your convenience, a list of supported formulas can be found [here](#).

You can define and use an unsupported formula, i.e. a user defined/custom formula, in the spreadsheet by using the [addCustomFunction](#) function. Meanwhile, remember that you should define a user defined/custom formula whose results should only return a single value. If a user-defined/custom formula returns an array, it will be time-consuming to update adjacent cell values.

The following code example shows an unsupported formula in the spreadsheet.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showRibbon]="false"
[showSheetTabs]="false">
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range [dataSource]="data" startCell="A2"></e-range>
        </e-ranges>
      </e-sheet>
    </e-sheets>
  `
})

```

```

        <e-column [width]=150></e-column>
        <e-column [width]=120></e-column>
        <e-column [width]=120></e-column>
        <e-column [width]=120></e-column>
        <e-column [width]=120></e-column>
        <e-column [width]=120></e-column>
    </e-columns>
    <e-rows>
        <e-row [height]=40 [customHeight]="true">
            <e-cells>
                <e-cell value="Monthly Expense" [colSpan]=5
[style]="{ textAlign: 'center', fontWeight: 'bold', verticalAlign: 'middle',
fontStyle: 'italic', fontSize: '15pt' }"></e-cell>
                <e-cell formula="=SUM(E2:E10)"></e-cell>
            </e-cells>
        </e-row>
        <e-row [index]=11>
            <e-cells>
                <e-cell value="Totals" [style]="{ fontStyle:
'italic', fontWeight: 'bold' }"></e-cell>
                <!-- Calculating total of each column data through
cell binding. -->
                <e-cell formula="=SUM(B3:B11)"></e-cell>
                <e-cell formula="=SUM(C3:C11)"></e-cell>
                <e-cell formula="=SUM(D3:D11)"></e-cell>
            </e-cells>
        </e-row>
        <e-row>
            <e-cells>
                <e-cell [index]=1 value="Number of Categories"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
                <e-cell [index]=3 formula="=COUNTA(A3:A11)"></e-
cell>
            </e-cells>
        </e-row>
        <e-row>
            <e-cells>
                <e-cell [index]=1 value="Average Spend"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
                <e-cell [index]=3 formula="=AVERAGE(B3:B11) "
format="$#,##0"></e-cell>
            </e-cells>
        </e-row>
        <e-row>
            <e-cells>
                <e-cell [index]=1 value="Min Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
                <e-cell [index]=3 formula="=MIN(B3:B11) "
format="$#,##0"></e-cell>
            </e-cells>
        </e-row>
        <e-row>
            <e-cells>
                <e-cell [index]=1 value="Max Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
                <e-cell [index]=3 formula="=Max(B3:B11) "
format="$#,##0"></e-cell>

```

```

        </e-cells>
      </e-row>
    </e-rows>
  </e-sheet>
</e-sheets>
</ejs-spreadsheet>`,
  })
}
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;
  // Custom function to calculate percentage between two cell values.
  calculatePercentage(firstCell: string, secondCell: string): number {
    return Number(firstCell) / Number(secondCell);
  }
  // Custom function to calculate round down for values.
  roundDownHandler(value: number, digit: number): number {
    let multiplier: number = Math.pow(10, digit);
    return Math.floor(value * multiplier) / multiplier;
  }
  created() {
    this.spreadsheetObj.cellFormat(
      { fontWeight: 'bold', textAlign: 'center' },
      'A2:F2'
    );
    this.spreadsheetObj.numberFormat('$#,##0', 'B3:D12');
    this.spreadsheetObj.numberFormat('0%', 'E3:E12');
    // Adding custom function for calculating the percentage between two
    // cells.
    this.spreadsheetObj.addCustomFunction(
      this.calculatePercentage,
      'PERCENTAGE'
    );
    // Adding custom function for calculating round down for the value.
    this.spreadsheetObj.addCustomFunction(this.roundDownHandler,
      'ROUNDDOWN');
    // Calculate percentage using custom added formula in E12 cell.
    this.spreadsheetObj.updateCell({ formula: '=PERCENTAGE(C12,D12)' },
      'E12');
    // Calculate round down for average values using custom added formula in
    // F12 cell.
    this.spreadsheetObj.updateCell({ formula: '=ROUNDDOWN(F11,1)' }, 'F12');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Second, if you want to directly compute any formula or expression, you can use the [computeExpression](#) method. This method will work for both built-in and used-defined/custom formula.

The following code example shows how to use `computeExpression` method in the spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showRibbon]="false"
    [showSheetTabs]="false">
      <e-sheets>
        <e-sheet>
          <e-ranges>
            <e-range [dataSource]="data" startCell="A2"></e-range>
          </e-ranges>
          <e-columns>
            <e-column [width]=150></e-column>
            <e-column [width]=120></e-column>
            <e-column [width]=120></e-column>
            <e-column [width]=120></e-column>
            <e-column [width]=120></e-column>
            <e-column [width]=120></e-column>
          </e-columns>
          <e-rows>
            <e-row [height]=40 [customHeight]="true">
              <e-cells>
                <e-cell value="Monthly Expense" [colSpan]=5
[style]="{ textAlign: 'center', fontWeight: 'bold', verticalAlign: 'middle',
fontStyle: 'italic', fontSize: '15pt' }"></e-cell>
                <e-cell formula="=SUM(E2:E10)"></e-cell>
              </e-cells>
            </e-row>
            <e-row [index]=11>
              <e-cells>
                <e-cell value="Totals" [style]="{ fontStyle:
'italic', fontWeight: 'bold' }"></e-cell>
                <!-- Calculating total of each column data through
cell binding. -->
                <e-cell formula="=SUM(B3:B11)"></e-cell>
                <e-cell formula="=SUM(C3:C11)"></e-cell>
                <e-cell formula="=SUM(D3:D11)"></e-cell>
              </e-cells>
            </e-row>
            <e-row>
              <e-cells>
```

```

        <e-cell [index]=1 value="Number of Categories"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
        <e-cell [index]=3 formula="=COUNTA(A3:A11)"></e-
cell>

        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Average Spend"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3 formula="=AVERAGE(B3:B11)"
format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Min Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3 formula="=MIN(B3:B11)"
format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Max Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3 formula="=Max(B3:B11)"
format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
</e-rows>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`,
    })
export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj!: SpreadsheetComponent;
    data: object[] = dataSource;
    // Custom function to calculate percentage between two cell values.
    calculatePercentage(firstCell: string, secondCell: string): number {
        return Number(firstCell) / Number(secondCell);
    }
    created() {
        this.spreadsheetObj.cellFormat(
            { fontWeight: 'bold', textAlign: 'center' },
            'A2:F2'
        );
        this.spreadsheetObj.numberFormat('$#,##0', 'B3:D12');
        this.spreadsheetObj.numberFormat('0%', 'E3:E12');
        // Adding custom function for calculating the percentage between two
        cells.
        this.spreadsheetObj.addCustomFunction(
            this.calculatePercentage,
            'PERCENTAGE'
        );
        // Calculate percentage using custom added formula in E11 cell.
    }
}

```

```

    this.spreadsheetObj.updateCell({ formula: '=PERCENTAGE (C11,D11)' },
    'E11');
    // Calculate expressions using computeExpression in E10 cell.
    this.spreadsheetObj.updateCell(
        { value: this.spreadsheetObj.computeExpression('C10/D10') as string },
        'E10'
    );
    // Calculate custom formula values using computeExpression in E12 cell.
    this.spreadsheetObj.updateCell(
        {
            value: this.spreadsheetObj.computeExpression(
                '=PERCENTAGE (C12,D12)'
            ) as string,
        },
        'E12'
    );
    // Calculate SUM (built-in) formula values using computeExpression in
    D12 cell.
    this.spreadsheetObj.updateCell(
        {
            value: this.spreadsheetObj.computeExpression('=SUM (D3:D11)') as
string,
        },
        'D12'
    );
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Formula bar

Formula bar is used to edit or enter cell data in much easier way. By default, the formula bar is enabled in the spreadsheet. Use the [showFormulaBar](#) property to enable or disable the formula bar.

Named Ranges

You can define a meaningful name for a cell range and use it in the formula for calculation. It makes your formula much easier to understand and maintain. You can add named ranges to the Spreadsheet in the following ways,

- Using the [definedNames](#) collection, you can add multiple named ranges at initial load.
- Use the [addDefinedName](#) method to add a named range dynamically.
- You can remove an added named range dynamically using the [removeDefinedName](#) method.
- Select the range of cells, and then enter the name for the selected range in the name box.

The following code example shows the usage of named ranges support.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
(beforeDataBound)="beforeDataBound()" [showRibbon]="false"
[showSheetTabs]="false">
    <e-definednames>
      <!-- Setting names for 'categories', 'monthly spendings'
and 'annual spendings' ranges. -->
      <e-definedname name="Categories" refersTo="=Budget
Details!A3:A11"></e-definedname>
      <e-definedname name="MonthlySpendings" refersTo="=Budget
Details!B3:B11"></e-definedname>
      <e-definedname name="AnnualSpendings" refersTo="=Budget
Details!C3:C11"></e-definedname>
    </e-definednames>
    <e-sheets>
      <e-sheet name="Budget Details">
        <e-ranges>
          <e-range [dataSource]="data" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <e-column [width]=120></e-column>
          <e-column [width]=120></e-column>
          <e-column [width]=120></e-column>
          <e-column [width]=120></e-column>
        </e-columns>
        <e-rows>
          <e-row [height]=40 [customHeight]="true">
            <e-cells>
              <e-cell value="Monthly Expense" [colSpan]=5
[style]="{ textAlign: 'center', fontWeight: 'bold', verticalAlign: 'middle',
fontStyle: 'italic', fontSize: '15pt' }"></e-cell>
              <e-cell formula="=SUM(E2:E10)"></e-cell>
            </e-cells>
          </e-row>
          <e-row [index]=11>
            <e-cells>
              <e-cell value="Totals" [style]="{ fontStyle:
'italic', fontWeight: 'bold' }"></e-cell>
              <!-- Initializing the formulas using defined
names. -->
              <e-cell formula="=SUM(MonthlySpendings)"></e-cell>
              <e-cell formula="=SUM(AnnualSpendings)"></e-cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  </template>
})

```



```

        <e-cell formula="=SUM(LastYearSpending)"></e-
cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Number of Categories"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3
formula="=COUNTA(Categories)"></e-cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Average Spend"
[colSpan]=2 [style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3
formula="=AVERAGE(MonthlySpending)" format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Min Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3 formula="=MIN(MonthlySpending)"
format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
    <e-row>
        <e-cells>
            <e-cell [index]=1 value="Max Spend" [colSpan]=2
[style]="{ fontWeight: 'bold', textAlign: 'right' }"></e-cell>
            <e-cell [index]=3 formula="=Max(MonthlySpending)"
format="$#,##0"></e-cell>
        </e-cells>
    </e-row>
</e-rows>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
    })
    export class AppComponent {
        @ViewChild('spreadsheet')
        spreadsheetObj: SpreadsheetComponent | undefined;
        data: object[] = dataSource;
        beforeDataBound() {
            // Adding name dynamically for `last year spending` and `percentage
            change` ranges.
            this.spreadsheetObj!.addDefinedName({ name: 'LastYearSpending',
            refersTo: '=D3:D11' });
            this.spreadsheetObj!.addDefinedName({ name: 'PercentageChange',
            refersTo: '=E3:E11' });
        }
        created() {
            // Removing the unwanted `PercentageChange` named range
            this.spreadsheetObj!.removeDefinedName('PercentageChange', '');
        }
    }

```

```

        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A2:E2');
        this.spreadsheetObj!.numberFormat('$#,##0', 'B3:D12');
        this.spreadsheetObj!.numberFormat('0%', 'E3:E12');
        this.spreadsheetObj!.setRowHeight(30,1);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

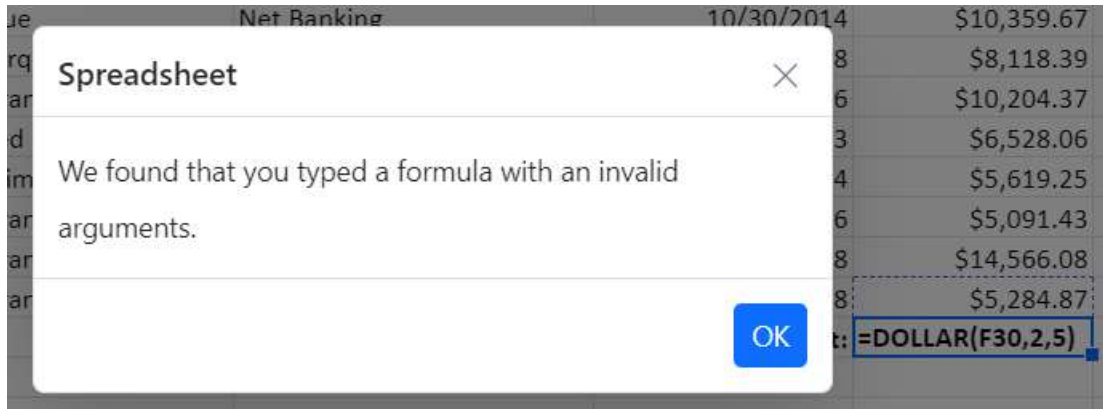
Supported Formulas

The list of supported formulas can be find in following [link](#).

Formula Error Dialog

If you enter an invalid formula in a cell, an error dialog with an error message will appear. For instance, a formula with the incorrect number of arguments, a formula without parenthesis, etc.

Error Message	Reason
----- -----	
We found that you typed a formula with an invalid arguments	Occurs when passing an argument even though it wasn't needed.
We found that you typed a formula with an empty expression	Occurs when passing an empty expression in the argument.
We found that you typed a formula with one or more missing opening or closing parenthesis	Occurs when an open parenthesis or a close parenthesis is missing.
We found that you typed a formula which is improper	Occurs when passing a single reference but a range was needed.
We found that you typed a formula with a wrong number of arguments	Occurs when the required arguments were not passed.
We found that you typed a formula which requires 3 arguments	Occurs when the required 3 arguments were not passed.
We found that you typed a formula with a mismatched quotes	Occurs when passing an argument with mismatched quotes.
We found that you typed a formula with a circular reference	Occurs when passing a formula with circular cell reference.
We found that you typed a formula which is invalid	Except in the cases mentioned above, all other errors will fall into this broad category.



Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Editing](#)
- [Formatting](#)
- [Open](#)
- [Save](#)

Formatting in Angular Spreadsheet component

Formatting options make your data easier to view and understand. The different types of formatting options in the Spreadsheet are,

- Number Formatting
- Text Formatting
- Cell Formatting

Number Formatting

Number formatting provides a type for your data in the Spreadsheet. Use the [allowNumberFormatting](#) property to enable or disable the number formatting option in the Spreadsheet. The different types of number formatting supported in Spreadsheet are,

```
| Types | Format |
|-----|-----|
| General(default) | NA |
| Number | 0.00 |
| Currency | $#,##0.00 |
| Accounting | ($ #,##0.00);($ (#,##0.00);($* "-"??);(@_) |
| ShortDate | mm-dd-yyyy |
| LongDate | dddd, mmmm dd, yyyy |
```

| Time | h:mm:ss AM/PM |

| Percentage | 0.00% |

| Fraction | # ?/? |

| Scientific | 0.00E+00 |

| Text | @ |

Number formatting can be applied in following ways,

- Using the `format` property in `cell`, you can set the desired format to each cell at initial load.
- Using the `numberFormat` method, you can set the number format to a cell or range of cells.
- Selecting the number format option from ribbon toolbar.

Custom Number Formatting

Spreadsheet supports custom number formats to display your data as numbers, dates, times, percentages, and currency values. If the pre-defined number formats do not meet your needs, you can set your own custom formats using custom number formats dialog or `numberFormat` method.

The different types of custom number formatting supported in Spreadsheet are,

| Types | Format |

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

| General(default) | NA |

| Number | 0 |

| Number | 0.00 |

| Number | #,##0 |

| Number | #,##0.00 |

| Number | #,##0_);(#,##0) |

| Number | `#,##0_);Red` |

| Number | #,##0.00_);(#,##0.00) |

| Number | `#,##0.00_);Red` |

| Currency | \$#,##0_);(\$#,##0) |

| Currency | `\$#,##0_);Red` |

| Currency | \$#,##0.00_);(\$#,##0.00) |

| Currency | `\$#,##0.00_);Red` |

| Percentage | 0% |

| Percentage | 0.00% |

| Scientific | 0.00E+00 |

Scientific	###0.0E+0
Fraction	# ?/?
Fraction	# ??/??
ShortDate	dd-mm-yy
Custom	dd-mmm-yy
Custom	dd-mmm
Custom	mmm-yy
Custom	h:mm AM/PM
Custom	h:mm:ss AM/PM
Custom	h:mm
Custom	h:mm:ss
Custom	dd-mm-yy h:mm
Custom	mm:ss
Custom	mm:ss.0
Text	@
Custom	[h]:mm:ss
Accounting	(\$ #,##0);(\$ (#,##0);(\$* "-");(@_)
Accounting	(#,##0);((#,##0);(* "-");(@_)
Accounting	(\$ #,##0.00);(\$ (#,##0.00);(\$* "-"?);(@_)
Accounting	(#,##0.00);((#,##0.00);(* "-"?);(@_)

Custom Number formatting can be applied in following ways,

- Using the [numberFormat](#) method, you can set your own custom number format to a cell or range of cells.
- Selecting the custom number format option from custom number formats dialog or type your own format in dialog input and then click apply button. It will apply the custom format for selected cells.

The following code example shows the number formatting in cell data.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getFormatFromType, NumberFormatType } from '@syncfusion/ej2-angular-spreadsheet';
import { dataSource } from './datasource';
  
```

```

@Component({
  imports: [

    SpreadsheetAllModule

  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false"
    [showRibbon]="false" [showSheetTabs]="false"
[allowInsert]="false" [allowDelete]="false">
    <e-sheets>
      <e-sheet [showGridLines]="false" selectedRange="U15">
        <e-ranges>
          <e-range [dataSource]="data" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=140></e-column>
          <e-column [width]=140></e-column>
          <e-column [width]=160></e-column>
          <e-column [width]=160></e-column>
          <e-column [width]=160></e-column>
          <e-column [width]=120></e-column>
        </e-columns>
        <e-rows>
          <e-row [height]=35 [customHeight]="true">
            <e-cells>
              <e-cell value="Sales Team Summary" [colSpan]=6
[style]="{ verticalAlign: 'middle', textAlign: 'center', fontSize: '16pt',
fontWeight: 'bold', border: '1px solid #e0e0e0', backgroundColor: '#EEEEEE',
color: '#279377' }"></e-cell>
            </e-cells>
          </e-row>
          <e-row [index]=10>
            <e-cells>
              <e-cell [index]=1 value="Total:" [style]="{
fontWeight: 'bold', fontStyle: 'italic' }"></e-cell>
              <e-cell formula="=SUM(C3:C10)"
[format]="accountingFormat"></e-cell>
              <!-- Applied number format to C11, D11 & E11
through cell binding -->
              <e-cell formula="=SUM(D3:D10)" format='_($*
#,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)'></e-cell>
              <e-cell formula="=SUM(E3:E10)" format='_($*
#,##0.00_);_($* (#,##0.00);_($* "-"??_);_(@_)'></e-cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;

```

```

// If the format string is not known, you can get the format string
using `getFormatFromType` function like below
accountingFormat = getFormatFromType(<NumberFormatType>'Accounting');
// Applied number formatting to the range of cells using 'numberFormat'
method once the component is loaded
created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', fontSize:
'12pt', backgroundColor: '#279377', textAlign: 'center',
    color: '#ffffff', borderBottom: '1px solid #e0e0e0' }, 'A2:F2');
    this.spreadsheetObj!.cellFormat({ borderTop: '1px solid #e0e0e0',
backgroundColor: '#EEEEEE' }, 'A11:F11');
    this.spreadsheetObj!.setBorder({ border: '1px solid #e0e0e0' },
'A2:F11', 'Outer');
    // Applied Accounting format to the cells from C3 to E10 range.
    this.spreadsheetObj!.numberFormat('_($* #,##0.00_);_($*
( #,##0.00);_($* "-"??_);_(@_) ', 'C3:E10');
    // Applied Percentage format to the cells from C3 to E11 range.
    this.spreadsheetObj!.numberFormat('0%', 'F3:F10');
    // applied the custom number format for cell form D3 to D10 range

this.spreadsheetObj!.numberFormat(' [Red] [<=2000] $#,##0.00; [Blue] [>2000] $#,##
0.00', 'D3:D10');
    // applied the custom number format for cell from F3 to F10 range
    this.spreadsheetObj!.numberFormat('#,##0.00_); [Red] ( #,##0.00) ',
'F3:F10');
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Text and cell formatting

Text and cell formatting enhances the look and feel of your cell. It helps to highlight a particular cell or range of cells from a whole workbook. You can apply formats like font size, font family, font color, text alignment, border etc. to a cell or range of cells. Use the [allowCellFormatting](#) property to enable or disable the text and cell formatting option in Spreadsheet. You can set the formats in following ways,

- Using the `style` property, you can set formats to each cell at initial load.
- Using the `cellFormat` method, you can set formats to a cell or range of cells.
- You can also apply by clicking the desired format option from the ribbon toolbar.

Fonts

Various font formats supported in the spreadsheet are font-family, font-size, bold, italic, strike-through, underline and font color.

Text Alignment

You can align text in a cell either vertically or horizontally using the `textAlign` and `verticalAlign` property.

Indents

To enhance the appearance of text in a cell, you can change the indentation of a cell content using `textIndent` property.

Fill color

To highlight cell or range of cells from whole workbook you can apply background color for a cell using `backgroundColor` property.

Borders

You can add borders around a cell or range of cells to define a section of worksheet or a table. The different types of border options available in the spreadsheet are,

| Types | Actions |

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

| Top Border | Specifies the top border of a cell or range of cells. |

| Left Border | Specifies the left border of a cell or range of cells. |

| Right Border | Specifies the right border of a cell or range of cells. |

| Bottom Border | Specifies the bottom border of a cell or range of cells. |

| No Border | Used to clear the border from a cell or range of cells. |

| All Border | Specifies all border of a cell or range of cells. |

| Horizontal Border | Specifies the top and bottom border of a cell or range of cells. |

| Vertical Border | Specifies the left and right border of a cell or range of cells. |

| Outside Border | Specifies the outside border of a range of cells. |

| Inside Border | Specifies the inside border of a range of cells. |

You can also change the color, size, and style of the border. The size and style supported in the spreadsheet are,

| Types | Actions |

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

| Thin | Specifies the `1px` border size (default). |

| Medium | Specifies the `2px` border size. |

| Thick | Specifies the `3px` border size. |

| Solid | Used to create the `solid` border (default). |

| Dashed | Used to create the `dashed` border. |

| Dotted | Used to create the `dotted` border. |

| Double | Used to create the `double` border. |

Borders can be applied in the following ways,

- Using the `border`, `borderLeft`, `borderRight`, `borderBottom` properties, you can set the desired border to each cell at initial load.

- Using the `setBorder` method, you can set various border options to a cell or range of cells.
- Selecting the border options from ribbon toolbar.

The following code example shows the style formatting in text and cells of the spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, CellStyleModel } from '@syncfusion/ej2-
angular-spreadsheet';
import { dataSource } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false" [showRibbon]="false"
[showSheetTabs]="false" [allowInsert]="false"
[allowDelete]="false" [allowEditing]="false">
    <e-sheets>
      <e-sheet [showGridLines]="false" selectedRange="U15">
        <e-ranges>
          <e-range [dataSource]="data" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=100></e-column>
          <e-column [width]=200></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=140></e-column>
          <e-column [width]=90></e-column>
        </e-columns>
        <e-rows>
          <e-row [height]=35 [customHeight]="true">
            <e-cells>
              <!-- Applied styles to 'A1' cell through cell
binding -->
              <e-cell value="Order Summary" [colSpan]=5
[style]="{ fontFamily: 'Axettac Demo', verticalAlign: 'middle', textAlign:
'center', fontSize: '18pt', fontWeight: 'bold', color: '#279377', border:
'1px solid #e0e0e0' }"></e-cell>
            </e-cells>
          </e-row>
          <e-row [height]=30>
            <e-cells>
              <!-- Applied styles to 'C2' cell through cell
property binding -->
              <e-cell [index]=2 [style]="{ textAlign: 'right'
}"></e-cell>
            </e-cells>
          </e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  `
})
export class AppContainerComponent {
  created() {
    console.log('Spreadsheet created');
  }
}
```

```

        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
        <e-row [height]=30></e-row>
    </e-rows>
  </e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    // Applied cell formatting to the range of cells using 'cellFormat'
    // method once the component is loaded
    created() {
      // Setting common styles to table header cells
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', fontSize:
'12pt', backgroundColor: '#279377', color: '#ffffff' }, 'A2:E2');
      // Setting common styles to whole table cells
      this.spreadsheetObj!.cellFormat({ verticalAlign: 'middle',
fontFamily: 'Axettac Demo' }, 'A2:E12');
      // Column wise styles setting
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'A2:A12');
      // Setting text-indent to 2 and 4 column
      const style: CellStyleModel = { textAlign: 'left', textIndent: '8pt'
};
      this.spreadsheetObj!.cellFormat(style, 'B2:B12');
      this.spreadsheetObj!.cellFormat(style, 'D2:D12');
      this.spreadsheetObj!.cellFormat({ fontStyle: 'italic', textAlign:
'right' }, 'C3:C12');
      this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'E2:E12');
      // Applied border to range of cells using 'setBorder' method
      this.spreadsheetObj!.setBorder({ borderLeft: '1px solid #e0e0e0',
borderRight: '1px solid #e0e0e0' }, 'A2:E2');
      this.spreadsheetObj!.setBorder({ border: '1px solid #e0e0e0' },
'A4:E11', 'Horizontal');
      this.spreadsheetObj!.setBorder({ border: '1px solid #e0e0e0' },
'A3:E12', 'Outer');
      this.spreadsheetObj!.cellFormat({ color: '#10c469', textDecoration:
'line-through' }, 'E3:E4');
      this.spreadsheetObj!.cellFormat({ color: '#10c469', textDecoration:
'line-through' }, 'E9');
      this.spreadsheetObj!.cellFormat({ color: '#10c469', textDecoration:
'line-through' }, 'E12');
      this.spreadsheetObj!.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E5');
      this.spreadsheetObj!.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E8');
      this.spreadsheetObj!.cellFormat({ color: '#FFC107', textDecoration:
'underline' }, 'E11');
    }
  }

```

```

        this.spreadsheetObj!.cellFormat({ color: '#62c9e8' }, 'E6');
        this.spreadsheetObj!.cellFormat({ color: '#62c9e8' }, 'E10');
        this.spreadsheetObj!.cellFormat({ color: '#ff5b5b' }, 'E7');
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Formatting

The following features are not supported in Formatting:

- Insert row/column between the formatting applied cells.
- Formatting support for row/column.

Conditional Formatting

Conditional formatting helps you to format a cell or range of cells based on the conditions applied. You can enable or disable conditional formats by using the [allowConditionalFormat](#) property.

* The default value for the `allowConditionalFormat` property is `true`.

Apply Conditional Formatting

You can apply conditional formatting by using one of the following ways,

- Select the conditional formatting icon in the Ribbon toolbar under the Home Tab.
- Using the [conditionalFormat\(\)](#) method to define the condition.
- Using the `conditionalFormats` in sheets model.

Conditional formatting has the following types in the spreadsheet,

Highlight cells rules

Highlight cells rules option in the conditional formatting enables you to highlight cells with a preset color depending on the cell's value.

The following options can be given for the highlight cells rules as type,

* 'GreaterThan', 'LessThan', 'Between', 'EqualTo', 'ContainsText', 'DateOccur', 'Duplicate', 'Unique'.

The following preset colors can be used for formatting styles,

- * "RedFT" - Light Red Fill with Dark Red Text,
- * "YellowFT" - Yellow Fill with Dark Yellow Text,
- * "GreenFT" - Green Fill with Dark Green Text,
- * "RedF" - Red Fill,
- * "RedT" - Red Text.

Top bottom rules

Top bottom rules option in the conditional formatting allows you to apply formatting to the cells that satisfy a statistical condition with other cells in the range.

The following options can be given for the top bottom rules as type,

* 'Top10Items', 'Bottom10Items', 'Top10Percentage', 'Bottom10Percentage', 'BelowAverage', 'AboveAverage'.

Data Bars

You can apply data bars to represent the data graphically inside a cell. The longest bar represents the highest value and the shorter bars represent the smaller values.

The following options can be given for the data bars as type,

* 'BlueDataBar', 'GreenDataBar', 'RedDataBar', 'OrangeDataBar', 'LightBlueDataBar', 'PurpleDataBar'.

Color Scales

Using color scales, you can format your cells with two or three colors, where different color shades represent the different cell values. In the Green-Yellow-Red(GYR) Color Scale, the cell that holds the minimum value is colored as red. The cell that holds the median is colored as yellow, and the cell that holds the maximum value is colored as green. All other cells are colored proportionally.

The following options can be given for the color scales as type,

* 'GYRColorScale', 'RYGColorScale', 'GWRColorScale', 'RWGColorScale', 'BWRColorScale', 'RWBColorScale', 'WRCColorScale', 'RWColorScale', 'GWColorScale', 'WGColorScale', 'GYColorScale', 'YGColorScale'.

Icon Sets

Icon sets will help you to visually represent your data with icons. Every icon represents a range of values. In the Three Arrows(colored) icon, the green arrow icon represents the values greater than 67%, the yellow arrow icon represents the values between 33% to 67%, and the red arrow icon represents the values less than 33%.

The following options can be given for the icon sets as type,

* 'ThreeArrows', 'ThreeArrowsGray', 'FourArrowsGray', 'FourArrows', 'FiveArrowsGray', 'FiveArrows', 'ThreeTrafficLights1', 'ThreeTrafficLights2', 'ThreeSigns', 'FourTrafficLights', 'FourRedToBlack', 'ThreeSymbols', 'ThreeSymbols2', 'ThreeFlags', 'FourRating', 'FiveQuarters', 'FiveRating', 'ThreeTriangles', 'ThreeStars', 'FiveBoxes'.

Custom Format

Using the custom format for conditional formatting you can set cell styles like color, background color, font style, font weight, and underline.

In the MAY and JUN columns, we have applied conditional formatting custom format.

* In the Conditional format, custom format supported for Highlight cell rules and Top bottom rules.

Clear Rules

You can clear the defined rules by using one of the following ways,

- Using the “Clear Rules” option in the Conditional Formatting button of HOME Tab in the ribbon to clear the rule from selected cells.
- Using the [clearConditionalFormat\(\)](#) method to clear the defined rules.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { conditionalFormatData } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false">
    <e-sheets>
      <e-sheet name="Car Sales Record">
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-conditionalformats>
          <e-conditionalformat type="GreaterThan"
cFColor="RedFT" value="700" range="B2:B9"></e-conditionalformat>
          <e-conditionalformat type="Bottom10Items"
cFColor="YellowFT" value="4" range="C2:C9"></e-conditionalformat>
          <e-conditionalformat type="BlueDataBar"
range="D2:D9"></e-conditionalformat>
        </e-conditionalformats>
        <e-columns>
          <e-column [index]=1 [width]=120></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = conditionalFormatData;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:N1');
    this.spreadsheetObj!.conditionalFormat({ type: 'RYGColorScale',
range: 'E2:E9' });
    this.spreadsheetObj!.conditionalFormat({ type: 'ThreeArrows', range:
'H2:H9' });
    //Custom Format
    this.spreadsheetObj!.conditionalFormat({ type: 'Top10Items', value:
'1',
      format: { style: { color: '#ffffff', backgroundColor: '#009999',
fontWeight: 'bold' }}, range: 'F2:F9' });
    this.spreadsheetObj!.conditionalFormat({ type: 'Bottom10Items',
value: '1',
      format: { style: { color: '#ffffff', backgroundColor: '#c68d53',
fontWeight: 'bold' }}, range: 'G2:G9' });
  }
}

```

```
}  
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';  
import { AppComponent } from './app.component';  
import 'zone.js';  
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limitations of Conditional formatting

The following features have some limitations in Conditional Formatting:

- Insert row/column between the conditional formatting.
- Conditional formatting with formula support.
- Copy and paste the conditional formatting applied cells.
- Custom rule support.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)
- [Ribbon customization](#)

Freeze pane in Angular Spreadsheet component

Freeze Panes helps you to keep particular rows or columns visible when scrolling the sheet content in the spreadsheet. You can specify the number of frozen rows and columns using the [frozenRows](#) and [frozenColumns](#) properties inside the [Sheet](#) property.

Apply freezepanes on UI

User Interface:

In the active spreadsheet, click the cell where you want to create freeze panes. Freeze panes can be done in any of the following ways:

- Select the View tab in the Ribbon toolbar and choose the **Freeze Panes** item.
- Use the [freezePanes](#) method programmatically.

FrozenRows

It allows you to keep a certain number of rows visible while scrolling vertically through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen rows. Frozen rows can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the Freeze Rows item.
- You can specify the number of frozen rows using the frozenRows property inside the Sheet property.

FrozenColumns

It allows you to keep a certain number of columns visible while scrolling horizontally through the rest of the worksheet.

User Interface:

In the active spreadsheet, select the cell where you want to create frozen columns. Frozen columns can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the Freeze Columns item.
- You can specify the number of frozen columns using the frozenColumns property inside the Sheet property.

In this demo, the frozenColumns is set as '2', and the frozenRows is set as '2'. Hence, the two columns on the left and the top two rows are frozen.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { tradeData } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet> <e-sheets> <e-sheet
[frozenRows]=2 [frozenColumns]=2> <e-ranges> <e-range
[dataSource]='tradeData'></e-range></e-ranges><e-columns><e-column
[width]=100></e-column><e-column [width]=120></e-column><e-column
[width]=96></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public tradeData?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.tradeData = tradeData;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limitations

Here, we have listed out the limitations with Freeze Panes feature.

- Merging the cells between freeze and unfreeze area.
- If images and charts are added inside the freeze area cells, their portion in the unfreeze area will not move when scrolling.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

Context menu in Angular Spreadsheet component

Context Menu is used to improve user interaction with Spreadsheet using the popup menu. This will open when right-clicking on Cell/Column Header/Row Header/ Pager in the Spreadsheet. You can use [enableContextMenu](#) property to enable/disable context menu.

The default value for the `enableContextMenu` property is `true`.

Context Menu Items in Row Cell

Please find the table below for default context menu items and their actions.

Context Menu items	Action
----- -----	
Cut	Cut the selected cells data to the clipboard, you can select a cell where you want to move the data.
Copy	Copy the selected cells data to the clipboard, so that you can paste it to somewhere else.
Paste	Paste the data from clipboard to spreadsheet.
Paste Special	Values - Paste the data values from clipboard to spreadsheet. Formats - Paste the data formats from clipboard to spreadsheet.
Filter	Perform filtering to the selected cells based on an active cell's value.
Sort	Perform sorting to the selected range of cells by ascending or descending.
Hyperlink	Create a link in the spreadsheet to navigate to web links or cell reference within the sheet or other sheets in the Spreadsheet.

Context Menu Items in Row Header / Column Header

Please find the table below for default context menu items and their actions.

Context Menu items	Action
----- -----	
Cut	Cut the selected row/column header data to the clipboard, you can select a cell where you want to move the data.
Copy	Copy the selected row/column header data to the clipboard, so that you can paste it to somewhere else.
Paste	Paste the data from clipboard to spreadsheet.
Paste Special	Values - Paste the data values from clipboard to spreadsheet. Formats - Paste the data formats from clipboard to spreadsheet.
Insert Columns	Insert new rows or columns into the worksheet.
Delete Columns	Delete existing rows or columns from the worksheet.
Hide Columns	Hide the rows and columns.
UnHide Columns	Show the hidden rows and columns.

Context Menu Items in Pager

Please find the table below for default context menu items and their actions.

Context Menu items	Action
----- -----	
Insert	Insert a new worksheet in front of an existing worksheet in the spreadsheet.
Delete	Delete the selected worksheet from the spreadsheet.
Rename	Rename the selected worksheet.
Protect Sheet	Prevent unwanted changes from others by limiting their ability to edit.
Hide	Hide the selected worksheet.

Context Menu Customization

You can perform the following context menu customization options in the spreadsheet

- Add Context Menu Items
- Remove Context Menu Items
- Enable/Disable Context Menu Items

Add Context Menu Items

You can add the custom items in context menu using the [addContextMenuItems](#) in `contextmenuBeforeOpen` event

In this demo, Custom Item is added after the Paste item in the context menu.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```

```
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet
(contextMenuBeforeOpen)="contextMenuBeforeOpen($args)">
    </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  $args: any;
  contextMenuBeforeOpen(args : any) {
    if (args.element.id === this.spreadsheetObj!.element.id +
    '_contextmenu') {
      // To add context menu items.
      this.spreadsheetObj!.addContextMenuItems([ { text: 'Custom Item' } ],
      'Paste Special', false); //To pass the items, Item before / after that the
      element to be inserted, Set false if the items need to be inserted before
      the text.
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Remove Context Menu Items

You can remove the items in context menu using the [removeContextMenuItems](#) in `contextmenuBeforeOpen` event

In this demo, Insert Column item has been removed from the row/column header context menu.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
```

```

standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet
(contextMenuBeforeOpen)="contextMenuBeforeOpen()" ">
    </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  contextMenuBeforeOpen() {
    // To add context menu items.
    this.spreadsheetObj!.removeContextMenuItems(["Insert Column"], false);
    //Items that needs to be removed, Set `true` if the given `text` is a unique
    id.
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable/Disable Context Menu Items

You can enable/disable the items in context menu using the [enableContextMenuItems](#) in `contextmenuBeforeOpen` event

In this demo, Rename item is disabled in the pager context menu.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet
(contextMenuBeforeOpen)="contextMenuBeforeOpen()" ">
    </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  contextMenuBeforeOpen() {
    // To add context menu items.
    this.spreadsheetObj!.enableContextMenuItems(['Rename'], false, false);
    // Contextmenu Items that needs to be enabled / disabled, Set true / false

```

```
to enable / disable the menu items, Set true if the given text is a unique
id.
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Worksheet](#)
- [Rows and columns](#)

Template in Angular Spreadsheet component

Cell Template is used for adding HTML elements into Spreadsheet. You can add the cell template in spreadsheet by using the `template` property and specify the address using the `address` property inside the `ranges` property. You can customize the HTML elements similar to Syncfusion components (TextBox, DropDownList, RadioButton, MultiSelect, DatePicker etc) by using the `beforeCellRender` event. In this demo, Cell template is applied to C2:C9 and instantiated with HTML input components like TextBox, RadioButton, TextArea. You need to bind the events to perform any operations through HTML elements or Syncfusion components. Here, we have added `change` event in to the MultiSelect control, and we have updated the selected data into the spreadsheet cell through that change event.

The following sample describes the above behavior.

Sample link: [Cell template](#)

<!--

APP.COMPONENT.TS

```
import { DropDownListAllModule, MultiSelectAllModule } from
 '@syncfusion/ej2-angular-dropdowns'
import { RadioButtonAllModule, ButtonAllModule } from '@syncfusion/ej2-
angular-buttons'
import { TextBoxAllModule } from '@syncfusion/ej2-angular-inputs'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { CommonModule } from '@angular/common'
import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [
```

```

CommonModule, SpreadsheetAllModule, TextBoxAllModule,
RadioButtonAllModule, DropDownListAllModule, MultiSelectAllModule,
ButtonAllModule
],
standalone: true,
selector: 'app-container',
template: `<ejs-spreadsheet #spreadsheet [showRibbon]="false"
[allowResizing]="false" [showFormulaBar]="false" [allowOpen]="false"
[allowSave]="false"
[scrollSettings]="{ isFinite: true }" (created)="created()"
[allowEditing]="false" [selectionSettings]="{ mode: 'None' }">
  <e-sheets>
    <e-sheet name="Registration Form" [rowCount]=40 [colCount]=30
[showGridLines]="false">
      <e-rows>
        <e-row [height]=55>
          <e-cells>
            <e-cell [index]=1 value="Interview Registration
Form" [style]="{ fontSize: '12pt', fontWeight: 'bold',
textAlign: 'center', verticalAlign:
'middle', textDecoration: 'underline' }"></e-cell>
          </e-cells>
        </e-row>
        <e-row [height]=55>
          <e-cells>
            <e-cell [index]=1 value="Name:"></e-cell>
          </e-cells>
        </e-row>
        <e-row [height]=45>
          <e-cells>
            <e-cell [index]=1 value="Date of Birth:"></e-
cell>
          </e-cells>
        </e-row>
        <e-row [height]=45>
          <e-cells>
            <e-cell [index]=1 value="Gender:"></e-cell>
          </e-cells>
        </e-row>
        <e-row [height]=45>
          <e-cells>
            <e-cell [index]=1 value="Year of
Experience:"></e-cell>
          </e-cells>
        </e-row>
        <e-row [height]=45>
          <e-cells>
            <e-cell [index]=1 value="Areas of
Interest:"></e-cell>
          </e-cells>
        </e-row>
        <e-row [height]=45>
          <e-cells>
            <e-cell [index]=1 value="Mobile Number:"></e-
cell>
          </e-cells>
        </e-row>
      </e-rows>
    </e-sheet>
  </e-sheets>
`

```

```

        <e-row [height]=45>
            <e-cells>
                <e-cell [index]=1 value="Email:"></e-cell>
            </e-cells>
        </e-row>
        <e-row [height]=82>
            <e-cells>
                <e-cell [index]="1" value="Address:"></e-cell>
            </e-cells>
        </e-row>
    </e-rows>
    <e-columns>
        <e-column [index]=1 [width]=190></e-column>
        <e-column [width]=350></e-column>
    </e-columns>
    <e-ranges>
        <e-range address="C2">
            <ng-template #template>
                <ejs-textbox placeholder="Name"></ejs-textbox>
            </ng-template>
        </e-range>
        <e-range address="C3">
            <ng-template #template>
                <ejs-textbox placeholder="DOB"></ejs-textbox>
            </ng-template>
        </e-range>
        <e-range address="C4">
            <ng-template #template>
                <div>
                    <ejs-radiobutton name="gender" value="male"
label="Male"></ejs-radiobutton>
                    <ejs-radiobutton name="gender"
value="female" label="Female"></ejs-radiobutton>
                </div>
            </ng-template>
        </e-range>
        <e-range address="C5">
            <ng-template #template>
                <ejs-dropdownlist placeholder="Experience"
[dataSource]="experience"></ejs-dropdownlist>
            </ng-template>
        </e-range>
        <e-range address="C6">
            <ng-template #template>
                <ejs-multiselect [showClearButton]="false"
placeholder="Areas of Interest" [dataSource]="languages"></ejs-multiselect>
            </ng-template>
        </e-range>
        <e-range address="C7">
            <ng-template #template>
                <ejs-textbox placeholder="Mobile Number"></ejs-
textbox>
            </ng-template>
        </e-range>
        <e-range address="C8">
            <ng-template #template>
                <ejs-textbox placeholder="Email"></ejs-textbox>
            </ng-template>
        </e-range>
    </e-ranges>

```

```

        </ng-template>
      </e-range>
      <e-range address="C9">
        <ng-template #template>
          <ejs-textbox rows="2" [multiline]="true"></ejs-
textbox>
        </ng-template>
      </e-range>
      <e-range address="C11">
        <ng-template #template>
          <button ejs-button cssClass='e-flat '
style='float:right'>Add</button>
        </ng-template>
      </e-range>
    </e-ranges>
  </e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    public experience: string[] = ['0 - 1 year', '1 - 3 years', '3 - 5
years', '5 - 10 years'];
    public languages: string[] = ['JAVA', 'C#', 'SQL'];
    created() {
      // Applies format to specified range
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold' }, 'B2:B9');
      // Merges B1 and C1 cells
      this.spreadsheetObj!.merge('B1:C1');
    }
  };

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)
- [Collaborative Editing](#)

Illustrations in Angular Spreadsheet component

Illustrations help you to insert an image, shapes, and graphic objects in the Essential JS 2 spreadsheet.

Image

Adding images to a spreadsheet can enhance the visual appeal and help to convey information more clearly.

* The default value for the `allowImage` property is `true`.

Insert Image

You can insert the image by using one of the following ways,

- Selecting the Insert tab in the Ribbon toolbar, and then choose the Image tab.
- Use the `insertImage()` method programmatically.

The available parameters in `insertImage()` method are,

| Parameter | Type | Description |
|-----------------|-------------------------|---|
| images | <code>ImageModel</code> | Specifies the options to insert image in spreadsheet. |
| range(optional) | <code>string</code> | Specifies the range in spreadsheet. |

The available arguments in `ImageModel` are:

- `src`: Specifies the image source.
- `id`: Specifies the image element id.
- `height`: Specifies the height of the image.
- `width`: Specifies the width of the image.
- `top`: Specifies the top position of the image.
- `left`: Specifies the left side of the image.

* In a spreadsheet, you can add many types of image files, including IMAGE, JPG, PNG, GIF, and JPEG files.

Delete Image

- If you want to delete the image, just select the image, and then press the Delete key.
- Use the `deleteImage()` method programmatically.

The available parameters in `deleteImage()` method are,

| Parameter | Type | Description |
|-----------------|---------------------|--|
| id | <code>string</code> | Specifies the id of the image element to be deleted. |
| range(optional) | <code>string</code> | Specifies the range in spreadsheet. |

yiAAAAGWlUWHRDb3B5cmlnaHQAAAAAFVuc3BlY2lmaWVkozTnUQAAs/pJREFUeNrsvQmQpOlZHV
j89593ZlVlVXVXX9MzrRlpDh1IYgUWEgYLIQKWbLsrwPbBBActoNDDlgbFrBgl3vZ5XRgmzA2ax
tY1hAYWFhsLiGB0aBbmqt+nqyuM+//PvZ93i97wBuxEevltDQj/19HRVWXZf6ZWfk93/Oez2vVdY
lmNatZL/5lN3+czjWraWuzmtWsBqzNalYDlMylq1kv0uU2f4JP7fX0b36/53Q3bdfdT7L57XGdTO
u6yq+GfqvKsqi8/4u+p2z+Si+NZTXR4E+dden3f9TJ4t1Z2/L/xalnPr4brWZekRa+I/ZTZ3Mz8N
vtYOP8FdRpliYn11N+9k4QrqzaXsrD/60XhD9x5Yv+0Wnz12zA2qx7sI7f+1P90/3r/2AVR29bLe
ajfLXstwZb/ef4DyfXb2L/1jVMp3N0hwn0B114nQ78wMH44kX4pY3KqTDpbaBMLmkazeZFli9c13
/c84OvuPy2b180f+EGrM36L1zP/Mq3+5ODm78gQHxDlRXj1SrGbJEGSQqURY3xpbN45HVvxOTGTa
R5ghvPPAvHLQhKAe4WnP4QvW4H5/ZGsgkq2K6HOM2RphlqeXyymh31h72n272tL7zyBd920vzFG7
A26z9zPfl/fJs1Pd7/1jt3Dr/q4OadywhbsCoLVZZjvspwPJnK9wie28alhy7gjW/+bDz34cfr2d
rD/PZN5AJIOxC7uCWwny+xtbOB+y6fQ7c/QmlZyOMYleWJ129hemcfYa911B/tfjRP48975B3fkz
afQAPWZv1/WL/6rs9/3XKR/Yskdy/Np7OgSEuKajrba6PIMyzTWpgxRlEUiLiSLdvG5UfP43M++y
344AC+gjpPkUZTdMMQSZYhXixRi0PbG/i4eH6M7TO7iISdHa+FoqzQG23ptWoL8MPOTRTJPxGW/a
7mk2jA2qz/1/Wz73yzs4qc33Zd+9VpVvVn0yXKvEKRE2Jre4y6KiFYFQCuxJQtCXwyw+3DY7hejc
UyxuWHz+PtX/BWvP/xj+HGU1fx0COX4ZQebty4gclegGWYyOPMCA9dPouWMKoyXfCCEGVZoi0+bi
bfa6sWEh+kt109LgB+84Nf+D9kzSfTgLVZf27983e+bSNKkt+x0XpUPzDHghAdOq1Q/uMijieaLa
8EfFkWIctrrOYzZEUpwtwCp5hhKWbyX/j0y3j0wdfjTz/6BK5efRpvfu1DuH59hm7HRTQ5hNXy8f
DDZ7Ax2hWTeoVKGHWZxHJtHzvN2B5eiSmsSes66PX613tdDfedukt3/JE8wk1YG2WrN/47ne8qX
Tb/9J1vXO2fEyOE8AKPAQCmCw6QiQ+al7FSFYF8rzEbDEV0OZYJSUqYd5FnOHwZi7YcmBxfB70i9
8goOzjPX/yFF5+6QzmixzT4yOxbifCyEC35+CNb3wdotNbqMUMzsXEzqoAbuBjICA+Pb6Bnd09pG
IX2/nyaLy99y2Pftn3/UzzSTVg/a96/f6Pfc13L+eLdzphN+i0Q4zGbThuIIXcPEKuwo048WRgB
OoUaKQj3E+XWCvZi1iCPNZiuduHWjwKa5cJBUQRxN8+X/3NnzoY0fI6gyj4TaeFvI5rGYnaMkhME
siPHLlLB577DHSp/GnqFxxYLuN2XyCs1ceRLpYCdF6AuoRJsD3EPrlfLx97p2v+/L/9aebT6wB63
916/Gf/Wbr90j2b1ZZ+sagvxGMds4IyuYCVB/wWji69SyS5QRB0AI/uzQuUMSx+JSu3I6ETRMs10
v5f4V2u4f9wykmywROKOarsOp4s8L5vZfhieuHeOChl+FP/uMz+PDHPojz4y2sqgKjfoDP/YuvQ1
BUuPHkR+C0AuRegN3ds7AFvHFSi3+8gN8aYnrrKoabg/lgy/Odb/6af9IA9h4v5zu/8zubv8KLZP
3JP/1G6/pzz747mq7elJalm2dCm0WK5WKO2deHbj37NCIXSdCkzeu38bt/TlOZ2L6+i0NAFXwIP
awMJ6Pk5MINw+PBNAONns9TKIaTwrThp8s8MjDl5AsIhQtB+PRnj760edwspyJqSzXqCv4dYHzF8
6LmR0hShhHsnE4WSDojNafbeLw6lUMtvfM6zo+CBzhftP+k++5cd9rPv9DzafYgPVTfn3sF7/Dmq
1W7+4PN9/AAgXbEmNXfEMCl1HZsNVGuzWALX5rQaYU47fTG8D3Brhx+xDz0yUKR0wluU8rFN9W7p
GmBSJxRrO6Rmdz5MvH8STGYrXEW5d2MTm4jb37rud0diUb+7fhCXNmRY5ep4Xx1kB8YxeTo4UCeB
5VcOQqAMlzzmen8IMQd/YPUcjh0061AzjBG+e33/8rey9/U1NacY9W03XzIlhXf/0HrWUuvxtl8Y
YsjQQkFnbPbKLTBsoqQSyAu318KubsXMsDu+MORru76IUu7HKO+85uYHt7GxkrmOZLXNs/gtvvYb
C1gXa3j8PjBU6nc2R5jNGgjY88fVsAO8VqKX5vvMTFvT1sDjFUtE7tAPvHclY/cwxXQ09128KgM3
QGffGJhckP72C4cw6LKcuJfUymJSyrhenJdGzVxa997Dd/uNV8ovdmNV03L4JV2PZ/GJ49+4ZBXg
vYTPeLKFenM2G1OY4EpFkOHMxmwmBbKCoBXOAJyASYrO0VJnSE+cJ2C47jIYlW4lsCp2K2dgdDVG
L2joQNT6czJGJSb2xsCVtXeN/Hb+Khc7s4uX4V/eEluUaJtLIxGPZR2h5u3JniwkZfrulimZbIj4
R5+2MBeCav10XFY95t4XgpprIcMEXuYLGcXW53B78tv/mM51NtwPoptz76y9/zA3m8ehMZMRXgMa
pbxifIEwGhfDqbbw6WkxN8fgdpKSZslGD/QFiPJqv4mHMBZy4PKsTw7Xe7GPXBryYwCjJSWUwR9j
pYLBZa3FDGHm7eOcIwsIV9Z7i8tYl0dYozulewt3ceTzx7DflxgnO7Y8xOTwSKy/i+J8/VldxunW
eo5MuSbWOVlvzOVtN7NV2ZqPE0wuaoeM37/vfv/qpP+2+/rQk4NWD91FlP/eb//Bbb9f9ukdEknQ
ozRQIAH92NbfmewfFWqKsputsdJGKwxsVYmFF2w8WEXCvkhi2bSv7rZYpDg4WiIsCAwHsroC822
vBrSwMh0M8deMqM50QyyTA0XwuJq+H/ZMTnN25gOXJbQV9laXwwhb2j5Z45ct2xFcOFMxJUSLodn
B85wYuj8RvRkt+JgcHzePeEGIQIFsl4sseI6/uD5K8+MF/92Pf7Be081nf/LXf3/TLNmB9aa8//f
lvf3h1fPVfdnauBEHbx8WHHhV7WEzRpZi9YuIKLDEa9bG1uyM/S4SxNgQMEzVvZ2Ia91q+mMhTAW
8qzFpgazBCu5VjvliwK2sczlJMFymGALzmYUetLk5WC7Q8F6elgZitcelgCd+6hQuDECd39vWgCD
0B8+EpPO+CmMUZVlktLFphNV9hvmJHjyPm+amY3V2w0yfpC5zI74b9DaHeQF4j00hFfzWf/NCzdy
bf+sQ3fPeZDly58IVf8vU/2rTb/ReuJhr8SVi/+1nf+Vo7q3/r8GQynh3egtseIiUumC8mwlpq+J
oD+I6DZfHAKZHKIaLYP2xG/NLQx6ngYDgJ44p+2hQzFWRQzldNC/rq2kGziEPsOytW50liWmMPTyd
JQQGRJNRdxJKCS3lCFFvMMZ8Ybwt65XK+FwBjCkKWotTrY3RbGzOR6qTD2/iFY8Y9dJ0Cv18aJHB
biImO2WOLA9ROM5PW6bil+rC/XdcSELZCbz93r12/1w1b70mjQ+8qnP/BbX3D74+/5N/c99t1Fsw
MasL4klU//2FffX1XVbyzixZl98RGjqMbR0QJpAiznOW7dmmD/1nWxeSyty12sx8UxrpzBR+3j0
9wuD/BVMzXduiiJ2Ztf9hG4ISohD9dgadd1/DF9E3zFB1h0zovxM9kf04OX+4TCRtmwsTMxdq2g9
AuBYQVqgKcttFUnvZ2t4TVu7j2zDPaQXN6MkEl4N7Z2YJlQ3zUCI7cEFzjudv7GPQ8ef1iC1eV+N
hy8KQRjk6m4J0HgZ6mp6ed08Pppdr2v/raR9490Hr2P/6HC6/4C81m+M9cTQXTJ3C97+e/c5CvZo
+Xln355PZNDMZjpDOCKUYRz5QZmVNdzieYzGbY3D2H2ekcz159BqONTYibiTqFmL0pyqzUIJMtAE
wFSLblCMe6ci3IAZCIaRohl58lCVveXLgCTJuQthws4xh+K0SS5mg50cbDFldrEVj/kAoCd89dWP

6xsOlygU4Y4AYBKex56b5zKASQ85MVNrf70DhNcfXmdVy+uIvldIKHHnoZ2p1AWfhATOmWvObjoy
M5CFI8cN8ZvPxVn47Do3nabvtPnj1/319649u/9qDZFY3P+qJbT/z691nx9PD3b9189nJ/cwe9jV
1c+/AH8KBS4HxmYUnAJhXKtMRCfNG2HyLav40zF3ew2X0M+zdvCZhqCBxRlZb4iAMNCNm2i1Zdiv
mai/9awPPbaA8DnB0PsVhk2myeMoLrOvJpO5gu5+gFHQGUhTpLxFzOBcRtOHmpihH3338Fd25ew/
LoUIDaEb90Jc8hpFlE4rvWqOIV2nJoHO2f4Fj84/vPbsIu5YRwGC6Wg0cYdrHKxaxf4HSxUvB2O0
N0eluYTg6QrZbBwY3l0888+fEPzWenX/EFf+sf/mqzOxpmfVGt9//Lb/6Fk+PFOyYnRwjE9/TaLr
aGFxDJBu5tnUUMDLgSgJQCrfL8xCpfIRQzd7ZIMReW7ThMzzji16Yqu6Jtci57TwVkj1zPDeD4ng
Z8LPFT7fLPRB0y2qt2C5H4pKxgXEaxAL9CIVdl4f/OlgBbu2cUzIe3b2E2maIUBk7kMLAsSx6fYn
drLOzbUskYgl8cY1Se+MVViePTEwx3dmAlmfqDDpYZD4eOppPGoz4u3ncB8XKFqMiIeq0xptmOyp
pfeuD+/+mLv/aHv7fZIQ1YXxTrqX/3vW+ZTQ5+ZXY4CeaysdvdDbQ8W0xR8enGe0jiHFU6U6mVdH
ogJq4AsnIxOThCvEpKW/vwB2NhNTetHUuYtQQlC23PR9jqwGullYdkUUQtLFemS/lcG60rLitbQF
eothLb7LJKuFn8UgKplM+elb9yDqgZO4tibbonXjSuU5srLS1JB1lep7iq0tZnWq+V09BOS+Szwabh
DIAVJqZNQX18LKJ18sg9HWCP3+UNUrevlQAJuImW3hdHqCcBCi5bbR2xnMz++ee+dbv/L7mrxsA9
ZP7rr+h/+MluuTebS8TBMWUxQJvRRJypeRiZMVvwxV+192lp4QNMTAqoiIgu1Ndob2gIcq9D2uD
AMEBYD9Vld1zdN4YxHCTOyha7U5nHhrqJQH5j443XJLLYWyikALARwhbA5Ga5MxGwVBuUhwZyqGN
MCxFyDU7ka1PUCNYV5/VzAmqtWk7xuAl0YfSGWAEsl/HZHbsfY2hjJ6+vADhx9PvrJq2iCC+f20G
rLoWK38eylm4ijHifinz/46MPz17/hL37da9/6t3+u2TGNz/pJW7c+/O6fWS6iyyhjBH4H1TBdt9
Nj4FTLCeMoUpEyIS8BBzQgxN+Refu9noCwFCZuiVkbaw7UcwkeH05gC601FUGsN2Qu1RL4ZOJb2g
KO2q01guvw/qWLWgDuC4hrDTPx921huVjYN0MRpnpofL48L01mvYXAFp+2FKa2HTGJbQ1QVQXN2w
ClPGcSJprPrS0Pg2FPCyOsThcbNK2zSCPMY9lKbicYboyxMx5jKe/t9sExzo83cP+lHTmgOhgf3M
LHP/ihvhv4P3HzuY8c/OWv+ch/q9k1DbN+4ln117+7fe36jeeWR3fG6eREfb0zZx9ANDvE8Z0DbI
66AlTxC4VFZ7M5omWs5u3m5hgdMZHD2kKvI2auANelMew5FOUWJnU0YOSIz2q5tpq3tQCzyJbyVQ
pI1SPUfteayBcmtARURelqKkY4kz8SgNUU/Jb/CZOmKwEs00Qucna002/2PO360VM9EB80TfSapf
E3kYvZTWE1MLIsQIwzMChLIKGTfZdLiPM5TDqy3sYbg1xZm8Xi2ksrL3ASMzqjhwO4XCE/dtTPP
nMTXQGrZv3PfTKh/7aN/0vq2b3NGD9hK4P/Nw7f2dysP8mT9AUJTU6smkFAQI8oda6w+RwH7bYqK
uTifirgh4BQ19MyF5godPvC0g9BAIEulzKbd+AJhTzUkDiOMKO/F6L2VskmmXNk7kQWVCWlGas0K
4wo6WFExCWesQl866VYeJS/FsW5cMKFISZmOcZwS3sWzKPY1WocgEeJdQErAxspdFC2dZiHphNW1
4bsZj3Jf3egjKnXZRiky/kEGIQaraKsJBDIJAD58zuNrqnviqvv4uXsjrhZjNAmQ7GOHq9UO8//
EPveef/d71phGgAesnbl39tXflju7cfiadz8dVNhNwhIIjH/HkGO1Q/M2whXwWwRKztczFRBbTtS
Ws2ml54td5mgflqbvkhWAYyPaMmj59RHVQawd3PzurFACKuZnHwqxlJj6qQFFAXzilgrJe6wor0w
rgSJfTa/qthQFuRdwJSzMQxIBULWYv87j8R5CXVS6vdyA+bixgjbR2wg3bch+BsbyeQvzjWA4ayx
fTWg6mWA6MsNPG6dGR5n6PFgnmwrwsddwc9LBzflNeq/i/0VRNfAbGHDHbxtztbuH0wSg9PJ5//TT
/+W/+h2UWNz/oJWfOT/Z+38mR89NzTApIUy8kSqmM8XgbZx97JZaHd9AR5ownU+xevh9uOkdbfM
EgdOB1u8qcHlnOZfBoAKclMKDyffVvLdujbarsKDt7f7i8gc9gNU8JhRFi+PDGNS0EWeZSAEBpFaS
dyuxaTuBJT2NGGerWWFJAapBKflswppOnaofrYTiEHg7A3U7V14CroLddVU9sucn181+8jErBbYi
lUWYFITOkN8VOjxRJdVxhcLjidzCRbdfFFN3Dm3I48R4DjwylaHQ9nzuxpwG17qxecPbv9k3Lph5
pd1DDrvWfV3/gBJ4/nzz37+B+cW05OkYgpoT2YYDzs4uL9F9Huj8AQ7er60xjuPQA/LDWPyk6v9
0T01L8UktAJ0xlot5smqlUjhCTlf6iTrkX+qQCZitfqknJvlbWBzNdQ5alymEhDMpVrc9kMjjEJy
0LI9XCQgdGhOWZNGpcMnpMVhVmpn/Kw6BYp2l0+rJsUV6UXEH8W8vV4gzuH/qoFGmr2m0dwbgKhY
H18Bhf2MN8wmaDGofHU+SFhSiP2FGPwLdx9tweenEoYt9fHaiXMLMx79uJ5jIab6eaZvTc9/NZv+K
NmNzXMek9XNDt6y2p2PPbEv/SCWMO8586IvxaUCJ0cgVMhPTrA5u4WOpsj+GWivaF+KD4mTWSCVU
BBhrPJoAIwi2kdYVn6pBodEqa0S6rl9xhJkvsJMxKADA4J00F8WccLFMjUP2RKx+M1GVxizEh8Zj
lDFCB2wbQQ1uY11lR/kYBDWL3hfpzmKgjSbeV/mXCthZDK4K4dJHC01+uyIOVsvp3BrT30++TLF6e
EBRhvbiE+PMJb3eXBYir4TaleGK6+W85VGvct6hW6PcqcjeQ+++OsDliwy7/pYs5v+bDWyLvdgZd
HiXbZtB57nqILDeHuEjS0Pg15HfFHxPdNCNuYQw/FZbXVrt30xf9vw2n0BbR920Fat1hZwsYJIA0
U0g+Q2VQ5rDTZVan5CGddRs5i/s/xQ/NuOmKm+YFZui8/p2CYgZckhYlTM+zhaRWQKKVytgKpdgt
HX3/P5XAGV5nBp7trEMOuPK30NLJTgl9ZUBb7Kk3pMKzmsomJreqVm93IRYSVSPN7agGeL9RC4Wn
G1MeoJs7K4I8XB0cT4u5api66LFVJhe7/d3f3Ib/zEqNlNDbPes/Xsv/9pxy7TnXgxEVYrVXTMED
oKepvoCRMxiIuzI5nRtH63RBgLWEHmbsuEfaPQR5b41Uy0V86YCUbsnP69UfV9uqGlqkQJusKOFX
9cGO1gjdIyIMV0it8RQmcRv6WC38zFsoSwZrRYQmM2OHEZ4eamY40pGoqwab5X7lt5jgodMsZEYC
qb83SX56aAmv6qYAGkPGERLWbzSg4R+V+aaRtbfVN70o2xcE9Rb7TV7N3c7CE/ZvljIn8H+Vu0He
yd29WDp78xhO/JgdXpacnlycGd8WQ2fdfDwN9pdlUD1nuy4tkzF+J41WdhPX3Hdn8sTNDdtx0Kiw
qzVhHY/+KK2Sj41N9t6GAPLbQnGMX3r01Ka2stu6v6vxarmwhYYUgGhQibyyqqlJJG3C5sSMPI71g
wz+BO04AhQKwJMSWyZAVJLWfdoiiodhCgKVdjCtZov+sbBtLf/nczHYdNccJiOT8UC9RLiFTWcq/N
dkT0t9XHYNYc38fB8uKyTA/tsWTk/nKpMaBLX21+6MO+LT+vpYX6yOd18AGsiLkefsjzffKhFmTS
0tDNnZ2f7czkc1ZvA9W8Vq91201+37QReV00VSeuKLDgSsPTF3ZWOKyel0QtmgbTjtDmRHKYNVjr
8+Phnl7cumb5kcae0e1peS6uHjMMJbY0jMMmoHv1SoToNFhJINTvlfDWNazV/qazvmWsLuGxeQ/

xgW8xkzq4hoyvQ5KsSELHmWM1mmt0Eqe3qbTK+s76v+r6WqjEJrkvlYT1hdlZU8XriAyAU05qjPh
ZRJP5pJAC3MZsukOcpEmsXymOnruNo2efRRJliJcn8LtdhD3xXeXAOTk+7PzTf/hXnGZXNcx6b8
LrcF5Visk6WlISxcXuxh78OkZ7OFbTEGLmscDB0ZoCAQZV0dz2uvEbWoigPqTtqy+qaRSatRoBEO
jQIq4Lkz4RgBAApZ2uMVyp1rD+XMND9GULffpWosW6MF9t50y7e076oHXto3S0Onlt8lZqxtPcJn
27lY2cT87nsNkfK0zJHlq5D3O6Fa8hh46ViWluV4aRmUMua402V/I+CwFpSw6pilVXDFSJCR7HMQ
ZnLooZPMZikeL04ONaVly5A7Q6nX6vJ3844E6zsqxwvuDr4HDaycs2jhcVdnY3ZUP04fv7yoJax2
tTgLTsc9e2QlYXKADtNWNy3KJdGXOSJqajBf68f2WYk+kVAasGnpRRy7WfSbNVwGPl8vtKywGpcg
hJNBuWpK8qPnJtsyoKcJiuqY1PWmn1kmFma82ayqJ3o8laUWzr9Up9kPyPKR45RDTAxfoAuod+iM
Apdfxkp9PCahZhFrF6sCNgl8c4K2FuV3O6rcEIVn+Ak9lCGPdUTXDmks92N9DvsZd2EYin/Tny4K
bAvwHrC7t+4wf+tpPk8NPkFJuJbbRaQ6TzCcJOH36rRyNZWC3XTWm1XGVPCvDSH61L06PKnCqZy5
i0tqntJd8JK7P+l+CkWr8iff02qIzOYv6y0hY5vd8a+mreKuBLyYorqgpToM/UD69NNmb6BbWaup
x8rq+BeVzHpGkUkLwi72tXpqeWbXrMs9KcV8vyws5OUWlnULRciPnf0TxxzMAT2FMBwfNn51BeTh
G0+3KI2EiFnalGwdrmyeQUeRYzuhlYrFWlDVgbsL7wwaUk77Q63SCPlgiENE8P9rG7t4uwK74pm8
HddUTvdZ73Aa2aBQqe5lBpddJLrNW8tLSMUEFZrs1jPkZzq5YBER8rq2o4WJjYMkxM0lqY22Jahi
Bms3euFb8G+OWapWlea2Co5uVnHVR9d2NYyPj4dceNtkBnBFmB/+fMfprRatSCQSjXvcf1l50Unr
zPQg6BtvjohYCUlWjE9z44mQmIK4w3PRzvH8AXk5pCa0cnSy1JpO+730jg704Y7Za3l+ysJsD0wv
8xHXzfLiaBK+yyFP8riiL0ugNNgdhrs5FJS7OhTX5Tc6g0NS2TdhGHVoFk07wlsZL3yCiusmZhgk
76ZLXW/LJh/e7HqPzJiGyn0OvVAoBSdcyqtS8qP5NDoyqNWLe5LtSn5aX+rJqtUnPc+LOuif7S12
WBxvNvVolze31e+718BfafZ3SyNNv0PBNV5vXZ0cPUzu3TFY5nS218ZzeS59uYnkxQ5rb49xu4c3
CC2zfuaCCqWQlYX/CVxNmF46M74pi2cXR0gr5sOqvOtdhAfUzZuJZd6+anOUqQ8rtubJJalekUOJ
bzEaTq165NX1SpmtC2sJOyNDtp6kQZVwNBPATWwNH/q3dZqI9LM5blhxVLCHk5KlGwNLGkqmEiJm
2q4OXiYzW9w/PDdtbXLjRXzCIKoxyhlf9m89QGqD57a8nCDosu2loWqRaCPI6NBaxVJoOn8hp6/R
YG/RAH0wU8ObSylJQtnjt3DlmeYDmfYVfciP07xzzg6nfjNzmrA+oKv+elszKojNm7HwniBW5lyPf
qZtqkQIQMSCIZtbTUzK6yzE/RlCV4GfKDYDgoiuy40r2mpnZyrr8qmcQP0DBZ7UqvcBJkqcxgY4J
lAEKPJ/Dnrhym/wgHLGoAQDKgLBnaoRCr/1wNE+2ANyKHBBrPo/2TJkdGtdJPE80/NwcB31jTWAJH
5AVcR6/yJLhNQDvVmKfjFLIvuczSN/m+sHhxhv9BFHKfIoRtBp6yyeQn7XH44xn2dZs7MasL7gq6
iq+xhUms6X6GgdLRQg6jfWlfrPba+LHEwrmqZvbCPpolKhbA0XRlLz0w617UyL5+kH+i1TzWDbz4
OeZqZ1NxnZmXwrqrVparkmUmyZ3lNNCVHL1Paen3kry36+TU5NYdyNDGMNWE3w6GHBi2mk2VpHp0
sTyGIQiqDma/Y45DlZahmjz9tsMKDbTb9WQMr3TOXDPEkUsAR2FC/FN22JryDMXBrNi8ViqamdoN
0MpWvAeg+WbMQZS/BS2Xy65cQ3y6K1yiD9Q5qYhRb1CfgcZTj107nhafqS0Zjaod9KELH21w3Mdz
GtKwKYhRG2EejVWjlhrHoNYGU75kUZWbY82CxNpL/phNoIoIUQTNUoLKmn3201X19fx11GVfeTZr
pxTRW8bKkr1oGpqqiNOqEcBlrPyy/9uZjUaarBKVfYlJIYtgavasOulaOPCCm2stKY7qh46RPx7V
lHzZJHpn4mpzNc3z/FMk5QpA2xNtHgewFWx741n840tTKdTrGadNENhRHphzJnCWPimiKEWjtZbM
tam6u+Aqlem8a2KqC1FLR2Fa6LIBjgSU0+NaVZXRqfkiDSgn9X/U2NImuAKYUTdlHGkXl0dt0QVJ
RqYcG+RpUzTb84HpUkHbXxrtTnJUqhLI+cdcXrPK4WZJj64bK+2wS0NoVpwjNwxuCTU2pQqSVmrT
VJkDKnFdZrcx3aXEAwZnmNQDBMFmb7nJlFmuaZrMQ/LxPs7D7YbXZWA9YX3kwp8VxaIs3zNGCdd8
m8I3OUjQvstLZGhYGoUWiZnlQ+0PGf75xR8DKaK0xKlQYW9GvFD/1a9VVjUxRBM5i6vgw++fJ7YW
ayaEUE8VAQwFU8ICoT0a2Dgfi2Ky2GYIcNkVgyGMQGcwGIVXkajLLVbi5M5JlPSb9WLYJSI8lWve
7S0dk6plBCSw/hPD8WkoeVrdfythKK0V/LRky0qqpi0E1eayZnSkvM3/lypJkmPLAOpqcIggB+nO
Fkkg1wo871D7/Hu/DIG/LGDG7WC7YsL7xNBlsIY5A5okWEnMJjdW6KDZhCYWECZVg0mks27KHiAs
3DWm7DH8IK+oAnt70WyrBv/FYmbtmf6rXhBF35WVfVI2q5DW0EaGmxvsXbNKXlWioDSilhBnXYyt
bqow578r0D12lpAwH0MX01lWu237FVzg31MIBGq01Aqa5sDVSZwJcJOqmKIq2BtSmv2k61Ke7gZA
AFOn12z0JHQJkKAECDeX9yClBbqpD7ruTvM6fekzyqFXAYdAnfC7QRn4x9cnLSn995atzsrgasL+
hq9YZTL/DTQPxBFu1HSYH9/X0tatCILLtWCFABhio1WKZ8jxHYgr6nMq9rNH8d46/asnEpDapsK7
6e7GS5LYD0uqqbpL2rBLGYzAwoMXhEUFcsA2R/Kx1Qb/18ch0W79dMrXQobOZpfTCft+TchCfQQc
mmespjttSkmdy1P0ybl4dMYor7qaJrPVBdDd1RD/XrdiWV2tFkqVWv0e0FNOWiVXf1TY5pnqWSS
psb6lg+Hy1NK13YppTS2ogJvRyFgVHT69eanZXYwa/oKvdHSXR6WGqKQzxOa5pBU7VBD010x1jM
mrABBGY2Yylu/5KJtak01VBeL/ceF12Z+CipChKghK07xWhmYetFJJF/qfJpZbaSdPrSauq2Y3/d
Wa/a6MBivrwUR2eWbZNTGqy5RQsbY42XynZrXNEZHih9vymIqlyrBNi9Sre667R4+N0FbZdrzqq
Y4AWoz/1rr6A2+Xt8KtNWuqii96qNsC60K+TyZxFrlxJpkzqj1WraYyT0c33mmqWJqmpWFFXX/lW3
9YCMpdMcxDxfoki5Em4utVJkp7t2pJI71liouZppvU07EopqJSvOVf5Kl2T9yQA1hVMFRXYEzhraA
1/ZdQHFZYMVhkgVsKcPBCokaQdpPQbyZLaeSM/810Fa8lZqowui3laesYEJvNCTGUVYmP8uLQ0Vc
Mvvm52AFxRlCoih1Ib4kMDeEtfsI720EOFOePS5HzL3IyTpD+eyTnAie25uAKuMHacVGpZTGaJMG
+lxM2JBLG8NdYRM74/jIZjXM+kIv2d9LyqssV5GqA3JjctZMO/A0faKtZILQbDZR4bPCq5Thsj
xCcXjT6Ckp+9YaTaXMC1MsjgDD6Y7UF1T/WBvASwVtncZwvJbpXaXaYgm6emqX/T2FppHyxUxV+j
MxL5nvzMXUTgUMeVabiLCYxnkiZmiSaA2xoyltGVJ5HC0FT9NALprkWom1zsfWMM3xrFwq6kKB7Z
bQSiT10VlWiddhADrk4KzSjPQotY3PuiZl4Gn/dImNXkeLPhI5xOaLbL3toQ5zDhzrVrOzGrC+8H

Copyright © 2001 -2024 Syncfusion Inc.

RflxxWqm/mmpRK7awlQistVAgF1GHOax6z7dnYsDL46TEQHavsqRY+6PS6zEye055UaN1uWa2b0h
nNdRzV+aW/qa17jmWmBFBreA3O2qrWsqrlooFcnl/Yv+u10Rf2bMkb4MHRyhWW3Gtva4jVnZsqQW
qH4Xouj7xuYVSPJjflKuWa6a/+pNXsqoZZ7w1Yu4PUSkfisPbUv3NL0/9pffmm0g9Vslc0I3ey+So
6xCf6lVLhhhW2prsCRjSij9YhIo85ghS2Vf9FRkMjMVDot9y9N2aKCzNOcrsf6HIY+BVlewtlc/
qsYTaFE8Wqy6QzWi0YUTZWQRd1l0yLFqQnwXaNysvQwBU8EaeG6lglwh2QxUL5fWKOfH8SCzL1r
53pmzcVqCRa1oOvl2i3Wlr3+u1a0e0jo9RrirkWzF6/S46HqVsxE3o+KhuzeH35fFnrzSbqgHrPQ
owtQdVvQgye6lqqCM0WAhhrrwdS6fBh0yCu3zk5nDkVFuaTFfN0Le9ptj7TJHWZGClSVXpwjL5vbT
R562ypEViL+RCVJHXMFRCTcxfr/JNs7eYrSxS4H04R8crXTiLU1UxpAgbm+ZtMT0VospqUC1glv
ZyTo+tcloFtDSp5fWWYr46qlZYa9634DR2birKyfClUZzcrcX2WqjkeejTjkIPg3YXWVpjtljCF3
OcJnayXMGX1xW25E4C7tZohOVqAe/MEM5eA9YGrPcKrHuvqK2DD+UoVmpSejqkyTaCESGbwmlD2
rZ1PGpafCmlKhOcqTzVAVIHScRq36vWryxfSIqt/LtfJSFSeoGUxd4loAzhk2qiCjahSpajdVrC
ryxcSkJm+5nla+1ksCr+GLWutyXM1NdZLcl8qMLHioa107az+valip/jFHZtjMHWtt87qs8G7Em/
50YJsxHcLYvJLnrRAGnuoBt+VnoZgJ3XZbyzCz6QTFItVIOVVtcaZukli7VOC1d3u5di5WDe7pg
HrvYkv3X7SEmbTCgCXwSUV2A5UhZ9kSb0jzmVflawltgtTkcQmMc9Rv5MOrlq+9HfTI/9p66/Wc3
DE57XySmVOHQL57shGKikSzL4nQDKT4jjFjRPlwKYCmtvskmEUuvZUTIlyMGqkKyt7erCYca6Vmb
70+10VKHeM6hN9VduMxOADXctFvm421yZ4ywjGqaMp74cjMDgcmeoWfstBR4AoZiQOuAmOWABJvJ
LXWiBzU/WrzeDoCp4wePu+B5uumwas9xKtJQtiff5562qlCvVki1UldGqjhe+/TNFoK0Ppz2rL+K
62u07FGLPUWj8yVZDquAuLvmJm9JrkvkUcoyzkKd223CdWRqOCTpE+rxQQsxuyf7ZnxLfZzE69RY
KUDMmCfM/RulwdtqxtbnK/wgxPpiYUu4KIPmt9Pb40PRA4FItRqlqPGGwXCr6sB0LTD/ZaHvp5IC
buQPXRH115eaeFi4yHkJeoafS+P9XHVM61qTM7MI7/+DvNRuqAeu9M4OzyNLJa6WtKhAsflB2Za
2sHWiVkuVT2HM0CKQTzBmQUcFSGF/v7rxU9XHfHxVLSU5XQmqseqIZ6pgJF/RxyaZ301h8Yd6kUj
lPSrpQaJvpF6ozhK1a2MxMbyNjm5mrtoqW8f6aC7bN0CyrSLV5nKMzFHReV4CfmRpgdg4xWMX0DQ
XEYQS/qU5Bi8EOWipVylY9erE6sKrTEZ+0Da/bhy/XDpMC6WSuqaLuqIs8i4wU6nooVhFFCHutI+
/s5e9odlQD1nsXDS5K26LagjVdTzh3dU6MynlyYBTNTHaFUFq0Xk8fV9PXMekQYD2MuFqPrjBqEW
RYDq3SqiHVATbsyYI1l1TyiKqGYkDRZHqGNaxlgJ3miQ5fTYiFg6KK3MRQT09EiBU3dWMY0pknOtA
wfo9FfRodtoxVc8WeZEffWQBhb4taT47QvlyYzudU183oqmuL6fkgNcOt1VUFCDi45FOYntxHNJn
JQEPiunie4muItTZ6Vush9zqL9GW9v6oIbsN5DKziZV9p+zWqgOjWsynrdu0OcdECra+allqY7hW
WHhCALCZy1VjDqlWkW52Ap2pcCJlvV9WHm2IShAdmWH1vawG2tVR5YQlirGLcZgcE2NkaiWcEUzx
ba6035Y7jtttYUmcCQPH+rszZlLZ1NQ1+6QKJAtel/B20dMukLnwcFI9f8XqsZvbYkmDJaP29pmQ
ILBqvYu2q3Qo1CHx0nWE1Zb7wy8284KY9DgrIU/U5f/27V7BS91539V81uasB6blERV3UWA1sXUT
GXyT+zZabDqc/HWL7LsI5hXje/aQpTEpTTyXVkvKwRYd34jKkSpIgjZX+aRvEEOGebViI+sTCmK2
ZkkVLiRUBH97KwVL4FHLUYBhr5pYJwJqCJaX7q6A5hyvYAYWBm2+g4SUzi81wa2hYLiNYm8NrONb
dqq2w4hb4ZJfbU32agS4cmq7iEq2M7NIqsw7EYdXa0HNIiy2uTQYn5wSkqYfte6KLXCbGkwHe0QL
cXaBEHLWE5fm4OXvWZ/6jZTA1Y7+0f9TWfX9W/9kmro4JPVs30T20rg63FslXdyR1IKVP1R2sCzb
ET/3JlJq11emKudpWB+cU5rVSOKMUndDojVbzH2sS2MwF4tBR2bAmjZardyy4YT56TqZrKNXnPgq
RGo+L9nKxW6AHhDLbhsTVNCy3kkWRZF1PQDhbNvDudoUwTHpmpdaZTZPJn+SG1fz12gwEym+Ldv1
EUZpCpMvNptceVZYwC+u2xg5EcNq1WgO54A7ndxnL/CIG4AOzMKQ/voDfqpD7/6/+giQQ3YP0E+K
2tUUb20/LZ2mgTwb77567VfLU0QCSgFBMWq4myZDE/Rh4L83SF8eyBMyc9YaewtX6kY/xEnfgGY4
qKD1gIeAiaomyhptxLGRsf1DYawhaDT57Rd6qXefIoEmALQ0e8m4Aw8IxfyfQOWbMiK6cmkERWlw
OFvuvdemAzSb3Wg6Is1kOW+ZwVB2eF5iDCOjilnr5DQ79YyiOzHJ3NL0LWAFZ3KM8bwJuLRbEQ6+
DgUADrXB0/9qp3NLuoAesnBqxb53/PquJHGQ0uBUisFqo1FwKNqDIXqkOLWXifiskrpm8+OUI6vY
mq3UHQ34EdCqvmvpCBYy0YPWYqxGYDQDTRa7CKqIqXGpwqWp5GX4tsgXytwQtftfIfp8PHHe2lXzM
wT/T81R0vflqefqR/LgcYaCCPA89hMXrWcdVpmPWjKwk+N4/zWNRgZvGLJoSZhqclEglSVEo3/69
QUdMtRimuQLacop2JBjKH42Zb6u27XV3+7PD2A59ZpZzz+kc1v/Mm82UUNWD8hy7K/76upl8qt8
ZaqM8aXu2QWZcIEntMnbIbR/y1YnqI5OAZFJ0Wwt2L8Hp9IeKwkQdlzXCcIT9ZKwt6g76qJLJgnx
VE1HMqbA+FFSKKpohXKx2GtTo90aomipr5/aEKm62OD1S5wm+ZaK3DSgoBWRi61WBTZ/uippMcYd
NSyxmZvsn0UKFEaqF+aLWOpJU6V4emdGkmaZlGFS2JIjVlizraMlUrIpmttA9W5zXzfSfy90g46u
MUBrZXNHkQrXT5+Juf/uUfbXZQA9ZP2LJe/dYF3v+rvytO4Ds0/WLyMWaoU22qeyoFhEftTg5RL
R/FVkvO33+1Qh7mxqV5SRyaz2e0aXf63ng/JxMy/FCBWuZJsgpBi5gyZIFksUCq+keOWSpA2Y4pr
EWXzeZ3oLJzghAQ91r6mqE9q2KYZg4CmawZ3dQntdDqZ9vXy+7a3SAgtaA6let6jNYCf501owwc
CZagfTdF7PmnXMa2eNiAbWqIgoB4NjWlqGaKXy+OUMCDvCrh46l+9/Ynuj/+Zm9zRg/cSbwjv3/V
X79uI52dDnTOGvmXrOIFPNulzWB6cRsukNRLMDtF/xBgRhX4Gqpqj/BZrMTJoEsUdCANxYltp6g
dyAYBWEZHRhH0duR2KX2m3QxQc20Ef1a4FnK4mYxnpJfOxgMKkeGoNERGJHC0f0eIYQSCvITCC4q
W22wlHOpXpEKJvYshSYUoZy8JMjjOlS8XzZrGWMmpzOuJrM+1UMg0F06Prsl5Ynlcn4xX6PwzPXb
i5+zlvf7N9+dVNUKk6yfbFD7zcFnMb06wuH5084xM+HOza6BmPQpXeoZVwT7cnctoiQnqaBucKY
TXyK+2vlRi4lqqh6QupBtq3tOpfDNA2fc0SOUJQHlHVR3fenGCilrJ7VLzq/QN2WZxeZZeA6q9b2
rzK4psK7gyDUrNTm5iNL6oEV+Hmr/CljSJWcGkQ5qzly7rUF/aNk3RbM8rtRLLTbtgugl+W8DZkd

ckZnch/rc1gLU8RsgRkXIgVGwY6G1h1mvNnYdf+7lnL7/6TrNrGrB+01bl+v+nbOZHuaHZskYz0t
J+0wB2TF/1SPzMCDuXHxM266jL56zVily6gGLC0pzUfCvNSgZ7KIDG4U9ppe3rlhbUV5pjpQpKLu
Y0MmHX3NZifHbRwDHF/axwuqtEyABXqWAPxLSW68ap5lzzKKYmb3N86b0kAGlfKX3NTNZmb6xVG
WC1Ve19ubWKjF6t5+VaSO/1RVzvQ+3PUApvvPRcx8U9zfGwTLFz/3hb817qmCJSb473nrq57722x
t94Aasn9z1+L0fEsx9tWXVfZbnGYkWX/3RIksQH95E96FPh7dxdt2YnpKwtORPSw4JLgKGYymK2E
ioyB2cWIAyOUQ200KhW5etdYF/CWc1h8cIsyfmbpIqfZasLGJBvxuqD6t5WNYO81yglj67a+xAQF
Vq6mZ+cgft3obWGL0Jnw3wHLrMAActkeaZgjX6LY3Rdm0JhAI30zQg2fWq2BbqtdSdOgVx841JM8e
u3D3Ayj8Xc1gOh38H4zAW/2SkNWD/5gSY/PCjK8th26j4tWkdnqpr2TEKQE8/HV14p9wsUDKzyoe
+nTels3mkFGkV22PsqjOgmppoInLfQD4GehTLNtAywFNMzIFOKH+gGG6gCAWWQKAuXdqXBKAwpE2
gtL3Fm5RRzETBxqh0nzgnr+46pPfrOb60/c86kgKxyPaC5U109XI9S5mvGegKeFkXRZEeuYyMtLY
2s9F+2SHSzMWA1EQb3fVeHKfdaHQw3Bg1YG7C+CPzWnUdq+/ofrbTvUxjGWqc320FC0LpdMRUD4c
3kSCuAIL4czVZ1+xgtFLBW8oV4ohVI1nKh6Rg7rrUogv4v6ZHlepn4qA7Z0lnP0nESLR8sCaY2Cw
d9VYpgg7unmio9sA62kkTGPCqO1IKx20jXUzg7Ih/7NTrmmA28BWmaorMz9Y/Y9xrt07JcIZ2AF
lGjYLN9lofzVLH+bGavJwOV8hBxGZ0FF8OfHmJjeh+A9YXC2Dr6tfqsnyU5Xq1YxQObbutPa2tUV
9F0OKDJ7CazDC49BrNXRYHApTUN8CmacruFVf83YxRWduoEGqRP9XwbWR3YuS0sCdzpJYAIq8FFI
6KZXNwchlVOsWts1lXLD8P2MvKlrqWFlpUppNWDgALXneEQF5nfHCEUiVSHA20KjVfXGtJMCuZSg
Z9GG1mdFsVHqolkg019bX9T8m21vSSbXraMRcLoS0Wg+/5+rNWf7uJADdgfZGYwrX9A8I0XyFkM6
Zagsq6UBisNYTf62mNLpvCs6M7OFh+EPWqgCfbtZs6Q09VUZJlsoJRIHNVo0kL+WmSCTn18Ryl/D
gVE9trd1GFXRSJSZWUXTbvLDW9wrJAFjfk8wiFb0YuVrZcKxADuiVmtl8jaIfae8rGAuZjdeodI9
REGSPA60HHxd0mBOo2sWuIgafKdA3ZLirw26p2oe2B4osXeayHEL8CUzauV2nwtCx97F081zSYN2
B9cay6tXlqpceLEuXYUVUFSyVOOFZDxdAKYc7BDvoXPRx95Cr++Jf+EMvaw3G6wETA+tfe9nnYee
1jCFem6N8dnkMtgLNYvne8QLmaa4G81x5R8AlFukSaLNj8o89R5gnSNFWTLB07TOVE8QpJnuF0Pk
GWZRhQ8rTrYOfKGQSttpjZcxVbU7mZWmZ6+aD9cBk9qZaplywVta1jLoF3VvxVa2wQykJY46zk4
idOL0QqpUcDnIwDP1trYRkK9nRycnJtza7pAHriyMi/NhfrvP3/uMDB85lqAIKJVKy2bwjOMKCZX
QHTncXvjDS1kM+Xv+XHHZ4PR9H1Bb40+eewrVOgMuvfWOCOEUL0YSzfQ7e1WuoOqGA6VXIrrj6B6o
nnUAsj5r74f4GN9CDCcj5FIOyclSvkk5UWKgT3X0Idh5i//204eTrDnzx7HdfjJT7rvvvxsp0hXv
7pW8/7tZ3tcyZtIz4yga8i5X5o0kf0UdeiT9RiqjV95Imf7KuiPvOrWsko7zNPI6MDRYZu9TEYDn
EkblV2oI+ds4/97je96yeXzS5pwPriYdcSf6+2qt+W3d83KRozC8YZXTSFDblJ3VAYdmuIvTe+Cq
2NHl6/CvGWZ2/jaL5CuRCzdnMbmZjM7a0zSFZH8v+LcNt92HtbCDbGYvKK77h3RqVc/I99CP2rz8
F5+BXCPjhif/8HcC6dh//gK7B84sM4fe/jKpPd3IiWgr+O5mHvf+wSws2esj5zr35/Q86UCHlipq
drjznTNqxk8xrwjidyGVBan2YyV9FSG3tBDERr6YA68sI94QtmYU2EaA7e3u7z76mke+tNkdDV
hfbKbwnyI9ngsK+ty0zLcy2GP1NwVMifqB2gDeEp9QNvbGIw8h/+BTGLzyZbgsjGr95m+j2N5AuS
2m7s6DSMXkbQ/GRs6lNUJ+7gwsAbGrQ8cT5NvbOHjmw7hw+VHYJ7dwbMfY3T2HZHKKw499GBMxQV
eLKT7z7B5eeWkPF15+FuMrOypoxqHIZEItN5wvdVwk28kp4k26rEoj5EafWaPETAixSELHgNCvDk
1JYmXeEwNqXrcFdAZ6jd3zezXQfuUbFuFffXGzMXqgwvuhW8Jp3lOm7f/wZx6rOsXtFixy40dvCQI
VrCudl1klc6pyagZtr5lZi9S7h+B1YqQHjMnqWPPonszlsjtvnm0awMtbChEGC13JYOaa4L5kFbmJ
fyLMcnSD78AbRvCEPu/yay1QL9lQoPCAu6/QH6D17C5uUx/G6oqRSaAASp5bW1GydP5siU+R1Vnt
BRG3z1fM5aM0MoMnkMix8o00P2pMi40zI+LvWjVuJ5t3roDncwizNceuDkv/iir/mOL292RQPWFy
+7wvmbYkKj+sPasfq06RaUKfZPJVEObA6VKS8vWkZvXbwgDHSdEvioerZqLnni53oHU7QXS5Qfeh
ZZR1jw3H1VIyWHT8ISsxbLGZznbuLSM9eQfvAnxReuMBA/MxdTmjnRmqr4Ax+t7b6Y1G0V5aa00/
PRa5ZrCJvmSayFFCz2t7XHNdcKqVonnxmfnndZKC6SewUEoa10VyZOVlh1+Tkthw+Lux2G3VviM
XhR99nhd2/2eyGBqvw6hV+5tdcy/7gx6/Jjn7U8n0TiaUig9dRitIIsWX8WUtNZGG382dh3zw2Rf
wbsuHJshLfkHc2nc+ExFy4UaXpnfrgOip24UQz1IncZzBAQcEHjMDkVA7EtFDRE+Z2+iH8LfEdKT
9Kk5eMycooYWQ2j6MSJmW0WfzVSpXe7LWWkouc0jLstsnXBRI0hClFyoZ3NpsLu9I8pmJhIq/Jbn
lm8p3vX905f/9nfNaX/dlGYb8B64t/VZb1E3aZ/UiVVoG15qKjzMLcfXbZ2OvADYpEfteFvTnSAN
4cnJpRGa5R12frmQLSE3OYmknDkQZ+qISIO5mRM/VrLfYvOJeGLB46ci648He7yFiCyEBSZRhV5W
PW+sYshGQ5YiGPzZny0WFXRqFQ/VQxe7Xpnf9niaLfUsa2nbYZldHqy20P2eIEabaE43R4IKR+b/
A3PusL/lZTAPH/czUjHz/Rq7P9j4vSeq6oSynMr3ubIGT5/HejO6fHyI8t2nEtIVq2f4ml4DHIBCF3Sj3T1
XNctedMx6L92t1LEZzrTzWbhdGmSynMr3ubIGT5/HejO6fHyI8t2nEtIVq2f4ml4DHIBCF3Sj3T1
EzBSRZVb7qdfFDbQ6RUKxifhVysnBkRsE6Y6s2SorUceRISvlFthJfd3GkQmzsAmGO+99+We9/T
3NBmjA+tIxhV/1DnEB3XeVeZWWRaJCZDqMSQLubQYzR6ItcSzmn2kTdMUK3hrCum8L9aaYmWPxVU
cCypGPui3gKlbad8qJco7gpQ7N4/0zLXRefh7BhTOwaVb76wlv8sUZsB7HeAhrmhBXZUTGswz5aq
VdQTR/c2XVSospqE+c0z/mqB7mWlXQey0K7tHU7QmYYy2HLBanSOdzeVxy1N+99Lbm02/M4Jfc6r
/p635u/ns/9vVuUb2BDdlSfaPmEaVTCBysmUz1jzT14WuxPmuJ0WaN8AD1sCVg6oldGPneUmEvqx
yo4j7SBB5/nvdheZZqIdGXVTYW05aFRn6rZcCq0iuxaXmT3+uwKWF9FjLkearmLlUtaDKbemPzRY
UllhIWJoytTfLt/lhTNVXOxoKZnEMr5IslNh5+9fvu/6y3R80n34D1JbnEtHxzXlUf86rqsiWggJ
fq5mfAhsEmBphM8UGuMqI6/KkyI8bZj1qpFrCnpX0cl1iHXTVjjeIglfoHKqRNMDCncOJ9h6xLm5

uMK0Cl2mLnkY0+fdRCxchtqleUFFBbIk1zZPJ/BoqMGGKmLXSFTouU5wq4fSxj/rYGWr2U0WdmS9
zJLUSHRwgc5yi8cOGvNp94Ywa/ZFfvjV+f1Zb1rUWf1D6dKjCU5uy06vLPQM3bZaq+qPqz/NCEOT
mf1cpqI0jGbh5tbXPVJOW11HzOBajxStvaHLnNpNmKmb2QNb71GvCcPWVSLdSGYoQ3ExYmMHMGku
QFZmIap+L35jo9XQDMLM1aL7jUCEjC/BytkbO0UNg0WSCenKKUL28wuPPqL/ngWfOJN2B9af/xg9
4v1rCfY0FQzTEXbBC3tR7IBI0IVqoFMtBUZFr1pKWKjLgKM2owqTASoJSCMXCVL52vmum4C8vKdX
YO3eDKL/vY8kVzmvNrQh5fZcbk5s8AFQHP41Snzilg08xEkzmOQw4CttkVlRnGXHEciN1WEXCKRR
CkLMqID25iceMGwsEo7Vx44H9rPukGrC99dvlvvqKubPfbZcenOn6CudY0NeqAZL16FKeUrVgLfMa
okEjDIbQqH63yZXL98t9KgsV2TTeV+2UL90Dqbm5GTiMT1zeR+1rrpXUNJWtDAFE29VsyvGOULk9
J3zQjaTgt/S+3UyZRhMx3g7JiGdsekm0oqXghI83iJeDbF0VNPo0uRtO7GvHf54Z9uPunGZ/2UWK
3+uV9cLfe/16+tyxTTdjjTpjbN3VBB8Hit+xkJSH3SsWCb4yxijdpyfIYWNLiGnet4TooTMzlSMJ
rp6YX2meoBICydm4da+2bWKlnSMbNgqfpfaMomlS8GmNRVhplAlwtY89pDRp0nKhPmleoZuRT0zt
uc/P7o6SfRk9t+u43wwqWFv7132nzKDVg/JZb3qi+p8Qc/k3MMhg51okqh7WtwSEdFgtHaEv9tA
c8Z5wpWo9YFBCSidnMHqWGdcnAZOgyU1Y2DJqZQJCOliRTWpqW4ZQ6V/xNNodrKaGYy3lqel+Zoi
F7UhyNvm2eE+C2ionrNHXkZjCWP30TFPtYnrjDoK4Qm97DGe0gdaFK/B7/eZDbszgT53lu15WWJ
6mTTTQxIIIXwSLWCPMyema0kljWOIX2qmYt0VkBi2zjG9dkKAaTUFLtZyoMKEFEamAsLCNKUvFBm
FU9qYWZbJuKjCazuTosQLVpG9iYfNUvhdq+jK4ZKkZrL2rRam+bJpE+j1Pcty+eg1OEmNja1Nzuc
G5y3C6o71bt259WfMJN8z6KbOYT7WdjoAhUgCFNFOWuQ6Foi6v5XdNhVEMTBauixqqUkHsVutKJ5
jIcF3QzzWTxzVCbJnAEf3STBg2zUotxs8t1kvJASBAPDxoIT9LolwHGscCOoJuh10RmMq6tgKVUU
tlacS8KQwu0MZs/wRd8Xc3drZVUC LZ3JSvXazyNLDC9nfK0//r51NuwpqpsWo78+zSfBlkBQrPYp
eZdrFw6LI2qbLTxcq15pYlSgSsQyG00pi6zMVC856ULPWluB4MBrHAniCjaJkAOSuKtYgoDftaZl
ZNlprobxIJu7IIguDntXPzHPRXCyMyrNKMHCXJ+68Op+jJix2NBur/Vv0+yvEZlPI990JcOHP+0o
2bt/+V8+ce+IXmg27A+pJfTK2UrgdfTN9IWCyRr7YOSS5VeLsWlnXDIRy/NOWAlrWenM7SwJgFus
rClcWg0VIHwBHMuxO3hrfWEY1xlqsUSw4zyoMMzLwqgUjRxFWSGV+V8qZ2rdFhgpURYz5OGZ2HBa
+dpJjeOck408Wo19PUDwv6i/4AwfZ5JG4be2f006wP4mj5LnmbDVgbn/Wlv8QMzRgk4qCogLlWAV
YkDEt5Fg0SeS2dhM5a3Mpbic5CVJiBUrYdDkfXKcQWkDbkOmKwlrZLT75R/SbVOsWA90KqokCSJHg
6RgHhBABXEFEMw2suCiEx+lwlbZqzL0Ppfy4B3FWOxf4AzwwFGG0Ntf4stIBqNMHj1m+D0d7C9dx
Ge62O+XMC13Eu39p/+ouaTbpj1pe+zZulMfFmFHSmMzfa4LgbvaVv801zn0YIqg6X8rLKNlArb4e
ibJpGWJNJwLcU0rdzcjFMVwGuKhtVHwOTMnSYcUMVywsooHFZkXbkff1foOI9M00ZkXlRGSdgyYw
QAHh5yvWg6hZ/kOL85RtjrcOPWmHMwlv2PYucVr0ULJvrOuQvaIHAYnWA42sRyuWQ74PFLC/6V5t
NuwPqSxSk17xm4fLZlewF9z1D+Q9VfyOxaOvqNBQtT7XVNOAiK6oLEk84+TU1UN/2/2XsTKNvO6j
xwn3PuUPNc9WZNSMgSqJkkJIEGJDBgY7Ad2+nEjtOdwUk67nYn3e6O3Zle9ortFa/EdjdZnSwbD0
1jQIGYCWFLQkqgRgmBxBQAiSB9MaaXs13PkP/37f//Z9zn9zdiWPj9+S6WqWqd+vWvWf497/3/v
a3v911Bud8qjOqgcs54Q37ntvbhVZwv+e8qU6U6w8UNGJtFd00WZ/PZ6KDPSiABGg3qM7hU+hrzT
s96WzvsyzYmCwdPcCump57j92RUR15ze2y5Iy16z7jwLGLpFEbka2NdZmeX5C2M+yRbiYERJecPP
nMO44evXzfYPeN9QL2rLXoq6QOJqnU4FmdgTSjPwMnkWxtrsg4BLMR7sLgOGpRSzMwMBp7X9XyAQ
ShPxaG20XTuft3z4WtPRAAXNg7gMvNYvdcQY+IEhHAPYxT7fwMHkydqWpdgrD4Hj3sYK8tdfc+Lz
16SBojTfaz7rnPWB9fkKO3vElmD13CSXiHL7nIHx9dVtdXZGFhgZRF5FljYxOQRW2OTjT3veu+sV
7gAFMyfgpKDGABsctGEqk3R6SRtqXtDG53b8sZbxLqsCATpH7IFOiBGRUchHivDG/dP7pAceERC6
2PKpqbmJQFGQIGM5t4ojqjXM7ZSTOKeKdQh8C00BeKp84zLkxNyOzCPP8GzKYzeU3Sy2+Uq268Re
ojU9Lp7MrRyy5nWH52bU1mIJnKdh2RWzfH7u5uClQxtjbb1/zLX/3f/sdf/Oe/8pv7d33fWC/Mx8
jkRtFe6z1/2QS/L46dAUWTunciecztyV6rJVsg+oBRJZRSjBoYKUbWHmU+NmpqlCIJ6nkgHzUWW
VfB3FQnBvtdpyqHns1w8LlvuASo2kA4xo7LnTe2ZLp0XFZuPiYy52bnLu67sLqztz3ycyVr5JLLn
+p89B9absN4fClV9Drb7h8dn5h0eXbPXKn4F13dnZJm6y7Y//iY48177rrw7/2R3901/YTX3ny9/
Zv/L6xXnCPuDayhdIIZFkgjo2eVlAlayMjtCeEyNvOWJETUofJhbxod4s5KIpp6hSzN4JFSmaVgGx
tkW2KX3DZyTI2LCT4BIk4jwlFSOC+c9NCv2nGeu0bD39nYFxsxfPhrxxTI6Me28ciotZ/DtuUslPn
yVXHrRpTK7uOTC2h0ypo5dcowh807etizOLblj7HO48twMgKUWc+N6vSmf+8zn5F2/9/t4bbMeJ/
/nK1/xstNf+eqT9+3f/X1jvaAexchoBlqiFBHE/yDlENXJ20X/60j0pDPcSNrbLdnb2pReq8tBxi
A+JL45PXZ/h7INx80AgIpEEeJcVR5eIEikXfclKJqz6LpvyYME6SBH+LpBPaeLjh2Wqel5ajrt9T
LzmlYs8ZfdyHGR8wePyMhYUzbPbsvEiXmT09Ly31OzMxZmD8gbWecSaMuUzPtsr21Ta+eOIP+xC
cekwe85z2y5Tvw1KayetF0h/qR66595bVffPwrX9lAf+R2IYJYe0//vIeg6C/05ae+fpzSVRbjD
AlDmqBtRpb5nKIOEGWFHlpvyM9hL+7W9Le3ZHO3q70Kb8CYKmnYBGKq84SGB4DHUZgDFUH56ohjo
aRk7FXfRi0erJ8epn/PnrskDOyOYknpmSQNuRsx4Xgr7hVJo5dIWtnN2Tp4EEYqs6unZXDR464Qx
yRldV1lmimxidkc2dbRkdHZWxiVNBX16glVXOG+sJnPiO/+7u/K9vbuwzBGS1gqFVSx+vXFpbmrv
7UJx9Z318F+8Z6QTzS579UH3z3se8kEh31TFNoCkNum5hPVxJ2xghb0NDbS1TYeckUjeH4GV0y3Z
aCS/Cg6HcF4Z6Dm1N6XOGvISRft830+oasn1qGfoRcdPFhmXFesjE+yybyVecR+wdfIQeuuVF2yW
4qZMb1n13nNdvdthw4cpi1142NLZkEc6lWl01nnBg21XTHvbJyhHEBvO3jj31e7rzzT11b3eZYS2
1ScNGA24hqids86omMjIw8euDQgTc89OCn9iVK9431AvCsJ7+eFM989rli8qPssuHsDJlKDM+ZNE

YpqMa+VEqLghLY53gLtnhDY3inw17VnGiwb55AwnfGmbW7zhNvsjF888yKM/ZUDlx6iUzOOQMDIy
pusnyzVvxI4xW3ydzSpXJ6ZdkZVV0mXUi7ub1Nr7iweIA9rlubmzI7t0Dj21g/K4sHllgGOnnihP
OWTZe/tuWxzz8id999t5w9u6klJeeV6y42j2MlzYH1hC945vHx0Y9c/tLLf+yDd969PwJ931jP70
e+9mzU/8aDz8SSX8Z5NxxgORT3gSBvIm6MSj0wwdCQIDCX83IW90B7u9dhEz1ILSjWDVD0rlB76be
m78LSLSH19Vbrdvow545ucX5Jms85wVIq69KQpq5PfJ7NXvVZGJ6fkzKntMJE1ztB5e2NTxqammZ
/u7e1Jp9OS6dk56XW6BJa0ntqXkyePy5gLg9ec1/7CY5+Tu+66S3Z3W/oZ4pvb4VXpUZuS+HAYxo
pjmZgYf/9bf/AHfvqXf+nX8v0VsW+s528YfPqbUfqth56Ja43LEFZChBtEfRgdbg9GZ9Qm5t1zTZ
ZBUCBFLloMupRvAWsp7+45I24R1AEpApPSe6096bu8lvNqID86MqaDkqnNFJE73Jq9QvpLr5DxQx
cxTN5yOenU/Cx/f3Z1jchvs9l0Ye8GDQt56e72Do97enaGBry+tiLlxisLJ+Wxx57jB6111dWFH
jMRLy2kjPKvTIdci+NGuav8bOYECSF1ZPvOvRzz/xD/dXxD4afP7ehMPfVwy++SB4RaK9FRDxzR
VUUrdeRYgiItqYW+N+D20lYd8rJpP3qKTP3RciZvBmNp280ZT6OLxyzE47GAae740f1M3J18rY0S
tlZmJK1jY3ZG93V+YOHGBYvbO1K4eOXCT3PHPMDD1rw+WkZ8+elXF8KPjYzTg1t60jI6NyYkTp5
xHfUQ+9MEPSRvMqrgzhlPDDACpa7N7riEwIgSgWqQ8uvOFNhQkYvb2Wv/glptv3P70Zx79n/dXxb
5nPS8fj/9f/2LkqiMTx2ujo4uYjwpeLhhNQIMzEulHODc1Hplgwz1Ek6Jc5Usxi7Xo7kjW2aQkS0
EyfqqzaLqqBlFrwLgznVbuct/0otfJdv2ATM8vStIckfX1dY7yGBkd1larxfddnF8g6QLeEmR8vM
+mylXn0VjuNoz15WUaPb5OHH9OPu9y1Lvu+jBpjcaisgnpeA2MloOWC3PEXlvi6FYsX8fbAbIeW
fmZn7xU5985Nf3V8a+Zz3vHjtPf/XObPGVi7VGXaf2S+gdO+8HEZjAKKJ3WKHtHANrXIIZslUGm
i4jLGMzkh6nEzOCarC9/PDrCJn7IO5K6S78HIZOONfOnCYRrO6vEiVcbBn8+yGM54apVm6Lifd2d
mSOWfQvV5HWrttWVpaoiGe8EASqI4nT56WRx/9rHzorntcTqzgFoy0gfNlICEFWiPKSgz0hMFJU1
PyFKRN3fOo71JlgiMlacZ/8vbbbhr5xMOF/ex91bFvrOfn4+17/2D+2e+evPElLz0gS+OTUi9UhZ
BeC2R7yJH2B+xEzaIuJ4sX0A6GFKifkUoPzJGRluwFCQJ27kLNOMLIjZpkjXEPln+T7GTOQzfHZG
52lnRFkBRm5+eIlgLVB83adZ1c22TjQFTUzPOYHcIIAHxhdddPX1GRicnXMjadv/uyJcef0I+8p
F7GaqrJRxlXpkCEqh5AmUosrEa7rhdHEADh8HD46JzhYgXKE+5n/e5aeDcV1bWf+n617764tdc95
q/++/+7e/uh3/7xnoegEubZx5o99LF9TMrSnDwYuf1G1KHgdabKteC0Tf1Gr2oMn8HJOeLywOh04
RyDbWGgQRT4iVi7ROGQNRh3EsluvS1srPblcmZKWMojsiuy01hrAhp8X3XedCDhw8RAFo+uSqNkU
RGGxOytbFN8j4QXxj26uqqTIyNM28FovuFx56QO+58r3Q6HW4UFnKPJE0CWgzTUbaBAiqMN9IG+z
iJw++hBwXkG1PXYah4D2wO/Lts8F8/+r1HZ2+6+fof/+xnHvsrX9bZV4r4S3ycfuz+1288/8wVhw
4dkme+/YxkUC4s0jBekfpLQFDxcw6VhrYzzC5JDvRGqMPmAYVGZMoRJRBUr0s2dZfKv71D61fdLp
1+JNOLi0SCAQrhMTc3J51Wl+WcxaXDbGdbObMsUzOjLnccoUGCWgjia4x0Z2sbBAZZWVslIvypT3
9a7vjAe8iYCuqKbBzISfC3kBYIMr7MY+IY0eyO3yH0rfnaK9r58DeZL9xod9HAefa9Hz55/ORXrr
325XP7xrr/+Et7tNdOvj/KulPI3x7/5mlpb60SiS1yP3qx0DCYnThwsQCdBl0aLaiHRACTmsni4
Gfmg7ixMyiFJfclPWR3yLx7FEKoc0vLvBvz5w6KePj4zQ2kBWQ087NzbhwdtCZ8SoNuCgSWTu7Sq
+Lks1zzz2nYJE7jLW1NZmYGJP773/AGer7WGdFux68YJIUzHeRqzJfZWkopQFiI8B5wTgVRBrla9
j47sdJmlbG5iSQnNG/QwShrKz0Zf1edvrG177qv9k31v3H9/yx/syXf2J39dRLm+Nj8h0Xen57dV
e2V1ck39smoaHgVPMK3pXDjP3YC/S7srba2pRBe8d52g7RX04nX7pc5Mp3SO3Q1dL3t3Zyeoph6s
b6JoEjhKswVBjm2NgYObv4/cyM87Sdnvv3thxcWqSRfPfZ79C48Xt42omJCfnYA5+Ue++9V3NRT2
yAt0zzwutAZdQhVsMcoVe13BSbD91WCHOLnN4WxkuvS6+c8gvn3Kw1CU6hjtSYBQDWaH26vX933b
Wv/OTNN91w1X7Ouv/4nj3OHv/Or4zU4ua2Cxu/8dwZObE5kGXn+WYPHZWxxqS4mJMLHdaexH77Tg
H+MM+DQj/Ap7xPIbVofF6SS26QZOFyerdOq+1C2FEaAXJN5KUTU5M0OoSWi4vzNaiUYvCaJBnjzy
MjY/wdQmXmp+5vzm5uUmh7fHxS7r3vPnngox9176ceVQn5LpTF/FhsKB4JpmoiwtnMgCSVolEwSe
Ncm9zgNeIaX6dk/5hyp5BFReiNTQUGDq1k92m39uP+1296/fUPj42N/sTHHvzk3r6x7j/+wh7bqy
ff4QzqkjhnmrdTL67sStb/VxOnNqQiy5dl9rkgox0GpJORFLHOMccmktQF3RhcNyQCCWZqM/wOF
p6iWSHrpW4PkeFICISMe5MTDKNPnTrFhY6wc3tzi4Zs5Zft7bMyPT3JEPX0aRf2Liw7EUI3Nrdk6
kppR1CxiVyXx/8wB/Jo4982r2+O6TfxOHLkXrSei3hd/HRABQYaZjIp1FRSvshH0ckeIt14SQ0dG
jAEYBqJG6DUBnVpfmZoJARu00BkjTdQbs5Ukve6jaMU6+78bp30+/Tx586JMvesR4Pwz+S3gM+v
m/GpubbULyc7tfyIbLPTGU/LmVLdlcPyuD3W0KnEHqgTNrfG8qWPwQTSO8NOa87yWv18HR10t9bI
J9rljUCFXhSU+dOaldMULM0gMMcXZ6jsaML5RpAA7Boy4tLRCNXT5zhlrBMHCQHmCoe00f/P675V
Of+STD5MR3y5DTW2+o10QLn6jhUpANVMNBULXajKLMA90xjEqzoeT9GmrGudZfIEgGoj9CYkYEaG
ZwNr+4MKKeemo2+ddW8mUqtt3n+ZSLFH5nZ2fnqdtvu/nWfc+6//hzfexsb77ZGcAl03MHZO3ZWH
Z6BZFEMaXpBLR1fWVNZheWnTcbYddL0jou0LQqUvS31pB8/qUiM0fJbBpxxrLnjHvEeU+EmTA+1E
NBD+y6kLflcr7Z2XmLCrQqFmQGeM2d3V0Kes/Pz9Igar7BuGfoq6v4eVyWT6/IH/7hHfLNbz/FPJ
ozcUB8gIZxTvyX59RMfAgrGhYjdzUjo7YTyfwdHW6V5hVCf51UxIh8Z/dc5Of2uL8HpREZQIahW1
7cLXXvUSeqXHcRtpJCMFbWeeYrnZd94I233/x1d54/fP9HP35q31j3H//5Fzxp/Ea9XmuIINMrEq
K8oOLVkh0WdnNZWd+RJReyx1M7EjtDrYNaCJAUPa4LV7uw9ypJI0Vd8UDNdHJymoaxsrJCI4Ax7u
4AeOrLwYNL9KDL7ncgQ8BotzZ3nXHXCR4BODq7vsqelnZbO2kWFxfJUrr/gY/JU9/+1hToY6McA8
+IEgs4yWjXKYKNPENyJ9R4YajIOeE58Ry+s+wUqZfFcGd63rRg2QleFAwm1G1rdUWRMeQZ70PD9w
qOaMBv4D0LfIrbFOqWL0dsnnf0uyn6WvcLvKnt7z5ti+7XPct93zk/t5+GLz/+DM9Wq3d6wdp76

UjzkhcYiatnW0XSqpXGnEGu9PJZGVjj/klumXS1rZ209RnRF76NkmOXSV9gg3bzBvhQeENEX6CWQ
SACOWYra0d55Ujll8Quq6trcgCOMncogalcHJK89jNjQ165YmJKfasoldlwYWeQIu//OWvyhNPPM
HRkz0wpdxSAbDEKescvt6k4cOb29hImmukHUOs+yLMhdi4aIkGBt/r6zxZhN3YcBAWjzRGeU51Nh
voaMq689ZqqL4xgRPwPLMLlgmRN3CMsZO51+ILYFlzdGyq2Ry51YX9z/zk3/zxX/37f+9vR/vGuv
/4T364tfVv00Hu1tKYtNodGZmaYY42MT7KULDlnNap7V05u7Ejuy7P7OzuSb/XkntMec6RaXokGB
m0jnDr4BnhCZGTjk9PMW/dPLtF4GhqaoIGCAQYlEJ4zc2dXdZcsTkA7QXgg1wW6g4jrLnOMRw+ef
KkPPTgw7Lb2mHoC08Jw+R8V+gIg/jQTYu0wbJ+2qyPa07pjA/vjQ0Ez+M7+3ElpWHTECNffwF8FX
6cJIEkoMEQc4sUgKJnpZGqgBt0i5XhlEnHeX3k6FDN2Ntp8fhxbiunl4+6cP7nz57deOq/+lv/5Y
tidMd+GPw9ejhDm3X52EVxTdX/evCKS8dkY01D0NW1swwbztuFrG62ZNYZ5Eg8K/Vjb5B89pjUcq
1LIicFiITFi+ZueFiIaLdaAI72XNh7gF4QCxYGC8+z47wlqiYLBxZoJJsb2867TjCEPnn8BI0ZIE
jx48fpJT/x8U/K6eWTVJggdCkogL2UzegAebSh3CO+CG2LIhgaa8LMZxN6d0OMLTwxCIbNflbnnU
kxd08GD8/nbcbFKQLeN9HmVw6V5vt6ehPeg7XchMOehZKO7byrITQ6izASBALj3V6z1epcudGoff
AHf+BNX3fxY7W988N9/+Jl9Y91//H8+nJG8262lRYAo8AJYcHlz3BlBj54QISEU81tuU5dc4Mc/r
GflXxsQVrOm+xs7Epe22PISIQXNVIXziJZq1E8e4eTyJeWDnKe6rrLQYH2YvHC02AzALC05zwrhk
3NzE6RbHD65CkSJWC0CKkPHDggjz7yRfnCFz9PxQkYLowNIzjQWctWNI/Fgu4a/F7pkIn2xIuCSO
n3tjBEM1Z4cfOGyE/r9YhGCNULMqZm51TNEaqNMDqUc2D1+I53doYee4CKPWNKc2J6e0xwR24fa/
1WCnXHaa7oZig1JU13/V/T6/Wf+Gs/+vYvj4yN/OD73/fv2xfaGtrvZ/0ePJwXiLrtzjNuYV5Wbz
Zkc23Zudo9WX/6a7J58jvyhSe+Iu+/+08w+lh+8//+gBy65OXsmhmfGGUeeerESRLwL7vsMi7Gne
1NiqhRA2lji+EwvCi6YNDOBoNGSWRvtylZ81PBuKFICokVdNLAQBvO+M+sLNOIDxw6KCefOy2/8Z
u/5cLwVRoMDBvSMFkWsem9YaMkBz0NaaMich5NsdDAJZaoMKmutyedbp+bCZ5vNMZcyO2Mtalln5
Ont8jM1LSM1sYki1OPCoQ6DnHsXptAlZ+rVqPLL1tEjmPU2ZzSeIZXxQ/z7XZPvabiPf4+Oja2
4zu2NmavJ/eNfvvyfbz1n3H+Hx8Yfuv8Yt7AUS7QuImHXcgnNuYXxaDlxVx11VX0rocvvkyuee
2tsuRC2aRWyDNPP0WE9+hFF8tFF13CnlLkqPbsyF/Xz26yxAK+LlBdLGWEyAyPnaEsLs2TWLC8fF
rGJydkdEQJ+sBTgRgvr6zRUJFTLp9ak/ff+QH2z7LXJHUP6ghZpEaZaI8XXwGPC09X80YrZWVoU
FdNY+Yg+J3bIJPcxIeyMBKmlpjbejx49ymJ6eY56aR1onx6GUKLOW5NrHXkiKQMPBZFI1LVRQuyl
RxAog3w3Lk1AMlY8SRHpcUcUCWke/2utni+vrGzz13/NS3f+on//rf3jfw/Qcf17321RdPjE/f67
b5qXTQ4dyXbn8gK+tbko3NSDRzgOMpXnLpMZleXGAoCBkV5I9o/D5y5Jh6OZczguAAr7mlu+eMap
ekAXg2eMdRlHlqMT0tCRAZU9La6xDxNSVCGAdlR52HhUfrd9o0NgBRj3/pCXnqyW9wgXPcBvi7VH
TI6TFhtPSabtGPjSiVED9RzoGHNqXCbkPZgp4bXzAQgE0oL2EjwbHhuMBBxr/BH+bchQOWRD0gzH
U/w6PmWRxKochVAAQVXoiNGwg9robbMFTlK2tIjGYCC51htAid0fWDY9ja3r7s+edPvOutb7rtmY
48vm4/Z/or+nj5y668zq2y33fGdfHVV189hUX0za88KU888Tl57JGHnYdbI8jz9//bfyYvf+nfly
9++Wuysr0nuy6XBXKBwstedyATk2NchNtbu2yHQ354YGLBife7W84w51UidKvFGilnozoPCmOjav
5um4v10OHDnFh+4sRx9Z6Sq+DZ+obc8+G7XKib8z0R/sIgVEDJvVXsc1CEkT1sHC607LtjQ9qamy
gavGwjYn6q3m4QSi8GVCGHxkYAnWEjRgB8CqCRKIkCfwtDpTcstNUu9SAWjp2G6gzY6s04P0Tf6J
XVLh4XHntv30t7BLugTwWes4XEeH2322261OFKtWe+dOstNz09MTH6w/fe9+CpfWN9kT+uv/66qN
Vq/fQg7f4vSR4dyqNi6hXXvYb54/bmWXnyqa/Id595ynnJnCUV1A0/ds8HZfQnJ+VHfvpn5SMfeo
+Mj40x1MXk8alppRH2vVIgctN6I+HEcXhePDczOyM7uxuS9TI5fPgoPQnC3unpeU4wh3FMTc7I/M
Ks7O60XD66xvwWdERwgLGu3/ve93IBg+TP1jQsfBiNW/C5W9EwDkVwm9rqBoEzF45iQJZpJ+Hv67
VmmHsDY4Pi4Qja5pwhBqd4fXWLS3MOHl1hkJw7oitgYrKZKGWRjyTnBhvv9INOG+4Z73/5AGwfoaX
MJ+k5leKzDtjBRlyb9MSploIBKw7G6a8OmCE5wd965X4QcN897UwpC9b7xuhuv//Lk5NgPPPCxhz
v7YfCL7PHA61+dvOK/ePk/2d7aeqrXbb+ryIorB3k2NeHywptedxNf86UvPsJ6YL05RrAFdUcSG7
od+eKnPy7z114mf+tnf4GeDvnk5PQEEVrIfgJsYq5Xj0lwgPeDl0NIjA4ZoLFLhw678LpDLwLDxe
LfwNa+VGgAw9PckFBLxd/0Wh2GxB+++yP8HHg3bAo5SznF6J6Bp7Pm8Z4L42G4+JmIbF3RXmws8I
as7NagChppw/8dQn+E9r1BVw4tLfK5OG7QMxK1BYoMRJegmbalppjk4/YL5rP5MAaE12rDu3rUqj
axkTLAcdaIwL1XX+VkfHiin+X5Mkx2mwAGfoGbDfp1OshdupLfurm+9fyNN7z2nd//ptuTfWN9MR
jpda+qvfpV1/z63k7rObfgfz1Nele6S9rEbr2weIIE+KuvermcOnmcngWe0eqF+BnGMT4yzoWzvH
JazhCpshkci6nbltBwbm6BwtpQuQfX9+jRixja4r3PnD5JTzZM7O01pFuSI0EfCdyC+kGopnNfC64
iCTozK2soqfZgAp4999OPyrW9+U+feIPSttKxlvkxCpQeXC2fIPV2+iVwSr4dhaS4IODGMnF0VAe
8wNEXtF0a/vLpG0gb05eDBgzW01SitlXkqfFSWaoU5r0eTeeg/jGeiLAhYQ+2KzIg3EmvuZrx6TnoS
qKrA97nnIHQ6bdhqahtAvj3ebRT/Pw+4HpeUHAqDVGFgdp/nOddu/bt958y3+3HwZfqKDRta9s9N
PB77nd+za3cBedF2ticdTjOnmtE5PTXDSTziBe8pIr5PFHHuFCSp05yxizc0uytb1B4AUKho0m+k
k3GEaeWT7lQuFJmZ6dInl/d29Tei5cm3He1lBgvAfKNBhngcW/urYpkxMT9DZnnTFOOWOGR93ZRt
mkJUsLSwRVEBJjcBTEzO6/70H5+MMP0dsh30V91l6OnF9npG4TgHEiV4VxW6gLEgr+pkcDcQGAld
NYbDBCpDjhcWy6sB/GC9YRar6IAuAlNS9VJUMz1BrE1Dz5nyExOMO+NEWP635PUAkhcb0RCBg1L8
LGzNjXZRFRaK7rji9G7NsgCwpEicwbPY4ZXnu0Oeo2nxrDZLQIEgHpe3wNBOHwGnecl7mvf/3KV7
7yZ5uN2j987AuPf2rfWC8MIx1znuMOT+Be7W74UajOU6SMO3wszdFxF3JO09sg57v66qvp5RCKAQ
Vtb+3IqFvMzYUFabV3GQq3ujsu5J2huuDmlprLQWdJKgDLCB5k0RkkHiA+bIFL7N4HKOqxI4dlY2

tTnn/+eYJKMKrTp0/L/OIcvTYMs+Y8E8TO9jpd6bT3fGP5lnzta0/KZx/5nA99u14mVMNekPNRmx
xplk05w8CuQEYwjahEAtEeHhUbFkLtVqdNfWCovEBgtUS9RUB0VSQtCuErqytpQV6wAkU1ft1mzL
oUwGbPqjfWgJDi4chHE5h+g94djCs8n6e+Yd+F2ohiwEnGBjKG2beYe9toaqsduojabR9iRrohFD
iuHifwRb6byG0MTWewV/Z6tQeuuuqqr4+ONv/G17701Wf2jfx8DHen3c26x3mPq91iWgRwgvAqL7
TdCyHg5NSM9wZpYNdccsklmjvBmBHSuecmnTGuLq/J/MwBWxeGgxaxllvot1x1RJji14WzA5ZhYI
DI/1wYFrwWapKd7q4ziI5bXNpzSrTXedntrRYNM3eL8/Sp59kWh+MEPREGDk+7vrYhT3/r2/LFLz
whqytnAnAVRZr70VzR0VOvc3EjngQ3jDOAtO+M1ueohqbC0DVHFDKSUB2BsaKeiw2jwd7VUR47Dd
8LjTP8lSgIqsHLKnCkWk9gSoHNZVkaCkg1jBUpsiC0hs2AIbH7L8sSHAT0eDFQWJsbwaMi0RsIIBu
eD59Dq107bOi8ojkLpB+eoIXeJXBdpLxxfonXmpvv817jvT7zsZVd9udlovO1LX/5qa99Yzwsjfe
XBbDC4qzfIL3cLdxEghZUaIGqG/lIQcMbCA1NksQW86jLR3cwb8YZEAXmcnRSlvNYpmYmpb2HFj
ZxhgpAadaFum5RjU5IMxrhOAqUW+A5Wc8cneD7biGs7ftkwMFZlxOOMAdEKxsW56EDB1l/3d3Zkh
n3czYoZnv9fODAIhc/yPlYmMdPnJLHvvCozlMVYTXvqKiv1kvJGqpZCFnQ+yIpQTtdoPgPA4ABYS
FDGhUGuDDjzmWgVwNbiSE0qJR4v6QMf7Ex1D0x33ScOHZae3MYTcOdO/JHk4/p9z2TCSJrmDeLz+
cs2j4nu2e5gUxK/K81G+zk6fYyz7pKSJ2088WxF54ppeCaoltRkemc20odN6ewm/jn4il133Lf2+v
l3r3n51R91G+bf/eznHv0Ll0rdB5j+lMeNN1x75FWvfPkjnU73W+lu90bnQRbZp5nUedMt15pxng
5CYyzss6aXhoWakYwwtCkyjCakhgUEXk+C7pYlabvF0U8zKutj7mmnDxKAkth397pEKMFOQR8n81
v8e2FxVhYXDsiuMloQ//daPROxvObZzXVpd3ZIym9t73BDOYReVhjo8eMcjQHQ6e577uHvSi9Xoq
jwonqOg6GwU1vUan4CQEFxcTCUzIgKJelSoXBkbMwboII2NPxCecX6OZEXW4sJVunGFsm5tFfyj0
WfQx5LIw8ibYSUVMcJOSbwdXralO122EDxWnhPSK0SSPIgmtZ/1UNj84VRmrZxof5en6dl5oHPjH
PM85jUS5gNXtPvdRa7vcFPn93Y+tZ11776F37h5/9ptG+s36PHzTfdcNTlpY/s7e19w91UZ6TpVM
+5POzu2HEPjZJQ/aBZMI/qTd5sHRmRBQKatYAhb8TcUws34X2npyakM9DdGojp2sopGR2fy06K/B
ahL4EdD3LASFHkiZnDpRoSM7fL5ejRwwx3Uc6BkU3ifc5uEaianZuUDfczyjRtC7POUHfld377d+
Xs+oqWZ7go+9wccG4Q56ZhpWV+aGwgXczqLUEVhFHSYL3BFOJDW7eakB8C1DLPZSUVGBxz3khC7j
pgi50PwYvMe08VAY98J0/d85HtmODwYm/Xqkih3hHpbNhu2BARTaDrBqAarp95TZYsY0kEeQnk2Ge
hu5I1Wv/IwICwmG0v88eQ8VzEBcn7HZ/S7vct299q//NGHPv7Vf/Azf+/AvrH+RRrPLTceueH66x
7Z3W092W53b3Q3bMp2YeYwLo/DLk3CgLuRE2OTRD0NbFFurBfepvqgLiDUNudmp0n3Ix3PhWWTmG
3a63CiOIz3zO1l6vbWaiPOo2JICsTPAQkffOGxUQUW4GFJTOA80xGS4QtPIQSQA08Ow4SXHZ9oMi
8GeWFiZlqWT56SD991t3zr6W+ShKDKginlQ3HslAuNmlyY2CQqSq00CUPSndbypZEajSAYc9bXVj
joJ7kNAqExNqhRtlxWHJcDwaHDYyIsq/T5p7KiG4ZC7lx3WCJ7IzLJJRrzLgL77cgy2qGis9VlY
mYhli4kDf14zuMfph7Q0tzzXM1Pxf9wu9IRcw4nR0RgZWOCp8xZ/ZTXhq6bcwcWM2WvH7Tba7XfP
6xL37qR3/kh+r7xvrn/HjTm26au/6GV3+y0+59o91u39jt9aaiXHdNC5fYxpV6QjrZMANlPW3Eg3
gPpZ6gCIsbHvXbZgQcqi8Mj417Xb5lOAQyjdTkyq2vbbvkZPHv+u86A7J9zAetLNNTo3TU8BIC7
/QsKCAupIr3FciO9T8dzd3ZbfTdZ72KH+3urou086bwkhOfPc5Of7s8/Lxjz8kaa8LzJMLGYvdBb
EyMqKbgIJAykaix0sk5N4KwJSelnlu7jtuEvGEib6cXj7Dv0epxsJcy+GDARSKNsMYYaD9IvMdPJ
GXgamzZKVSL/UgE2N/n/hSkG2/wvVC+AtkHohwZ9D3gFPmQaw0vD7z4T3vaX+gqheewcTni9i/zg
AwbySo71ZYU/Z+dkx6a7Sp3t2fK5999ruPvv2HfvDPPST+KwkWve1tb5zY2t79k263d7XbcRcHXW
+YUDHIFeq3Dg+UNah2HyW+j1PvIPK/pYXFQbKA5IktTFxWeOTjz52gMU+MTmj2zqY06yAdoCulyb
awlVMrMj7akKy95wwJqhCdIPNJsu+fM3H3d8+326jTDlZyNmMmylF7/QYpgthAjrhcFoa8tdciS6
jd7dHLwaO8+73vZakIQmMpAJaBTnBrNrQ8k/v8FTVMhuv+PE2YulEf4aYjftobyzn1ctod/g2Ko0
md4m9YU/XGxfesqeEhIkHeH3uWECj1wSsacR9c4KbWUw20s5+tM6es9apxsfdVLGTxkZL9tOPHCX
hFShQ6FfozBABwKxMxbnPO8hvBNEQL9RqPk3/mN2yytDwgp1Pw1IHCgyNaxpAwJBguc3rNiRMnPu
h+9RP7nvXP+Piht7+lefMtN9yztrH5nVarfatbO4sD5pMqIK0ymikRRrtwmm9od+bCPYy0KDQfBKl
rfWAtilzaYKZIYb4PhPfvss86YFDhCt8vExLQUUDFwoTXyz5dcidinDzrHRcfff6jgqSCWlvLmRVhQ
WAJ5BYQVmk32tL24XN4AOvrq/JkSOHKCW67gwTxnpgfk62Nlschozyyfv+8P2y7PJiLLhuvxUAMO
on+XCUBP2aL6EYlVDri964BvS0ZdjPlRrC1DMrp/m7pcWDNHCEpamvpRr6a5RBu07MVQuPyuZR8N
5kJkV5aFq3nDl4F9/dY5taj55UPzt25GSLAWV6PzTSUC+Idj0Ca4XKp0a+1FRozYgGaU30FuoWvt
QU27EQbk5CvozXIVrBd/weoFa703n7DTe89tZ9Y/1PfPzYj78jeuPtt/yq8zTPuAX/jl57sKiIbe
p35cJ51EI9TlqEBZJ6owPDhYbIEHWUuzDKENoClmcnh93w8uYXnNWC7hqEhFBWmOpGzV4MJC4Axr
TloiNHSR1MatBumvZSJherZJ4zZoAlv0u024gH9rrd9z3lnqcQpFLTB3HMAOeCHJF7nKvpyPYD9
z/Mao+4BxQE2UoL3EYWWEE3IzEIgd8twc9dyIeOS7zVaK6zhW11f4PvPzi6yvmiF56mgvFMJAGK
2tcPCqaEzgeEiQ7wslqFoyEkomBi5ppCGhE8kYVBnr2AqS2XWv/h3+n8QqW5pHeZkvr3qvjJXBfx
O8Ug+rYS7/WiOmuBQv52aDzcAjyKzBFbUwmxbn5jaK5u5u6w/e8fa3JfvG+h/5ePObb3mHW8Bptf
e2fz4bpEdTD5DQc/X7oYaW5joxXBIJoWfPReIAvOBmwWDHnNdCyMecSfpszjbPgZtnYMRXv/YVbg
iQBSUnhrfD5410jTkvuefCYJe7jo+yYwRGjxwSTBp4SRg2cllsCnudbedRW/R0YCPBMI3WuLgwwx
IOAa25Of786U99Vu784Ad9HVUFzvKiVHag0dVjT/XLA53QckIsRII8US0gp2YsXNTuc5fPrja6AB
lDNzEJmwDKWZZj5mkWcsPQOQMhhdJNrrIwocRDr5cGQMtyQzM+3De90F6fGBKnud8QMh0irSykRE
sxnodsZ/F+enliU77g+3v0t4iRGsTBY2JXLTYyJftn18FQ7vK7b4wHHJXlflD04LLTp0//2r6x/v

88fviHv//i229/3eOtdveDu629K10o2RwgR4p0MaYeJSRimKJ2GAUBME4u8yRzM0CrK1LusqnN1+
FGoQbpQyYbzCQeSTx98gznnS4uzQpUDcemJqXn3h/Nz6PjE5RrabqFLYV2nkTaY0nmErQ0EQZv72
2w0RtoKwnyhd46GAlAIYTLMOD5OZerOoMhnfATn/gEARdsSGiVQyjXROjrPR01TjD8KSo9KuuduX
bZYGfic9KmbW9ool00CBdxbOD+QvepXhsNwmcgVjCFYJnDh76e7ECAShI/X1Y1hW38hoW6KEHBWx
rDKXhmv3GUaKw3FKK2qagckzfyJAobLX7mJpXwqNyp4J5mlFMlSpxryMxz9+ EzJ7HHXi5VkBtVn4
q0nhz7GjJrx7ECY5EpiwFS2eaCddTu9H7q9tture8b6//L461vfcM7Nzd2/003M0B/IqYSk/0S86
ZnbJnirYbKex7wzJWIwITldPA4Q2yk5sgQR5aTWiohr+2yJRBSsEzxlFPf0vcstIsFYBa8339wHn
dhbp7h49jYVGiqxs2Gh4RH3dxYZVmm3+2ozOeoAj1jE+PScBsGhkWR0bS4JIsHjrbXdw11WU6fWp
bnn/sO82qEv3CTjWZT64U+peQt15rKufBE827zrJrnDXw04YkKdFVMnCq3uS5H3WdSdnRUwR5sBC
Tq1yJP0UvO2fRKeZgAxnlvzzwalMJM29qYP3rjNxDK1V5xXAibtUfVPGOpzRR74bTYN71H/ndJom
Wm2G/KFh7zc2LlyomyLsLGDSHYw+Hcb5ISyjisDRdq9FIMwsYeFeX5unM9un5287f3jfwCcxzve8e
a522+7+fHdndbP9fuDqQEL98NIItk5keoCAenFDbdd30ALeolIaXXwZOPU9Y2G4HrcXNUDKqeIRV
ElPdGbC9QYRrOoiNWuuOe+2WHDx01JTE8YkpGg28N4wRpRfUVkcJonEKNG94QwHdcseRbknnV
fWcYoJPTTa8PCZ8Nqfvcuabtcl8LZWqU8gU2o6ZFU5m/GZPJyKn1RsuZC7dg9Z94XXh86w/OzC9
Lw18Vea8ZS/bdtdghfDWiCoQFtlWMoF7blnXyP3I/iQJRiaouEfoTemSLfRalbJHsaFnVcCxuEUh
tliCGFXNUUGOE9TeTNFCjsb/lheRRCaNxeOy97jbK7LayOyVlmJiRZSIXchvml513/s3PXC65084
Y3XBu5hf7G717rX4y5FX3w4NFx1MHQxO0uymSvny/Ck+YqwgPoKuQyU2XXGEqYAgzwtDcarZHI3S
UFz7Q22mTeVX3YDQ/cWb8w7d8hZ4098MBW0JbPf/7zcsP1tyhBwBlIa2tb5haW5PChi9lvunjwIM
kKlAt1HjVNMEpwxhnkeFCIGIcEk2lvT/NehooM8xo0cDSSo/H83e9+t2xsrAeApl6zJu2BIRWNET
GlsD7wc/3iqoJMMZBiX+vFA80DY86jo+WNC4i/U+DGWFow6CqQQ1JjHnNazJHhVRDUKrbjnepn+PU
C1xJBaboYJhdHM4JH7RjbdPfYzghB2xlKqQlRGTAbD95usecQ4qtIbY268WtaBV4zoiQeZhsOhX4
8Sj2RXG91tMyvoTXn9sHb8ZiMeVU7TfHFza+dn3BO//VfCs95yy7XRdde97FdX188+s7F+9k+mZu
ZuXFg4eM2g27vM5U6XuQXrvgaLOXRkCw1RsBPnReR3t74ycoqcZRTkqFwYXtNWb2JBr4080hqZq4
Vw+1790XgSqZcLRiJf2lA+7beffsrlq3Ua5uTsJatRn5NrXvUqOfaSl8js4gEZmVatYISmOuhYc9
FxaC85I0WbmQ4kTnQSm3tdolnj+RjwBd24Ycflme/83QAzsBKsgWK8wHryYynGppWG75Lj5H44y
+BNtAaVYxtJoSuiUfKzTA05PfoayZDZYZh2qAvDw0qnTiQwB8PbYwG9vHnQtLEWk7xsblHeAvvic
tatwxFCBYxiT804y5rPlyGwxYC69+Uz4HdZEwoOcdTa5Tl2U3c5HK/2cUWeTXd+vzHfyXC4Jtuev
WBvb32V3dbez+/t7N92cGDh5vsBW21pNUBQygjyJJVyi6W3zD0Ah/U1+5YiPehDkszqQ/LaqrqgN
w0y0p9n+pNOXeBm7Eqpc2rAdquberw7l499vnH+drJ8RnOtmkNonL55dfI0sGjMuXyVoS6oBaynz
Tvs5QD8EcrlIR9lZwG3mwSDTbhMmNM4XWHXUj9hS98gV4N64MC3D40rPm5MkbX05dVd07GY6WSEA
66xQdDxWehkaAa7prhV9lFcSKha8UMylhL9LKIQvLIi7PloU2R3OMiDqFvIP/HWiLBRlijf/X1VP
HIciVNsfsfwTuoGhrUbPKpoSyEex/Fobab+/CXDQMobfFLN29YzjXk12vk3i8ajsKAUTBw9wCku3
YHX/2qV4y/qI315puvu7LT6T2+1969xnnQ5uzsIj0Gam0wQiXNZ6oMUGnBANUP+RkU73lDRAn2Vk
c0pbxUyhtMRbxaIyzoquxIqK/5f5eopC9nRIoa8nef5TFCbw05UOSaaAyHxRw7dgnRYRgchMzwPg
sHDjKcjT2pAItW2Tz6ngh98TqEnlicVkbC+Z4+fVLe9a53yZkzJyphZhyYOQh/oUNs4maWlxpvdY
hCqBiG1VuwAYBiCama0Enj/7bba5cbWVHmqGB+VxVUD6Svmikf3J+ayXpwwfJYvq4WaYSANDHKK0
OY3QaTFENGHELVIg5lFNtw9PniBfcxqpDzbRNHiYvhea7IPDxuNuh7MoW+T02UTFH12ASgojSUhh
TwGgTPip/deUHB6R+/aI31DW947Vi/072v0+0eRc0RHmjMt2CpsWiIZ2wZgkV+50a+2ScXNOeF7g
7SACaR9ZJpfyJRRW98I80xt08rL00VSLBFXQp0ZaGQH3k0WXD7RZdjMXJFIffffz/pa+AIH3E5Kj
tp3O4+MzvPRT06PsJIYengIRotpfnarT6HVqFkg9fjmaB41V4v59889tgX/fuXIWWWK7OKM1Ch6J
CVSK/VLy0crXoJM1Ao/roVzhk/dImnZ+aoRhiMJo+GjJ1lFQ/4WL4cV/SALQJgrhlFQWXQrqkZHT
pj8D52jxURLoE80AsZLkc1zVd9dGAePOAGIKp14RJTKIKR4r4U3oCDMHIsPbRWeKwJLQJGzCbHz
IdzsWWOd8aqABUJWzWTLiCdFB4mm6d/vSL11jb7e5D7V73MjRoY5edcQs41PaI6/IX3mA8n/rcSN
kkuacLeug/KvOnzLeokYVipOxCMt3GiT03LzGjDQuIjJo4UOOGmDO596xxEVDiD33oQ0Rz5+YOyd
LRywgIoYEceSRKMKr4V2P4C4/abKixkG0EL1tvBpldG/iEQ9nc3JI73n8nmU4kaGSDwKRJ6koKYD
5X03CVYwjqSfKhCbsYyvfMmOG9QSVee9vU1AyNIYS8tSgYAY0+10jHNrIkyH9KyDHJAa6UbWisaZ
kz270N6G8R+RqtDlmjmr8Hw/fBVAyN5VQdr6GRjhdEGWg2uMKgoxJyKktH7W6tIKPxtYU19zlg
HRVYjYeURsYynz/SLKfU22ETp53GP2huuvS150xnrg6659h7s4ryZjvtMhKUASC5mk3ExV2tPWLd
ABsfPZQlQpEUUMG4mqAVRBAWru5JrzIKCyRktZa0yGSjPVckDZeeHzSo8ActQv8houSFON71Pl7l
d+5VCIDB06ctgZ6TiNEIR1GLGGjVFQo7ednaBSXVvjSfaIS45ut9OXz372M3Li5HfDqANZnZqipo
aKiW2sH8ZKzTOjqVL4qnNk2FPqFhpUCXudLumSFDIB5F4su0wJlEeNDWA4byy5xTW9vt54w2CpuM
5UBeg0w9MCqQsa+4tgsDZXsJvDDh23t08EspK8PTmqav3zDZYgl+e5C+e76sEmEHoV418uS3GZs
ExARZF9MsSkEjo302szp5Hw17buMZ87SCwqdzvprn99OCLzljzNptXLi9tIgQhEWB8mjVT8XqySm
zIeKFw/atc0tyK+8Z0qXvZjiINO2Z17AJ6STmuIfKUM18PjeMyjKveeJu3wsVRKPJHipkoAj1cLt
B6LX7/wAMPyJv/zB/WDp3Egxc1bVFDvgRPZvleaL524TwaAOH5vSqDhcArJ1fkW3ffhQgag8q8VO
S9rLHWtauF51SpKcNY7N9WI7QFz03OWRaID1BQbLqNpVoSqfJ9EdZmeZ/lG6gJVj2X1jS98cb10F
JoZz66Jz7wc3H8fU1NkjBIWUKRMPKSB12plyYxja2MaDxqDEaae0+8X5nW6HLPgUL7lCUTz1zyYF

VcK3tmC2WiRawL1/xaioY6fyzKkCIL3t4UQ6IiHyZQ6PFPua9ffFEZ6+tvuv4fdfv9S7hgKS2pXs
UYJODaYjfUQnstlCqsJxIXvyorQvqgD3VpxOg/jSTA/VSPbzbCuIhq3mPT0bRNzgriFSP0xIfDJe
w/tOnkeuNTP0jppae+Eco/qKsOWF8dpabt1PQsm85R5mm5HB3K+SgtjbtwufD83ciHZnjfO+/6I8
xroReCprBlmIALZKUR25iqi0vZOdrmZmE/v9cVmMPfPv/8d2V8dIw5snoVj5pXWEkcUuXC9ixVgA
XpBzYaPEcv6gM+C9vluYK4WablJFVuKCqpRGrkpRBCW14tkdVNff2WgnWp5qqRYheMeCI fCntJiS
KTAGpxaLpRSOt+A/KblKH3Csp5+iCkSr0njnxEZp1YVeeQ5fnwPY/85sD7PwhKIu71t7xojPX733
TrvFtc/6u7WE0k+kAiVRg6DjxTekWfs+V+EreGm6myWxBKJX6SmRQVKF/Y4ch6q/cqlPIo0G+pYx
n0huWeYiZDobCGmekQgBG8rUmc2Feo8ensUmteP3bsIuZ/MDiUn6zli6MJ3euBrsIg0dsKsj5KNc
bRRR4b+c/79reely9+9F7NsdxrEWdY+Duk+OANq5onktqH0k8+GOqEwcaIPH55ZYULH+EvJ8D53N
vCRer4oq0wIM5aKlIpz2Ko/GOBXOHZOpSRsLnWG7HKp6CsU6vUe0OombG8GVIO7/0YHRX6d2QKZR
KohmWdOA/AISmBNSunqcGixs65rsxBtaHeelUpI+spigYiJb6mHPJxH2X085KqKP6e21fk3yuKCr
+urCFCx18Uxvobv/GrkfMwD7uTOkoCuhe5Aq+VVLBYvSl0dvqDsteRuYbvWQy9pdT7iaGP4zsmIh
q1TWIwPR8suphzRhtB+b3KeDGjNQPVxvJiyIBNnQ830Yw2Ng9duXF462PHjnlpzMQrMyjfGHNjIO
9CKdNYwzOT0IyU5+Y/q+AG9r+/8//QNjogtrlyVGEI5OY6T4m/pVHU60PsHTMipBHwyPCKJUnBE/
R3t2UOdMLGSCA/IBQMORo8nNSHcnfdEOpkPIWmcYmClzavFXmONFQH+beehQQJUd5LjyBz0xwFhx
sQZVFFhexCIMcMhZFY4RvE+yWdyjjAhbYfgr0W0hwf9tjdNS9aVNKYoiG7b0LJDhGX/6sgHicGaK
U+SLOBPAJUREATN256/Y21C95YH3zoE7/nrsM1DBkHKjoGAXsZGfd9g9EQ8lptTA4XPPck7agWcl
JQ7qrME441dB6WpHESHE9AYF5Wq/khwXHit6tlyCrSaARYM4BqCHau5zXk9IrLr+Tnow1LF3NCIA
dDh3e2tn0nTUKewAKGiYJK/j0RCXz0/o/JU08+SUK9Cpk5L9oYpfeG19McOKUnDuSHC1NjNxcNa6
PY56CivGSQH8anZ2TC13Uth7V81cLg3HvyLLCAsNBzJV8YH9ir7AfAjvIo+VCTWu9fRMV8eE/SFI
uUobFFS+UGUQKL1ssaVe69dT5ZylvkaY0hVKXaWhRkfrqBWC00DrV3y3PDRmwsKkiwSjhkye3cxc
+rDU0RkYTKAJvUo/CemMI+fUEb6y/90j+7ttPq/BSJDiivVICBkHtl6iGzgYSZMZyqcFfNNXfxfo
Rh2YjPnzR3UGTUdnTzpFaqsTCsRPNisZkpVQ2hsFkUmrswzPVyIKETx4y0KPziy5mLQuuXuVSkqv
Hwfpsb69owMDlBAj9MCXmsDhIujxOPs+ub8r73vS9oQIWM7rTngaUkPffVUEoqSPbAq+xbfZTlF/
c5Z5ZXCWhNT0wOMZiMimkbox3XwCv1J7w3fSViiAJNMDzb2HQwc01l5Cac5aG0UxQlh5geKNOYJG
EfmXZOJDedJFEmU+jaqTCdJC+nxPlrEYf1YXSW965cXCWdUfPpvCgq2sV63ULdG0djQH0ahm6ec0
klADRDGiBAJ0Qubjfo2B+5QVrrL/zO++sP/rZz93pLmkToR3VAUXVBLW3skY6mAqY1fyks4i6u7
bLhZqh6O6d5VqY7gHc8WEySzxuUVsOFesytPPjMajUHqvyosqKiYdYS4UPxXKVOGsASDUM5uL02r
RoVofOMNhXCFpXpWpAaAPQATJqZmaIXbeJ3knKEBTedrFwIDz30kKyULTMEhndE2afwYRU+GwYtQ
2GW+Hkl1lPjeoKNLl/L6+tsly7PzcnI/VGMB6R7Ya5qHkZ1TDUKzQ8jTwImGeVOTUeoSUqy3psXZ
lmHhwsqY4KMgU0nfdQSQqpR9pt81EgL61QISp/uRIYw++dY4lmzUGBNPKdtj1o1WCi065JvWOB
8aERKIMD5XtmYDqahGhOhFXb9O4Av3oeQow7O6F05HL1hjvetD93yqM8guM31X242xoJoNBWEQFq
uAmWeaJLqQIyn7HG3XmrteOokrTae3V3rrKA6gh8mYUFmvWhoIBpcHgKDqYRlKGj0vyl+ABFeRZY
TWR44cIQML52mKgghlwVqCx0foGzwAhkXlFw1v8+cIFtedd97JcBWzWDAbZuCBKzNIRgyVsLy6kZ
VIsErUqL1FbCKHd1+cX/KMJ11oVvIJ5Qq3cvtk/cShRY1ADze+F3YnmYbyIO1op1Def0FqWGOgpR
XByxeecywV9hh7ZXEaRRyE3DSXzk09GE3y2IDsnpsMC4P8SOvf5BYD1GskgVvMhnHPPLPPwgZer+
y55I150KISbVjpxm8MWsoqgTxTg9RyWy0Ate51r78gjFutb77tf+r1uq/GBSaa664Nxx1kmeYzaG
wOIU8cBQie5RfPcNHfQPWzmvQsfX5uYbO+TYXbKX/plQWMg1qlwRWmJlDEQ0aoi0BDTuoAZTKk+y
Pew2uLlG5CBw4cCKWQyEuEaCP4GMsmIEmwzJP1BDAXhhdzU7lTD70R3fLqdmn1PuD8FB4FQPk1N
Y7WouCqkHpfTTKMLDLaslad07JLYZiPwyKRhqXWkNWLzXPxRpmZGFnQqAo9/eGsqGeZUSerrunIG
5E0td3YR3V18fFpHOGO38ism+9YJuXdVXF1YKGSaYSasRSUkxLQTVciyz8vXXxmCQqjdorO1jdOq
IkhIGBaQ55b4qPI/WeJefYk11kuDTkFpDX6ypCV08o8URFlata6tK44Iz1p37qx6LBIP1HWVE0bR
ciLzMvieZYyNpHmFXCi6aSrKwLQRiemkZt4OemSlWrx0HX1cgDVue0NjDbgbVnsQhk/qEOm6Jef0
NXj+fCMgyKy06bOCy8YdHvK664QtUm41I5r1Ef8/mkIoZ6/JEz6jlo5ild0j2PuTfvec97ArvING
MFlxrB8K0TxhhKxhaifKgHhxieuXQAnmh1bY3HsTA/z4VIj5+UQtyJ12giz5dyKEV4jh4/0mNO6l
qnNn4uzpHKjLiHY6O+BpqXm5lv1E+ScgnqZ2aezVXT9oswoqMIm4DVRoGo15N62fKHROxxH9GVn
KTJQpAmgQ/KAH1jXNT4ddeVza8czi0L9vVy/w9qkYFvg4MJJ6ls6gIUYt1HVUBCyosppwPWF34ht
vWKFVR1HkieUnlqzdhGik79kpmFUpX2esYi3I1la+ZE+UjCcKHVRkvh5RICuL1lJoBwo/ERj7koU
Yq54R35HsPcyrBGJF0s7R+TL1CleesX3adhVce+M4WNoY2gA6Lcwm9AtmCYN9I73vk82NtdL4T
APcFCUu+8FyXxYhw3PrpdtTudygbFJ7e5tsUwDBBrXwRQ14sqwKssdk0I7U7I88ilwWvj4CBdr3i
AiXw4zDSL11DGbEYoAs0aBgGKobxlSii8DMXPVrCTKQ9tCVRKVTe/1RMGtP+U9QtmJ9M9apXNHm/
HpDbOS3KI6TgWJE/TEELXKN1hDkS8pSdioNHQuSGelBwdjTqIh9lIUJedOaL/wjPWeex5I3cls0j
6gs9vPg7xIn4X6JAhnVbmrWmyOwoIsYfg8hH14P+UM971rluOSN2qfQZQ1aMTmoevCjLSkGeYq11
Gpv1YsOHgUZltReassl8ZNBbf58OHD7PgZJbjk56p4aiRzSN5c5ylHpiRpuq941IZ96uSy3PORjw
S1f+Sp11fjXq9szVIVPgvrDOFzdOZRiIAXGDQEOBWhoIJ321TG0K+uSkwdHVRCKkGvpTj8zqlDM
aURkmzTskgK/RaQ7pmlMQM/7c+F9aFG/P+auiah5y/mv8bgwm9pEE+VPwmJPEQH5fGFXvRtGrHi/
hNOq9281TI/OoTQ0koDuGqD9Zxzf2GkviNw+b7BAVEkw7ytdsoKfuAq61TXhl2dWECTLH895hea7

o5ZMcgTGXY1gh1q7wyvsBApCEASJKgxq71FvGLQskReSi5RAokidLT6l4ErRpgGWhgtcChdrkirp
RvsrBAjQ9chMkoUmnTSimMBgWidNALukNUwa/Vw0ITSUMjdeyJ53jPP/mTP6aiPkkenlppA5vs56
qRGYuJixdhcE3zbqQE9vyJE89TywnGqp9fv1JHhQfNvNHnllGubJ6opmURiUuOsPGz+bkut8YsH2
hPwUjU+Hw7otY6+KUPSi10Sdk8HEXWrxSSB9aS7YuGHheWesRZUD0k/zgtynwW9y9TqqdNcKfB+u
82PgMitpV+SnDMR08VNpNJsAa0Px/uyirJJ2lolysjrLzSTldcmMY60qx/0p3QKV6EKOYNNiOpas
laUb2ksW03f5sximeZSGihS3whX0LoSxqi9xwI/0JIW/Hg54a/WkwvhoS5Avul0oXCdijb8UVCbq
wTySdLAEY0NBybbFaG+RbsymGZg+WITNZWz8of//Ef89h6qbZ4NYKhZmGMhAE4VkeFt6PH8BPSrP
aIzWltfZ1GMT87x0iCqoqV0LacSVppso4bOkIEJTRRwC33rDIYB8PsQe4jHaY4Mj0545H5iJxfZS
RpaYMMqHNRdyx+9hgnobtKNXyzkCbpta4OdE6DBGnh6YvmWVPfh2o19iwb1BtsEQ13Uhn1Ck21Co
/++qZzL9OSSV11CO9hTQEENZOhur91ZNmGbU0Y56ZXF4yxuovj1krRwCapdTDNKcxYrQhOz5HEYW
ZnSXWLwrrwWKUzXp+yoSSrF+9C6lXpwsZKhXObcx7DI9HA4bvWzaluYlWmqryPN0C2fSy65uOK9lT
FEFXYX66DuWhUcs3odEeAPfUhaezuhc8N4v1bwD721RRzYUyq1GfmZNPQGGL+7vSdbZ9epMUWE2u
sAczZMpZHB5HDSuvWx8CO9B8jziIpmqvJobWLIzcfRjmq3jS5l4IsJ4rWJV6TF+FwnHPYVhEMNF
PEf0+GkEoDeRYP9dJat0/kkVrxZAJlCXuU26cnZRMlpzxn66hBqBuF9p3Qe2z1JK4eb9ws22GSAZ
xHXoJk3Ihi5ZcXRRby15JgY8oRGecMVYQM6heksbY7nX/qjG6R8ipxLZRvqqGpkQ74eyygLi7JX
A/vffLPVhQ9ciB5G7lgAjex0LbWlCen25vNR8KbKVKV825xPhqTlKVsjSu6NEjF6vRjFAG0PAMC0
UbuSs0Rh+KY5bn3Xffrc0KHi2HMfchgyEUmMLlJOWXfarYkMA7rioX4nhPnDopU86j6siORkXhcK
AgEl6XKVB1XTOIRBRA095fGrdnYVnzuB3jgAO09ihVqu0zCgRFRYwftJGzUnY0YpqShOaKnLXiNJ
RqzDufq9ZhPb85S1juehZqkIXPeckFz/Ny0lvkJ8j5GTg1Ke9ZNceMLOY2HjX5595AcX08/lHorV
Rh8JoPs6MS5EIva856f6OysfP4BxeUsd76htdFr73+VW900/yfg9VhpQDL14aUBTHTxQTMPApsCC
5rrrm+pmwWzg07xDRyonojeFg8gF9CrCvIVZrOT1yEsCurNDHnXgLtmo8t/z13toqFPWWXjjY5X3
rppQxT4UmNOMfdHXouacVrxPS2mKkKr9rzOkAGAMXeaMPQJ3iNQucSkuEVmrz9qMa+ns+pUyfiEJ
qemtI6ZKgpKHoK41ZxORLSyY8yLY9hM9CmhyS0ChqtECLqKNWakYV/g6GLZaKaYtSJcrAtHwXuU
UeWvaL9kuliKwMk403bDvaI8iYR4MHptSK9/KRIk5MCSKWZXY7G0k1SclAKkqViaE8s1LOK3zrm1
QI+8ZqKnXkwTC3kCoP+Bydrigg1BeMsd52+80Pdnrd424xfMTtuFNxVklQ1ZDNAABc8EasOVfkCd
OaG6VeDaEfPJ+WHPLQ1qWdMnzCGajHxm2DRx34sYLaxZ8NXVDzcvEQ8psbkZVeuQhesxhSzjsXO8
j96IkjR46xWwazbCwvDz23GBs50NGKnOTmfr+ysir33XefL/so9U9FvdIALlVJDxRXkyKEpZyJ5N
XicX4AfGBMaHuDsHi9VvkqQYHo3o5I8HWrls0RuamCIoUnPmDhc/CxkQhITOGz/AW4ZOJueeTXuy
c/WA3YZHeY3rAGnSv3QLIhgjynpQcU3jOmirgMVc0QRBVBCp9Xkl1lav2Jz0OjqDICwNviZbeik
iCXIttGCX67DGLRNeAAYmq2ZyECosWFP08nExe0MgxXHGI1Lwxj/Tt/52/elut1b8aQKJfPNEFuyC
Qaah6veb2g1OehOp3N9IVUAKTDEJ3uZaWbEDYXUZijwrICB+dqQzplRHyiz9EM1VKFb6EaCoPzqa
QTfA4EKZihzpsK3fHc8Fk8CLR4YMkr36sSInR+ScEDjrbKmiCS2ZeURFxx+vNfa0WjAi4+hnqg7c1
6CSn5MBQgaPP+ssmkxTNVa6/L6iswvLrCHVo2iLNPoMQ9Cw7h+lq95Wy7IPLyJPaZGmMfnZYOMQ6
KRr+I9gTIzEgjdK56vG8lQp40ahtZtLSeVMhD10ioSyBNMgXzUE1fb/nAsRpgpdF6QAYG2KvwjFs
t3q22Qeh8HYdp5lTSh7W9JUIWwI4y9dE9UkWKVEBphlVJgNGRuf1bv+j031pPPH3+nM7omZVDIti
98HstW4KfwLVCes5kHnY1KjczfiDgyT1cvb4hr2ST2YxRqAZQCM4nQvh+HAI+bxFJpq8o8saYMo+
MKcKcqFBLqmNWDn1ATfd3WygR4r+mZSZmcmB2qCdqEM5teluVxoKohbP3whz9MT1j1lmaKjRlFfDY
mZn6IMFugdic+Hh9k6x08+L+PNEZkYmlQedFxFVZHS67Mux1LzQ4211xRG0CBRYhAwvY2YKAXJAO
ANXNTQIYhDckWtNks/zKVsdTcntVTMN5pk2ZCRhWtYlmzK7qGiMmNIPBcax2R0P0QtWZo04RlUsh
CdXsBIBRtwosaoEZgpe1qtW0Ljg5H7yzEZ8XCdNi5C+mYpkK7POJhXdVK6P5f+eW+sP/bXfuTV/b
S4WJN9L+UILxkNC5KBqokQKEeCnvnpbH1VH6nwzyuQwvAnV+ZMt5MGnunCbFvwBrbQU8TlOMHqTS
9R3/J4zhX6thJDNR5AVFC95bQJTI/P09Di3w4GfshSGzxK8rGAwtB7733X11ZPTPkPU1ehnXUen
lcia97hs6YuBIiu+sDkGrQ68oilP9dHhmUHga5RzTz0JNKBQV4PG349BuZoe/DfGk9rp776uqkOn
QBmb5xbsqH1nnjq4wVvWXqG5FZlKjYWhEPgXwmsEYjTyQwnQx4DaaQWDeWhufWf0yyCfAMr2SZFr
kfQiahUCBKNlbaGwpRI3MmaaVuXg7yChuMiasFj1yqY1alX83DmubVeW+sm9sb74bYMYdFeQkV9q
4GsbPIy2WoLAu0lqyzRQ2yFC8D68U4u3gei2WPodieGkWWByU7k28xbzBa94yYotR8DdIleS1Naq
G2hsefJ9XHoahAvMwLpVwiDma2w0ZrXOTLAzh+hMA1L4saxiqKLqzcpQV33PEBffz2dt9jK5lsiq
La+ZBWsc2+YTO65wa39rZwzeXgoWMEd8LmY3ROUSG3JDSL93Ukqt92at9G1Xx8uOsIqS76aDnw2e
WqeLCWHHI+CXlxVJ2v47tf6mxqL0W6DUXkKuS9b+JLIbKJEXKiK2sSJ70RvOM8Ll1FUTmIywgrif
iIw9fBI8+ttrA2OIG41EN16uOVLuzW1hrbKVIQsAzrvqqFCQMLUaIoc12/Nur15vntWX/wB95ykQ
vvLlYtooic3apIlchY2JVIXuDEsSLQ4DKPgpyizJorZGUNa3NTvc9yRF3o+jfKxCkV6XlziI1+SyI
d5vCqQVf00vXroYdRcuXJwXDd6GVFBfAYEv/S3y8dOERwaXxkVAXeqC4Y+5YpkU67RwOE6NlnHv
20nDp9vPTQFZlLTlaL6y8QG4emsNUeTSYTrz9z5gy9+vJYmObFRUkeSTNveUjzPGBnuW8z44AKT3
5AWGggjklrs1wVGwKGNPc8q4oocHNMa6t+7mrqgbQ41El0AFRjZpchaq0eiP7ihdsthAw19cxrK6
WVWrcn68Nzx1E2VKpCKmX6XCqUV4bbRzi5mlTa7WKqGRYhzC+G0OHcR0Cws2pKNVybjkPZUUNrw1
/w3YDIPy0SO6+MtdXZ+4C7gFMFu0gS38EvoTex8PzKIEEZ18PNrIahyueNwtgDvB5hGiniPuwkl9
S3OGmeop0cGOrUt0K5DY6Khpe7vrMe7bWZm3nJSWVolKvhfmhRQRNjr8VkwkBMNjifqrqftaCMWK

KDpdIA3v+OO+6Ufi/17Jw+vWmjloR+UtvUkEPmYsoZaTmGwjdxr6ysMCSdnZ7RRSL6utgrG2Jh6w
jGmE004Vwk8zS/ghum5fs+rq+EuFnQyLISctQaKRXraYJ935xu7KPY59a8dnEUhizHceHrropZWP
g4LFSnFENGy1kp0q6YRkTuN9MEv2EO0UclC+MvrNUxDXlkMbSx5p6GGPmm9aLyczAZa+wwPFOMm4
Y2SgRpl7g2xGQq5WlprE+ft8b6A29906ILaS/Pcwn5YDlGz7yBhG4VBY+igNaGkx6SGSmlO4CSUu
TaQ/9W2pCo1IRT2EyXTOF/CGdz3oqUw3jDrU2jIRqjeqzSWCQepiUWck7+6huzLWw/dOhIgPfDgv
DdIKWxx/Lww5+Up59+mh7JaQjNWp0SKDbFG+ca21jESidKWZNOSfXr9NrMU+ve63I+eIVIDtIBjt
wlELwpbOJGiBrCRyW513wIrOfpSVIAMo/36XXb7C0mljA6wWiGERI6YsIkrKhUELumLxgqDtgmb
8HNcKsGO51tVTASjiZGa7EYYq9z0J1Xfl+W1YDBMwPadDFsnp6pf5ZnVRXnEtbtJT6ShQb0PIiCZ0
NlPuNV55Xpd/Z7G5/pP6/nzuf+89ZY2930Pe4CLCK0SSsnoaFHLchHVj0cRcsa9SEV/cJ3Qxh5Ye
C1eMXrxdqOGhQOooIgCHdvXxPDDW42at4DRuxlLHzdjVzXvGz1stDT6mbokRWP9GmEUNIareRTzX
910WXUXTIgA58JtDeowBfaa4vj/9jHHMCEUCLJmGtOkDHdt4gFuZasjEQQsbj3W1l1blkMHDpc8a+
RhnEnlCRSPGGkgdamIpnmdYr1GLkA+NnLXyveWrKn6gWDACUqxdJGx5kiZv3thspIn7FJBRX0Duv
e+uifFYbRFdR2Exo28klfGJUX0XF2sIdlnOtwsTHdPPEMoOofzHVQRc4/GV4cyGzmnQrzgRlCPy+
FaHtcoFuryAWu8oQtSuing16c5I+cl8b61re8cdQtzstNsiXzHTXGkTX0tfs6WUA+xZA9IL15FN
rjkNoQJmNtgKbyYokn++dRaELH+2g3yMB7WN+i5ZzP5OU7cpt4nZe9shb+BQVSzj9JATHk4bjdVB
3JEA/t0ECQZwHEYECE2ZEyk8bGR0KopRrGEP9+Sp588knN8zyohMVhvakG2qBORJzfeqjtGdUPRn
Di9ClZmJtnCIy5OeblygWvnS+UZpHIS7RgTpC1sSG39bItUgQPFgTR8V5+grzem35lyHSTPajm+b
T2raJnkSfW49+sDdeioaFYqiLh9Y0y8Z08lWPOFUUvfDOD5MM9xkR+Sa6Iw7BmQ4S5xnx4W+UXE5
vIPEXURwyRDbvyeTVxERmOonBfiXsUeaUUV6Y31s9bFNk5HVzcmHujoyPL56WxOq/60/CquCZEO/
3gJBgd29QqDJWQePu8ohaVQ6ViLwFp4RkMlTlk3+9ogzT0blp/J8kRqCt6aL3hpTmVKSXAqxipNz
rXixnVjeULtmIlwyPqw/j6imZTJT2G4fpcJBMMQkRYyCB7YNj29vtSFyvyT3orClUd4qhrXV9FC
VlMOzycTyUTkCGFKe4uXWW4f7s9FzZKZQbY6ePWCBOvTBvy9UrWIOEje5IvUETYYZPSTM/MjJWD4
2Ixxv0009jNM1UHW4UZqFIOWjYhOimGm82ViVaEYoZhFWH6XNbXULJkNVmPZNfioZalktmkLcJGD7
nxxHWGjaU61alzmVeeNITZojpjSXEjztQ7s4TGNKkfrot4R6HXC+BDdXlBWE3ihVIREzMzM53zzl
jff/JbbE3cit5ligXJzc6+o74Ek8fQuu0B5FIrshdcpoq0Uyh224U8GSA2KzBOoKwJlWXUOSEfbow
oaP4wdRgvlQA11lbhgRfzYS2HaRTZ9JVOE4G6ap5VWOZ8LSRZKikVetv1NTky7P+5zqjk3CJ8fWa
8qjLa125LPfeYzXPwWiB+nGndlIvEj65g+FtRL9TQqkfPsb29SdUHY8i0+wMaQZaX4Jwee0YVfI
JsCa6L11ULAnK4vql6X48BKDFA2xxt+M8NgY3IruLlHlyJch5dH/ai77ah7XWJbGRZmHqeqUHY3h
XDUWRswrlnHenG7fEMU6uIPhNFRzgs1PjGcs7iyZRoUc0/y1AlH6qVF8UgzNo18j527CLPA5nGvG
5uFYnQJytDTQE2sMwIE8BUGrV6/+Jjf+fnbnB2er2fcVd3kb2bmXFOy/AqowxL37M7ywlqRh20uq
F11NiOm3lpt3qGLK8gyL5zpbLDM0x0fwpCAOVhoggDwyG2Ua6yKxypIEUg9ds80WC4WRbGBA4Jzt
sksaHsCrCFNDc/4469wfKKDdm14b02W5bN5a2WotZWziq0Bc7YPlU9pbxQcCZo5rr3OX78OZlFWG
LpxMS180EWQsbcz+QxUArnWPM3Aj12/hzhrWIEnn4f5rUasyr1g8KqhlpxJJ+aTlo4vWcGolmuN
6ss2daxonjsrCYuSA2EhNUCyUYAl9+FEih9EJCyzv+OiKmSreMXtdB4F/bZm0loKqCptVls9wAxt
pQM3nqlSIIElW6frStMA8oc9AtRqQWFWJ6X9aFhZo1rsPo6PjgN3/r3xTnnbFy0nOWN4m49XtKBf
NAkIEL2i0y0HDYX5Bzux/iKHsBxG6oo4UxNOJwg6tqCwXlOEJeldTC0GUJFzsOeSqL6IWG4mmk4R
XD4KIY8TrOQ7uTNboTzSxeWD+DZ028qr7mTsNqE8hh77vvfpIL8HHoBjLvpeSHwdDMGG4eJkfiz2
ltbY2h9uTEmc5if05QyI8qotbYEDPfQ6qRjlu8IJ+IV4FIyiYK26RM55eq+dRjKkLZpniBbKsNps
4q0+zqDJ+tOd3AuuFhWWV5ru+nwhmbiR6RLXmZNxxdWIzpxBDeTwc0YoqOjswrOaOe8RV5lcaSq8
czKRdqdTUNF8mH0pnqNMFAcIgbXvU/1k3Eh3WhGaAiSIBgB2uv2WzunXddN294wy0Xux07eFCU3Q
+hiuU7makRIJTNlR9sOkRD6uaRqvsZ8dxCDLSFqbJAWc+0KXG4fn2O6it3NrvIlPukmHQ21F0TiB
G5D3URAprmji+dmAZtaDh/gTxpddK2LrrJ6alQgCeBo8gDMR8G8vAnHpKNjbNhhku/KPtWjYxv+a
uVpGpeigZG0+m0ZLe1Iwc8S4rhZVoE4I1jGE3flzKdyLtU5zeJmT8lFJXX2o/jKic8VaIKdx1Y+8
XokYpYXESNow6tYzElW2LfSsdCPR8SDNPxIfe5Df15FDSJFRgqQipknopcXKDBHrXXEXE654jkGk
/IsL9RYE2PIegIJ4mXbK2pPrBvVRSCgGUtVefwZOXcVh+1WbpmzCTq3CVxuIYm5ar6wzV6VwLrD
23TT39Z7Wp2l+UsboLdoc74SmyfAqrWUog1GOREfTh0il3LyqFBONSmJ6TusnWEV8yGUgqVZXDih
DxjaAQNJfsvIRJnXoAhwLZvmRiIt+6cCvUMFEPFSUVzmgcleFnJEMlhCqBw4wSYI8BVrl2jxxhzl
HlRft77ruXdl1TniDripzcGkPCeq3ux1s0Qh3aPcMcXl5WQ4fPBB2cxAqPGUSebFqO1lpoysXU/
AMnlGz9/N13xdQ5fZ30PhiUjc4wI8PiLztnJKgm6KJNFIOoJZUOqDjk6ipJg95X9dAwNpuxZx051F
bCdcf5pv0APHbeIHEatVipgiiHN6yklWpkhPvTnNbCnhr2E9GkTe6qJ6qWogfhe0esM9XiWoX/T
Jq8LX4KBNJBabp31MBw9f3LT+1EDhx6xdGcmfjvOqO23R/67wy1jfd/kYUto7lRVmvtEVGkSy/yw
LMYC+ldcSAYeMnYZf8Tm2MzpNKXgBjySSiZzMD9kN3oRtrU9IisKOyInCEsZMy3GQRXmUwcFHpq
GBQibsbGp2WVoJhyqIoge07EqGOiOpgRE9Kwvm7rPbnV2GqilntJNTY/LVr3yNxmbnaxMDcC2I/i
a6U5O5U4tsYKmWjtxxLZ9ZlsnJSZcHTVY2Lj+9nOEK9ec94prQ8AdQlIj8uEV6WIirqdA596JEhm
bBxBVKJ7GctB9aFYc7F7TlKf7nvnUwpQSLd1Fih4AJFetX4Jn/Z6PFnwph17el1R47TFxzKucvY
E2jsdld1JntddkkOpGk3iwb288Yg73yggsyDBpJ4hxU3JHVcjGVUwSnyPrp8+yCokDlPkQsJ3Xk
1VD03q5biSghUMCW71QjLhngzvUZrZq0JY+65VPDR55Wx9tPe2wdFvlhkurvYBVPjaLoF09PdDe

```

ERRKt9iEtQJSstgABHLAGYUINPqfOgwFMF3PFhUy0pgpejHGmiuyJC7Toc3AAk7HiYEG6epyjV+q
KK0ZIGgIXlgSMTBDeeggplVXtGYzVK9+e43kE+BgqFOCI+8Zq1O5H7wwYfIGdbP0WNBd1ApfdLgaE
voGVE90DOeEIIm0dlsEeg4ePOw3oYFolaEfhhmzrOLV5hmqIdfG5av5Tpe4pANaCF0lDGiEI77Uln
MzURplhZkppR6RDdqiTAzvTx7IF+UG4OfnRv6vc/WGpq4fRaVEDgNr686BoBvEBuojkvVVwL2aw2
EJNDgN0G3gmXo7elE/8gJCdNKzv1lKK4aSSoldMgj/R3wHTzokPqAAUaFT04Oomu+z5gUAsUfHu1
gaViegqMAGWjEbTSpt7o6ONrvnlbG60POfuzNpApHTRdCh3g/xmaLkXhrjROti/XABrNivzKN6yF
0093ULd9D3gEEqRWC8GPURCQhnCEkRdhZ+fqsvmrxm8GY766Ih8TPAm3Qz2EZeJZU1XMaQvehie
daC0z9pDa8fmxyIldYWNZx/23t7MqnP/2pgHoavVAL6FlQxFfFBxxbSk1hAz2WV8/IkaMXKTvHky
gI1ORam4y9CLXkfnFGeRAL87PxPE0y8vlapJPoQq+ons/GxoYP8WohnLaiv0hlVihJflbGXSI0xS
aUBsQ49rVtExIf8AgyyULdWHxVgDk8ucMSNLcQtImq6BdKMFHQ0eROPcWwaLiw0yiYWRilUeQVTW
OshV/zj8MozUFFJUKCF2XK4sdhAGwLM29qcWjER0qVlJueSgklEDuuiOUqPR/e59ZDD32i+LPalf
8jwAAZs7UsKZQagwAAAABJRU5ErkJggg==',
    height: 150, width: 180, top: 62, left: 50}}]
"></e-cell>

        <e-cell [index]="2" value="Gender"></e-cell>
        <e-cell [index]="3" value=": Male"></e-cell>
    </e-cells>
</e-row>
<e-row [index]="4">
    <e-cells>
        <e-cell [index]="2" value="Contact
Preference"></e-cell>
        <e-cell [index]="3" value=": Email"></e-cell>
    </e-cells>
</e-row>
<e-row [index]="5">
    <e-cells>
        <e-cell [index]="2" value="Email"></e-cell>
        <e-cell [index]="3" value=":
mark@gmail.com"></e-cell>
    </e-cells>
</e-row>
<e-row [index]="6">
    <e-cells>
        <e-cell [index]="2" value="Date of Birth"></e-
cell>
        <e-cell [index]="3" value=": Jan 3, 1985"></e-
cell>
    </e-cells>
</e-row>
<e-row [index]="7">
    <e-cells>
        <e-cell [index]="2" value="Department"></e-cell>
        <e-cell [index]="3" value=": IT"></e-cell>
    </e-cells>
</e-row>
<e-row [index]="8" [height]="40" >
    <e-cells>
        <e-cell [index]="2" value="IsActive" [style]="{
verticalAlign: 'top' }"></e-cell>
        <e-cell [index]="3" value=": True" [style]="{
verticalAlign: 'top' }"></e-cell>
    </e-cells>
</e-row>
<e-row [index]="10" [height]="30" >
    <e-cells>
        <e-cell [index]="1" value=" Mary"></e-cell>

```



```

        </e-cells>
    </e-row>
    <e-row [index]="11" [height]="40" >
        <e-cells>
            <e-cell [index]="2" value="Id" [style]="{
verticalAlign: 'bottom' }"></e-cell>
            <e-cell [index]="3" value=": 1002" [style]="{
verticalAlign: 'bottom' }"></e-cell>
        </e-cells>
    </e-row>
    <e-row [index]="12">
        <e-cells>
            <e-cell [index]="1"
[image]="[{src:'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAOsAAADSCAYAAA
CrfzewAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBjbWFnZVJlYWR5ccllPAAADDF0RVh0QUxUVGFnAE
J1c2luZXNzIGNvbmNlcHQgd2l0aCB5b3VuZyBidXNpbmVzcyBnaXJsIOHbh/oAAAA6aVRYdERlc2
NyaXB0aW9uAAAAAABCdXNpbmVzcyBjb25jZXB0IHdpdGggeW91bmVzYnVzaW5lc3MgZ2lybCCprq
v/AADyDULeQVR42uy9CbbseV4W+Dv7kvty97e/qQ7qquqqprt6pWwFAQNxYVHEcUHEGZUhcEGdcT
BicNwGBQ2JAMNw0HFGHRxHR3AmJrBRERVZG16ofZXb7nv3ps398yznzPf9zt5X5VND9MNJVR35a
nIuvfdzDx5zsnsz/X/fb/t+RlVVstle/+07/vjvkcD3xPc88VxHbMsSyzTFMEQfn3LDd8LvpSpLKc
tCijyXoij097Iq6+ek1Jca+M3ADg3DFIufYTti6ucYeK3I1/wp362v+2ff+WeNsD0w2sN9Y+fous
mfre7AcP3QsB33UxxKvf/P9LlPsBV45Hzb9u745W3GFqyv//YX/puvFBcA9V8DVse28bDEVGAZnx
qwrXNYv/rPfvf2y/gc2sztJXh9t2/52i+THMDK8kLSNMMj1TTLJce/8wLguwDir8Ia+b1/+Ru2X8
jn0GZvL8Hrs33zV3+BmJat1s3YWM8LK0r79+pmKSU2N3bx02eR220L1u32K97++Fc+IwaoJzFpbJ
yzX+xfwNyF/zUG7A//P/+HsVoupMwyUGdTMvx0bBeWH0dvVvKbf/c3bv2jrc/6OQTSr3gvPcYacO
onmrCuDCTBwsI/tWBpbfiqruOoD+t5LvXPPBvx9lY2E/kw/4yfdav/QT//xcd4z/+tj9m2I2ukZ
eladuWuZ5H1nRyZmN/FnxbyzBKC6sLXF2zSuOodH0vdxwPTxdg7FXh+mFuu045PLhcXnv86eqxpz
+wwWG2YP3s2v7EV75PLq7d6wxYEzsuP02wmgbReQFWgM8A6nz8tL3OrpGliQ2/OXAdr5FmSS9P0n
5epG340YFhOW5VZJ5RlR5e74C+VzjOHB8S5XmS4fNSYHmFz5o2291zz/Ompu8vzDyPWnsH6dW3va
84vP5YGYTN7Q20pcFvYKB+1Xs1/8L/NH0hleYjCC5y2/KCB+e/iAR/whv7l6DefKrc/B7g4W/2N9
w85+AYRthlgIN5i+16PTdoNePlwjTzwmoeOz2ny6u+EzZL29hLkmq31eyYludYEZz6VPe7hulbpl
FgsakKw7RL27QK27UKxwnWRZbOKqM8sQz7xAmap14YnhiVee/sF37y+PylD58NLz202rvxWBq0Bh
VXju3dsQXRG2L7Y7/96Y313Dinnz5gf8mbGICl14s3G6Wh068ugOrgcYDHU3g8aTtuC4/uejW75f
nBfhKtRjieKY6hCWPZcBx315AyDH3PhMmFpZTA9hoNvKZVVV4Xdng3jlaNcgHDCV4sZW6k6bla7o
tgWJalBX+zHLNwba+AlY3wWAG0sVUZM8dzT3q7hy/4zc7z8+MXP7Y4u/ly7+ihs53Lb4nwtwL72Y
J2S4N/7bb/9svervllzWE+AOwm0lvJZ0KJDVDi6lNTYl0BngQbftJyvHdjXwP8u0aXGA1sLM8nw
APPgA2w/E0yiJfY99NfHhio07btMwGcG7hmPDTiPG+LqzoQbRedZk+siyHSwrOxTaSaGekSSxJku
gjxl1BUOI4XQldTyZxETeyCs/zxfE9MasqxTGmllnOzUr08fm3mr3hh9s7uz/T3b304db+zbtH1x
9bma5XyLYAYgvWX+3tj37pk6+mY14fwH4iHxY4s55wXpd3+u3eM1WeH5VFFhpVkrVwH7HPUVhKz/
BV3aIoI4A1gC33jLL0+beyrHx8qg0Azg3TcG3LflsSpXJna2EpK+mEEjSajCXJYnwqebYCEBv6nB
634ch0eiKrZSR5GkuarjwlXmsXhiIa5vSCJo4uUxZuYUDa4XdwvL8pRf4U9OoXvYbzZ/t9Pd+bO
fKwz+z/+i77vb3Lq9xjcrthbQF66/K9ke+9KlX4fdfBrBdAPaZVrvzRe1W53HLMgamATTwi+lb4t
VcBnuXfRI4z72d6cqMpjEvAEQtRipTZN4ty6wyPssvjAAxHW0hHW2xbBMKfJUHNfHZzuyWpzJer
4Q12tI006J6zfExLGQsCZ4j+W6m4INgHmxkPl8rsdY5LGea7PVAWA9rAalmFhkDFjoclw/Xk4ks9
IiaIRRs9MdtDrdd5xuN1o93d/Z++NJj7/rQ4VufHjlmG2p8Ras/8W2b/wtnyflxur8Z/B7fQBrAL
A+wPAkAPvrhZs7TwWB/5Yiid7mN1st7KAJq9oM/eAEFmuJ3cGqZlaWRLt5lrtFWZhpLu2WWS5JFu
MTSgVkvRkSRwt8nitxvIR1XEoB6+o2OhLiYduOpFkk0XKmVtNXPamb3bpMEQsIGLAC17ADUGELII
wkjROcYhEZfzqZFrmyghfcEniM2WEEA0JdYFKbnZzKfnRZ+0FleuXb9uNVq/bzjOP/m4KFH/93D7/
3NLzYH+2tTtoDdgvV13v7kl79LcNhlhazOXlFAGGcsBcC+BYD9QKPZeQTfbb0VH/JENmV7ZVnarg
eglLk0m6H02r3nRyosL0tgCHKzyqt2imNaLKcKNm4uLgCcyFbqg7r4KF0ydCXEXXgV5aLcajtFF
802CisbSPswqo6ODAXPirAmKwUtPxcLBTi2D6Ad1+iZC2018TC4sp6DQCDJqdxLHcrhgAa2Ab0s
F70h1YaljmHKA9v38Pi0SctvuD2aVLV55rhM1/6/ruv3zsA7/1Zw8eeXqGs9/S4ilYf+Xbt3zN5w
v8wRqofZ9fH8AaG8A+DMA+47jezX63exX0dw/s8EpVmc3Van7IKiITr97ZPZThcPCybVunwOq7l4
spALNWMFQl6C2AR7BmWaqfzd81Co2PybMEVrQlZKOYVX2cJoPKWEiBlyfrmR5Tgf/gjTLyPK/ViL
Zpa2CJ52J7nsxgLfOi0oosPk/rzY+ZTkcyX8zEDzrSDFoCViCddijddhN+cUum4zOZnB8X8J+jw6
OrrwyH/R8Jgsa/OHzosR+7+Z7fNGp0d4vt3bYF63+x7Rt+89t/OYC9uL4Bfv1toJpPgB7uNnz/Ic
BjP0ris1EuTSpawPlYOrld8axKcIPP+r1OJytSmY9HoLQJqGahJRDT+RIWMQNQEz2SLE/EBYV1/R

```

A+Z67AZfGElKkGj8J2H5QVIMR7aYFb7aHSXu4viWcwth4AGWPfLvYZaeMBn4efKT72yX/HaSRZYe
r5a0oKq0CWXVJycoLncqXX9G97va60m77s7++LBQs/A2hn5/ej3nDn5Nr1h3+s2Wx+f6e/80MPvf
eLT/duPpVv76otWH+1AWtsAFt9HGCNDWAFwnt+G17aanc6R6HnPYp/90F5W/PF4nJh2BKtFqCtvg
RhS8Lak0GvLU3fA4AXaIFp8WgJ79w9kcl8JulWb9MOZ+AnQFRkAIupAMwBcIKVH+8BRPB9cRgAIA
DWHV4FumOcQ6HPwadUS9rqDSVaz6XKM1nFK0mjVNM4Ftv8YDk9gHa5XMpssYZ1JZuu2/BSfN7p+V
wi+MXM0/LawB3HiYZy48plabUa2O9S5ufHaavdHR0eHv5Et7fzT8NW+K9uPv0lJ5cff+/Wwn4G27
Yo4jPYTFgnTVAqYDdlEJvVrrixrV8zcvBAwBZ+U9wO8v8F3fRiVxk3LMHp4j1kUuR9FqyEsagJoW5
Vh2Y1mh1FVmWdrWKieLNaPjGdzseFPVqDkL7z4vEyma/HDhqxhaWm9CURGZU32ysJ6ZqDJhLUNRA
UNX4NatMaO01RgzkbHWJ1z9UUTI5EGKKuJfa2WEwmbTezHhf85kMSPQJcjIf2O4T9n8FOb3YF4oM
bn5+eSV6k4Bii0WUoDvmscrTS45fiBWvaz0UTms5XsDVTysHeIReKyOx3dHb4QJe/a3V3J0cFh8f
JP/9C/xvJydvT4+7eA/TQ361u/9Vu3V+HT3H7ge//OxxOTj/9lU9ckLfz/t4Aavr8RNgadRvgILF
ELz/1VXvRLy15bdrDO8qKRJ4lPH1TyutvFoWpDZcr52bFSUBP/Pj07lQR0M83remH6juv1SvtkU1
i0FazXZDLWnlkFd1Xq4sH64bLINDCUwGKa+NloAYZribKDEgBbRQv4wKDzFReiDPuHlcYZhM1Aug
BoDj83y5aSgwoHAotwZ1/WyzmOZw3wNsX1TLABX0Dp9RgYkOLfC6xhq9VKMlh5tbbdHavIM286nf
SqKm/6rjtN5MTXLG4s3dle3NtafDrv/2x3/bOTxZ0slgAgMcRgPlb8Nhttdu7zdc/ahi2j5eDzx
p7WZYP1+vFYZGXLsBoGY6v/meOF7QaTQlct0G6nJ1Ld/dQ0iSXk/uvaCoFUNbIrB80AL61Bn94DH
UPraW+ZwEQ2aCvgefLoNNWi2rC2jLwJEYurWZPWNfIHG9ZbkANWmwA1My7Akt6nka3LS7pMmgwfe
bZYgYKXahvGsC3Ho/HANzQr1LwezMsMHeOT+XkbCQVfOjAo8/rwvrm0u/0pRV6ekx5uk5NsU6Gg/
YHr924/g+7vcGPX3nymdXR2z5/eyNuafDrTEXMC3GN8uMpMe/6ffzldwEAYaPRGPqusxOnxcjz7M
tFmb91tY4OAcRGkeU1bQZFJbja3R21YHiP+oJmJ4TtjqZDlvNzBaPrteR8fKrUlBstVct8qdFbjQ
IzCJRKp7cnaTyXGL7uOQ6mA7/R9+h7NtXKLgC6MAxVbobpmgyfXxSM/gY4lrpqKYVfm8CSko4b+N
2xKtnr70iUxqDgkUQ4phZ8ax7vdHJeR6Gx751hTwiSjHfuHuN4SpmszuoySsMVVkrS2lpV5Zb5eu
98Ur7ffP7ZcXn9+tj62I9/BGBNt3fXlga/rtsp/pPvkU/MhY09/O+r+Zvv+7uOY/sAcel57qNJFL
01jvM+ANC4yLk24RsqpduUC7y3klFd/B2+Y7RaamSW0c8YFrSCVbQ9fxN5tvGw1F9uNLsAV6T1xB
7AZsJHnc9GAGILvm2LKVeluFwImKKhNeXBLGc2uijC0bQLBdOyeIH9F+L6vu6/yBP8LdWOBM9tqL
/K57hYpfESFjfUKLDr2njUfrUJ/zYIfZx/ILMppjQPrldlkhsvnhYUhlAWghfPDQmXFAYk1zA3i9W
plFMnteHxvuf/IO7fW9VPFTLaX4DO3rIzA/s3v/0n9nWADiPoAwW+nV+F63iGoboK/Aa/WU/PF7P
HlejUAziwCjcxwrBqajO7JClSXgPEVILZaVUZaYeJAwhU1VgMgMwAmGwgjxRUtCenhLWzBRZKmr
BuUjF9IvAxuwCkp5RXLsYseAZfMgZYF6xagvH3YJkH+9ckwd/Ox2daWsjzIMBYmRSC9nY7O9Lt7e
ixcbE40x1Jivi8OX53rVz2Di5LujwRE5/fagSaY93f35VG6Oqx9LsNuXzpCDdXLgdHV4XiE6enI8
ngc5+N6+opnE8wgk8ejtLqN5yft/v3gs/3/zwb/+3rcjNFqyvIlgtU2/uP/1V76mBC5QBsF9cJ3
AkSJN00et2b7q29b7VYvYW+KuF5YXRajXz+YrAb6lYGi3rHm7kdrsrjmtJfi2k2epJksxhcxWMT5d
PCDACPVhFGGv/OtDAizZNN0di0lNwnLFuRrtQX7Q96sFzsjINFLbM6pQKAM0DFet/5ZILXptLu7e
n5nJ+fAYzHmi+lz8p64tX8RIHugdi220NYy45Mxwx44bj8Jka7kaPLbwGdhr+9nkq5nomVL+XK9R
uy0+tisYkBYk86sNpSrGR3fwcgTmS5juHrTmWygF/tNyy46c3T0emTk8XyC0ej8Y3pyS1ne4dtwf
q6W9bNTwM/H8GjBatY4uHt7e48ZVTp+wGOHdNtL9brrLc4u9egj0jL1+52xGbUF8j2HBsUM5A5LJ
wJi7pctTWgVBq2WtiCZJdRWoDTBCD5ZeVFBDCD/4OPKNpHa4np1FaZhfx0c334kzn8T9Jo+qrsq
GFzmDRmM5Zwm9lPW+I58JWV/OkO9EIf69zuibelyWrOgDFdBDMtt8aqC8bw0LbOGZ+TqsZysH+ZS
wapqaL0sVYBv2uXIVVbcJPHmLhaDZ8jTC3sThpmkmPo5LpdIr9hG6eF8PFYv2e2Wz2ntnopPcz3/
dd23tyG2B6vSyr8dolzpNasSEBcm7u7u68t8jza6kVpiwfXC5OqXMkw+Gu9Pq7WltrghKGfkmKAI
wVQvPFuaR5KXGylNjcaMEDG1TyjZRLFK01aBMBCX6uDYAymzqWtvYGHwimJgjl1aAX9kVaDV8Q1r
SmzozoMuhU4vcIr3MdS5/nfllbrJHIypOT0Zkso6XsDAZiOQFAxYWhU183B6BD+KCMHGdxLgtjBg
D2lJiFXLohy+mkjmLjFIBCGLYt3jLHu5kKskHFM5xjifOJcMxN/FvEleon+K1Jcm26NJ4Zn00+4g
fHrIeMt3falrL+irf/8X//YQXsX/re/0DL2gdYdnaGw19/sL/zO6qyupFk+e7o+N7NOI4t3wvl6p
XrWuubrOawVmtxgFZSRM9hOYRoEIfbtKzJRgvShj5WlqhNikfiKeZWrlIdEpdNE8raRs1Fa6KBI
+CuThx4Ue30l1hVfBysdDXeR67YypN/ZCGF0Wlhfw2uTuOkYBkiWEU13I+mW3SOqKWvTJ9pb8RfO
gsjeDL0vImspidlnHx6Fw6nZb0NRqM9Quv8WEGWqELV4+O5GC3I038odPpaFXTejmV2XQs08WcdN
1ywlYb1vlx00On79+51f+Jf/Rt2/tyC9zf2fZxv/4LHwD2z/++D/gA68MH+7u/CZbqPVL1W7M/ns4
dnk3GPlrEdenLlyhVtJcvjpVo4lhKyRpe+JYM9k/GJrm/XALJQYQVOXJwstKuFlJ0legboZRLF2q
NKn7Iq6+J92xQtftvDrWVThAixr0Gh+TgngMmdL2kkgx/EKzwdi+a4CEJYmtBrWf1Yuy0ulyGYA19
5VQHwF33KGz9a2O3w+a47T9VwaraFY2P96ea5VU2zR47GbTlM7hZgi2t0Fixj04Yu3pRU44uJ4+5
227OFvgWuoK8A8MalwkqRa67xYzd20KI7OxpN3p2n58Pj0xN3ebVuw/rK37/yGL5U/8z3/Wn//K1
/3Ba7nem/b39v/cgDsnWleHM1m0G0XuCmt+TatWtycHAAwDQevJ+plX5/qBlZ3uCL6X34f+cShK
CLoMVZtFY/tMzqjp9sY1ELbYWrGwW0YZ2+Kah0o9XZFEPgNbTCoMeknHVvKkx2nkjQ9LWcSi221g
N7ulAkOE7Saz/saIqFBRWay7Vcpb4k+kmcawUSglUMUFXwjdkr69qhOH5HUzukyGYWmDSa43cbdB
rMwa4A2H2xyhjn5klDiyEsCUNbdvs9acHC7sOX7cPKRlhcuEAtVpFlWl5ntpw9Mltm37GK570f+0
f/8zYyvAXrZ779nT/5lfJN3/3/6e9//Q9/id/p9J4cDAZfWVbF+7I0vz6dnPdYJrizeyAPPfyolt

95jqtAaneH8FFDWKYzaTRbkqdLreltNdrS7+3BR22o37ee161rjPISCARfLT9a1pVJmxJCg5VHyZ
T6LRr8cewA+14qCMOgA0DNdT8FLJrv2uJYbIWRFpyu7Wi+NMBnP7DioLTLZW2RHb9+nlHnkoGodQ
S/tlLLyhTOOkqURjM6fdFtxMoq3kbRfAT/t133L5SRHF66CV/XUMreCh3ptppYLEzptluytwnLO2
zLpaNrGmzLsHhF0QpExT+YLJZvn57Prs7Pj7eR4Y+Pl2yLIj719g/+3O+Rr/u2/1N//1vf9Fv9Xn
/nba5lfAVcvcclaf7IOlr3tX8UgDo8uqw0MsbN32x1hWWIO9O70gYNvHTpGqyuV4tyVxmAY0kXlJ
YHYFgEUYL2aqURgM1qJDq0EcAcK6X/dgGqIltpdw6rlkqdp50KerKM8rJIooXjoLK+G2iLHMHu+d
RNqi2rBT+ZllkpLBYE+q5lmuvrqrZLR6LRNJQtQbpZAstiB72U/rBe24K8utbMn057ZDsBaSgg/Vy
uxcB4sklA1DCuQPBPJu7+nPrQWbWDhiJbzTQlJKb3uAD8zbaUrWfwRLUxcMytZryosdq84jvXC8v
azydGTn7+9Cbfr4E9v+71/8R/qz//1v/9dTnewew333XviuHzvfHz/3XEW+a3mjtbiHu4fKHjGs3
Peu6yPgK+5lnrc7rAvnTDUdIrZYAvNaWdBQC632vLE489oftgbnOxmMtqvpD7o/satWWEtgTwqD
hIXzRwWpsgUT01jlvlsdIplqAS87FpwrRKqCJnGfOnKnJagsiyaw9gNgFOylWLyU6ZKF4Ak7B+YA
El/FZaV2NFeh2JHe7AAi/1PBqNASjvTAsqxuN70u0MxTbWdf+r3lGaHJod+LkAftiVdDXSUKq228
GnV2DeuvlKzQCSlRZ0kJFMF7eVNURJ4vquu49zfrwRuv8hnp+TJmy7crZg/fs3f/zn/4DV6Q13AY
tnlov1V5Uij4S+7/d2Ds0cN3prMICFSmSEGzgM++pDugDQ50x1ndFaRCvxhwditWrqulqQJvPtB
ZXaWMMSgogsmPFx79bu33ZGbS0k2W2vCm3jo9lMvtK6nZruuv76lOm9FeLVP1S5l9Ju72gKYvxfQ
VEFDNA5aivmsSJRpwPKcp+V1YnZWnGXJQUMSPBPZme34NfvS9pOteyx/kik8VkJL0dnHq+VDre7O
xKvBiBqhvh0u/uglqa+FnoOiBzp6JX3XBENoK4Hw91pY8duIwp8zzvX9yJgVWNB9+93w2keFwKN
PZAoxkYQWDYXu1vP/oeNK+vthb3MLlJ7Z34JYGf1rb93/7NxpNVjKIvD9Pkq+Ab/XuqsgHYautKv
ad4b4kq2md5wQwKlDWVqevhQxsTXOD+ib1bEN9QUZVKA+SUv8oXamiYKFpm0rrfdUPBHAZifWxz8
AlZdh1rW9d6GBiP74Ha5izdthTqwWzqzQ5g6UOQHsplMb9NkBdsXOYplJb56j4EAT4WxmryiKlW1
jPywDXfHYKP3pAtWHxLBvEtXQX709BsAmLALV7BFDp626aZhes4RznFWoHkkmKzRm0LFsMezjPWG
m3H8A3NipNHZGm8zpYDpsScC2w4qVgAzYL/UG94zjVCHhVFWaz0SmTJHplt9d5dvL8T64vv+MLtz
fi1rL+0luz1XYBprcUafzFuIHe6YeNNU52uzs4B01b10xt7F6B/2ema9nZu6wpluPbL8r4/Ezpb
zuAR91+sYB2Gz1BW3QSQHqz1XxwTHR4obCpvpqiJXWBsWDjxrCOh/2G7LTeljuHZ/I6flYPHa6aP
9rHYcpcrYNlBqwaTb68HenYpkt7bAhoFnc0AAop+M7sGRHwpCyRRUJALoApQ1YoIFFx7AH0sTikI
H6epahVrnAcZ2fncnuYCgr7DcoAmk0Byop4zuNmsIusGCVGawlzrVYa/UVa5s9HAsDarTmebGSNh
Y3A9fH8Vry0ksvYVHAouRZsM4N7ZHfAgdWbQ+wMD0arFfeGi3ZPLSVgNlGgz/19u/+7rdaZVhtwg
K8z3SDx1vdgeu6tj3YBSBxI1NBQYXGDEtmJ3fvutDfm5/e0RyppmJwI7MA4PjkWEYAL4sRytLWwc
r0M9lZxpmu5OVGlKyytftQrFwxAKScur4AOqNBTVi665f25cbVa7Lb7wL4AuL250BHMq7ACxZGi
tAXbtum2OFksWcbF5bfmQ/0MfMizrdo2AvGDAAaIM5858RKLYbdNU6skifvjVVFVcpfXAX+PalzN
ED4PKyTi05bhNWSiGz+7cAxB6etbTQosyWak3r6LEly9EtAbR3sAC5cnh4pMfda1ZqvfIuP8vKyq
LhuY1rUZJdTvNkGxXe0uBPvf3Q3/kW3KtZiBX+KdsPnwGtFR/u6v2wPbBoUSHysF70cJ+CZsK39B
sN6Q53tJ2NdbVs+CYoaHVxx6qvWiYLaXd3AWpPlrOxRmtJadmpUjFC7NbC2xX8UEaDKx15YyowDA
IsY+CmI2HT1b3dXem32xT41s4f2aRSCewtPwTQM3xeo9nXxvJSVffhV3oNrURqBD3sPK07ePC3OJ
qpBlROQOonZlq2yKCWadf+bVFpg5+kRarpKYuCTGZVS6Di+NlNxBVks718x2/qefDXj02BTBn6+
Jvalj3xuBIF49oNlKzOZuvsNiwwmoidfi+eR4nzSB8oddp/8L81k/HB098YEuDt7D8JJTDYgKiGg
BwbzOM6hHcuIUTNM3leCRT3GBztJLOYE+6/V1JfMNPnzLAUwKonpWazu1rGSH7PFfrmQ4r5qMymM
55UTtg2I7G4cX0WVlUQEvGG9+m5bJq0qPWiZHcylI2mOeg3tEU/uRA+j34ss23ygvPPytRZsq9u7
fUr4WZU+uaJ208BZ9JuqxSpGvQzT2JpvcKcxvSZslwtTI+2pzNCWALFRYF9p1SBkz9TNBZ5kytLB
7z1Qj78GTyGWChwvuA0XZ7X7J4IjEWjNOT12Q42NfPzrKF7s8N+oKLBdBb6hsTsB4WkPNbH5b+lc
fk+lufkOc/9hHZGXb15VduA8BDysZYrWajm+f5tbxIe2VeTGQ7N2cL1k+0/TCTqp1cW3pouP51Et
2qTI+mZ7dNlsgl67UqNrBX04Jl6zaPJKesJlMpoSeu01dKqGDF7R6uAq1aokaSaYe48QtYM//B5z
lhW60hb24tF6Q/CYpKkNJKMc3CIgfWAlM1v9m7XEuJAmgCAN68dl3u3H1ZjIMDGvfk01trjS4pMH
OjKm7hhFhIGNFNpNHb04WB0V2WCfJ1pMPT6ak0OvuyMnWr12uqomG/11PR8H6nKys2xQPI7IFld4
1l+Spnyha69WKEhQIAPbsne/tXVeupAAVeT08kaO11YVmpnkjr7wahNLHIze49K639h+TaI4+J9d
yLSrHv3mEAWFemRhyvr4HR7OE9rygN2IJl1u3381heZCYrXBkhv4g4cFFl0LY3XNkdIMLHfbIXyyF
Pv12BRjJuxgE9ZALTMCKqWkRtq2Z5ZeUppHdDAFqwJO2lYoVR5rrTaXZVlWcEq86bXqqGCYg6WxO
lC1QurtXyugFOX+jGIZKYGwPKydhCoAFxDQUXN30uXLkk/LsU1czk95eAp9pQOJc7n4rsdjRBbsG
yS134sI858HxuG4niqqZYGfO319LZ0B5dlMT0WXAVZwyp2QjYBrGQw7MkEno6tfdBaOqUfm0m33tO
Kql2T1fwc51DJ+dmJjtzY2bkMgK4b+22cF28/r72yxp4Pd0BLkyrk+eksXdTdk4dyTF8/cMr11
R+dXV+PzC91qHjeJfKLP/ZLVi3Pusv2n7wu75ZGldwt12pqvKpMo2/IE3iR1bLuZknicqlXHroCW
GnXASg0tIwlykl/Tdb5t05TEGVx70VjEcjiaJakoXdk8yrmlQfZNSZKGae5QB6IrPpGXzYJQAylf
unr8hysdQbtqxYlG+oAv5kPJP55L7+fXI+kcNzbQAw097akgsC63RxUL7Ljh5P2/FqZYlMc5xskQ
v8jhgsQ/RDTa8IfWb4mNrNA3DRz8w5fiMBRW/Wdcy06A4WDPqoPiNawmkAlQbQmo2OLgra/cNAEU
CthSAslSS1Zzlyk7XPLaXi68mxakmxg4eMgRSZc3hWk3viNqbsS6e3IrWc/pMdH5RrPduJuq/Fss9
f96PTlD0U7j7xna1m326sbVn4OKA0Ny36oLLL3FEVJaQfTdnwp2ZgNkJ2fncJSxGpR6VMycprBxy
str8awKobtaPQ2iemXdgHgcZ3W8HHDh03VLYkNpTi312hLGxaOBQZUC2w6PXwrjsqEcq6N9prCH9

Q4DUDP3lKn2ds8KkGasFQRVtL2PVi2iQStPbX8fN2Kwmt+VxeHjA5mmWnTOKVBQwB3MbsjVnioWk
2MRPuUEGVPMi0Np1TlQ2WksfFjiFoklPNKFD72WwhZ6NjUN5L2M9YQc0hVQP8ezk/U/98PJ2o0u
HuwRHO3wCYezIf3ZIAVDvsHUGyP9V0U2f3uizP7wD8Q3n8qc+T2y/fxoKQWK1uv4HLcGTyfgsL5/
mb3W/dgvXjwVqUtmtWl/OseBvujSGoq4e/mVQ2oBq+dr9YBJ0hzXBHp6+lcSHz+VrzmXEC+pqzO0
WkBb9NG8YNW8JuT8v3mOapGJ3FDR/Nx1pi6APQHlUkmvDrgJwTWNfZcgrAhWqJCQKTAmiGoX4tAc
NCfsvwxQKFWwAS+/K4+rXchJcGYbSWIE+M5daMbhj1ur++MrzfIlzoAhaqEUcVDlU/zhj5RW01e
pqlZLf6MOqcvqdJ+MJFiGj1AasXq/LAg9DRjjn8ehcBoOBVGkpqZnK4vxYusMjTe/ce+Vn655YHM
PO/uWa0mOfMUCarmYS7lwVA59Z4Zj91hA8z9P6arKQE7uyinTRyAv7sErWg8JxXlGzvGxrduP2r7
/rj5u+bbSL0njIkPJGluCD+JVXqDDPcYeUXGGWb7e9tAZXZbaYsPIVWhbImzJPM62tDRxQQjvY1K
zXhQtWXij15bCoPGYDd6k+IgsraPGMwqpF0LCjHkAew2dlCxoDWQwqkUiZL2NWdbqECmmUuzGqlC
02GpAq4rHKuPCzSYHtg11VaeBQZZYeJkHPs7LSQnv2oM4nnlYNcTFhrpQ1wD78bQK/hJXlv+nnUn
yNqhKk9+ZwV61zr0uAD7VOMocfdNqS4/Wu3ZE5KHpn94pcfvgcdn73Rbl37442IFD1n40FJiwodY
9HL/8cruNlcUCFsTpItlyICd/56vWHJcDxj6fHbrvT3zMca2BuawK2YH3tBlA5pWEegpo9nqXpNf
iqe2dnZybFtOmnsUghBe2NcFONP/LzSk9JCXWSG242Am13CBoautIE1XUpuULLyKQ/Bc4MqhI6ms
ZwGG/mLbm7vQkuZW05GahqciPbp6PC3/T7SP6yMlarmAGYNI2t5ACqpVFWy9iUKapYN8scAXqKm7
mk4yvptFoy7OzJ/nAg89VclqsIQL2n0qPhZuQFG9k52ZyiaHRNK1D2YrmqVSvyRjzcUd+X4uOdb1
/CkJ9ryO27d+tp6bCQezt7ei5WEMh6fiINGcru4cNy/97L8uILH5a9gysS4pg6g0Pdrw3/dXF+W2
ywieZ/H4bVkgQ50o6g3u7ex6Rc3mi2uy0snDt4uG/2INNwKX+z/fjf+xazMox+GsdfWJbV71rMJ+
85vnf7YM0Epu3ihkxkBCrYaO9q1wuT/fPKJSjgZfXXeAP3GUxxARptd/NUCJsbb7yxTYxBKdJYx7
JUgqVky5vT0NkxF9+DUVflSwJLPlutAdhCW9gMUMTveqyUkoUMnueKVj7SJgKsthPUKhJqI0vtM2
XNAief+lg4WF1V98hmDyqlxqD1L734bD2Eym+pHx2BnjbCoYy1WL+2tgH8bAaSfOyryzJpnm+nJ4
5qQ03kfLKUNRvTYfnb7b60mj78zy7oeaYW2XQ7Mp2d4W0R6PB1fHZcp6ncujmf/bhZPJPM7g0Vdt
PCYQ1mVsD5vJo7Qetv2477vUWenWRJUrZ79/9Pb8qbdhsN3mznL33IqYryGqzYF65n58/E6+V+nG
YejJv2m06mZ+qHcXIardgKFNh2murjkXI2fReWyRejTMT36Wey79TSQBBnx5A6WqwWMjn9HDCqfd
dkrflOBnxswGW2i9mb6XAui/h9T8WxTYDAAEvTw918TMA+Brwb0VlmaR+z6Yh3KEfWo/MUuUJVZ
vgwgBrxfpcWnqCjH4uO3R8jsNo1ALhLMRPASj01/HYR4v3mVqoUfe5spWOG5ctG0yBfbNS4L0dHY
sBe6+VTmmeaT8sFxDmkkn9S4AzdeJN50zPXhE76KjbyLCBgSr97L/F+WbRDMeaJWUdhlaZrOmZS
c41JVVFLM0T+dFGpd3fvoHy8vv+OJqalnfhNuH/tnfMHBD9qL5+DcsZ2e/L43i9y6j9e500jZXUa
FaQTH8TANWMEqSjbVraMCFDeMBrEAXlpWWI2Rngx8tZpX06sXVOEzclQUupFWoE806tSoCiNLXqiQ
lWULj9QjRgRfMwXApOiqTADVmFBjCocj/2YWD/1UZfiQEpf4tkg3ry7IivoZR5BT7zZIJFo+udg
VRT4nuqQ8r7FLT16x0wYkwc5nNZjLHApTCB9eRGcX4gN9MX5fRaVYwFZtxkT58batQKiI4QXWOU
GP53OZzhd15BrnwZpin6DH9Wm0O+KFbU3x8DpyFq1K1GDR8axQktWZVjY5Ya2eyEg6SxVtN7xjOt
5HcRWWgO+H42j1o0WevoTdn+BMxp/s+/x1f/Rvfk7e1FuflWTLdUxocb4Tr2ePx6v19ThJw+XsPg
xoKGk8kTRh3etAhhMr3LhLUL/F7J4Wy+8OHgcIculh9VnID3+Nf3dwk+qQC1Y6sE3MLLS9jRS4pP
9JORS7qom7paHFB0ZG4V9LDGFB4SzjKSwMWqOba2SYmhDUAWbjeFmPJFdKnRVzSVXhIVMF/9Iotb
7YJBgYoY6WWnroclZONVcli0a3p4BvUAlxY/3Hk5EWbHBGc+HgbwYWGzawsWOn2VHqz04jx2ppI8
LFYKsu3sP3c5Y0qXMc2VphzACuy5pmCqolezrrlWWH1G3KQcUpLdMcXgcVnsh89IpUeaUpMHby+U
0TV7EC7zyWhlm9M4kWFx041Co9/2Tf57/5zj/84Pcv+Ka/vaXBnyvbT/yDP29kq0UjW0+fSJPoC9
I4emy5mA6ADKOizCcsjKtN1XP4ZbAK8K/oDLqWToPhnlTxQikr1fw4gY39oc0G86C1yqDSQXC6Ot
VCsDK0m4tVidYDs3PUMOUhyApW7f8Uje7Sz71QKbaV5laaKiJAL97Hv+mrKAau6vulpohYfEBw62
JQlbUoOJUKLQ6nGrVmeIzW1gIkWp09tYBS1or/jm2qnAsVEhn8MksylbpumWqJTKKEaEUsOrarA
C/u4UFIKSGE64BrbCpDQV1940uYE49WUAL/kGRHRZLULA8r48t7B4qA0gYGeZEd8czwRhO8P4IO6
1wXOsiy45xjM8RpUyle/9ePE/fp/ceP9v/5y4V9/04fd1cmXG60Uny4rrWZocmZYH4+nDVQplAb
pGxclL17FSpmGyErGFVOUpChcdg8dqc9gaKSwytyJB6TKwJpqVl8T1FypSulpkCVWtjqapke3UEFz
exwR5XtsUxCAUwkGJfKbdsb4k3eT2NTTTaSXUKBowY0PF1Do7og9Pem5R6s1qKVhZTg5gSWKzvS
/UMabPSiDTakfLscQsaFivtbGclvHSjcfk6rVr0mm2VC1Cj4MdrPRLAcxms62F+jxHV1RRPma92u
gMYwHwYWGpLT2Qg6Mr6iIwcfQLucAT0OsJPq/FHkxpMWLN862tc6VdQg78607OVfWNp+f3gnR+/j
SeCvHG0HbcPbzOxGOqWsqf5uOD3/51Wxr82b79y7/xjQZ8INdxnYMqi54qsnw/jtZ9FtKzbI5Bm5
R5xGcBK9jUOTSD4QEOJLWIllpf6wRdVQ9ktLTt6dd6Ryz1gz/GtA5vSP7NAehM01TLSTN15KV2uJ
Ai09c00FC5tFVrSXONBDlqy+nVJ2WV4NqjtaWBNGrsbXSdYKEDUF4jtxTMuZFqiqlICuLVNVj0QB
0lWlIeB4BAkspAEv3vaL3S0kONNGPhaVKRsTWUu3bXAFh6rtlahcKrdCmDwQF83DONLDN63Mb1IH
2dT0+kAYvYAN3t9Q5gZSeSdTr1c+Pb0u4f4BquL6bDqwoFc9005Frjya6czTwf9sp64QAXXlKrq
ogDrersoxxtPVfY2jeVEUSb2qwJvHKsGyHsgiflybpyl1m8vCymC+YFyWU/zfNMjapa/mXCwLgzTK
ZsArl762Nqha4fPaVapaKC2wy1/Y0N345Tz0wlUnhwbTtGTTcBFKZXdeAANY+lzqnZo1nUO5MZGVf
t5deCvzuFyU8uler1DRcB1vAYpLUsNQddzSVSjQJpK7PSpyrq9jn5jQR/XtjYpHVuS9Uzn4BAkDB
5VGWi5TaWJQheEfD7CMbnS37uswbAcYM5yAwsX6DUobKliz/Br11MuU9oIYDFS7Xdq/5gCb2UNWo
8qi0z19B7TJoYcn12XY5pgNWPtkWW02rabqjDBgVyWSTnUqEgW5y07bFauF7we+jsGzafso7PxW1

8dlPtpbD/47V8vX/SnvmcL1s/WrchSJwyCmlm0fGe8Hj/qBf00TqJOVdTBxHKj35vCV6XgdQJrRT
Ex5jS7rZYm+GHKNJdYaQ003CrjZveIlpoW2nlzPqeolHl4Kc6n6qDhdXSsrWMqwJvUgWppkuMTY
TY0hub3TG09DrOQv3amlJzFCSBa7puPXiqdOt92AWYZq2ISKte5oZKrbDrJeZs6LlZLiO+ZzI7H0
tnANpqDETGPwDkEACe46PX0tk/kssA28ntO1Iu4OdmCSj1QJbL87qtD/tm2x2vS1kFtcmjFhUVLq
KxhO098TkSCKfYCFpquTl4i+5FA6CNlyuJCTZFT3RB8hs90Gz62+4CF7obr6bMWDzdbPUCnPOP4j
vjbKGm1AUSq8/Euv6r7/hD8hu/+X/Z+qyfbds//wu/1wiw9Fdl8WQSzd4FKtoYDMvCDWNr/S6L3k
HFVJU+KxVkeV5bPKZphv2BmkNPb/jZgxY3s6wpnrmZNMdrDLlUf7RSuIgOU6a4A2lwpb6qs6lAqh
RctWSovVtR+oxmoT4cKSzHV7DDRfWcGNDcGkH/19rkZ2ldNW/Lfxu2ljAymOVrbtdUIbf52St16g
d+ox8OpD24rjnZleguaLElq9lIpUaX05Ek0zNpDo5k9/CSzm9lb2uxnmvdMpvTudhQxJwC3+yNpW
SpxgIm9wBiQ3K4EKTt9KPDzqf2vbHwMBjHyDjnxbb7R+IGHY2Qa5M+pxOkScPz23Ne1/PTV2R0/6
XHsXC+13YD+K3y5OZh6Nr2ix/mJ/o3gfgD3/6Htpblsy4MDjPlut7VeDn+dY7d2DccB+5p0mIUln
NoguZAzk60tfaVRfekauwnZYtcWTDQcigerARVGzcxMvxfWnCP7NdW4shaCxZ9seWtNqS2nUBG
abchUwYzS3JJCBCi4G1RYIsKqnelCMoNZ1ow8sam0Zoa10oTAo0MRUkGlq1fjB7pQqP2oYqRbO6y
KCz+MefLfOr0YsOZYdaosbQc+8rQ5xNm1ViKDV98NdoVaZ6YA5mJGEvYH02LbHHC4WHSorUoaGPm
YaL7QKi4BlgMviItbcl+vEghLLOdbjyFZTYSTOwutYIklXo0rGGhOgbjFzqzTBDECDH+MEM4dds7
7f18XiDKC993h/9yBO49WP44V35ZN34ZSVMUZ88KKuPvjX/mCdR9tals+e7Qf++jcyZu6flvQhyz
Rv2r5PQZbQtOyi0RlqITxzmDQa4saulAxWiK7mevMHni0xJU740/yy+dl9yZN6jqllbCwp0x4bn9
GUGmwElKo9sAhCEyfGBshWbekIcpUBrSlu3WXjPJhfQ5rsmM6rU9I5F9W1tLmdAHPdWgCNLqFGnO
3635xhQ4vrcn4rp58Hofa9EphqyZJav5izd3T4s1CmtCt2gHMB35XT278g2XIqnZ0D6VNDGO9JVp
MNe7A0GMUGehchughsJxStnKok6F8HO+DQq0wF2SR3tP6XqZ0yW2uqhsyCfm5B/SdWXZlGkqaxOx
ufBCsKhGtPbktpMmjzTVy7x3Gcl/AIDBF99WF+3O8eHl+Dx1fjCRPvr3jtf+Db/sAWrJ81dMJ1TV
CwrlGV13EjNfElNslWbT+ccpJakqy0h5OWipFa3nQEb0AB73wtvW5b2r1DDcYyLa2Wig3TLkDH+l
mXtJZwJNB4hQnGB7W/1lbylMs/fdaN1bxQAGRwRjIYZnoWq4LV9h+AtpYz9ZSGcoAxUz4KSNBK1i
Br7TEWGwaMLKZY7LocUuNU8KmtjSRpww8kbLZU9ZAVVoxmh+0DHYHBAozRKz8h8fw+LOSuTpJj5R
GVCju7e9qsQEpOqVFbc7ks02xpsClZnEjKWb08NjjOJSj3cjmnlRyL4GexhsEod57VQTMvfLM1wK
ZFFFw2sxSXXz1yAWNufjE5r0dUws/Ny6Jw/OCteZ55eHyBTOf/9VFufpp4hHh8Gfz5g6LI74GcG
7qQ1l9dgaR35Q0OIRpLPMSTl1lMG3fTFbHAWjvzDctov6yQdnKQikqKW0S1aWALNA/uHyoliCBZQ
2GR3Ln9kfxGk/9t0olU9qqNEgPlakdSq+oY6XFrkZd6MCHab4a9aU1pWO1ocMaSeYNbNWv571FC8
S/aypHh1Ix7rTZBwDvwOqWVlkrI+qoCnw+wJpR+pQwKOVfgQoRVWZulCLgl7d6qmgYA2TMoxI4Ht
gC5UKZ94wWx9osXsIvzzMXrMKEj9nHfnNZp6kCXxO+AG40GmTAI4rWS1UBCJt7WL/C51CkFA+pj
FUoBawzhwineiaGmjgpSJDtDCQmCAwsdhe3C2nI93GFRbLsbamJBgy7btdHfwHbELp/gkBiJE49
1MPeOi/Qgu2xjX7jntMETDKYuUP/BtX6uL5pf8mb+3Besb2l8lDEyjaQZtZ3LnuT3fb0e2G+RVkT
tUfGAZHSt3ArcFKlxp4JEWzw99abMIXs06lt5oVGfo9A9UYZ8dKYyO1qNuUr28dYVRVdf7lsZGtb
Cmykpz+ZsKpFkacLqIGNOKMn+qVYNGrhaPFOGWuVTWfG8iv/Xg4xr4Fp/0PSw4te9tcsAyrHJGms
nihswSziOvWCHkhRrQcpvtuqgDN28Uz6WaJSKtjgZ/fP+SjmbE8qWpG2NZSNjZkyYFzs90tbTQLg
NtDmBl1t/tCfNcJfZnuZqzk5dVD7i7c0noFCznM20EKBSwjCLneN0xLCarpkKxqFaRJK5ns2ExvA
erveMEYbleUVYNfWcnfLnM0w4M7GVcyJ9+zzdqB7g6XKDri+uNkAqx1g9btVDqrmOGKWmzDQIaG
0t6xt5e+Hf/mPDDt3T9Dv+7Oz1XbJQ+HfwZbKQFGkRRbVlo2o8gaEq3EyBZDLoXg/2ZH6QvSv/
xWObv3ijZxU42+H7JYor+ZdWqoBq7CkvNNye5oNS37AVBlU0jI3KnmR3kDUXJUKSuBaqm3WzkANK
g2szeM3jJQxQWAgSaxNrNXWQ5obL5K0HCmZzi0mXXftKz0J+3S0b/nDOWuQVo37sJm/XM6lUazHm
rlU0YUQNzywTzU6QAdgIOWMVtPJZ6B5sLahr2ujoHMPFa18qqJ+Rli7k04fdz/AcHOGdYxCj2pt
02bq1IQSUL5nlZ/cRqJvq4vr+jvbS8lizzhS5sDvepNHonGd52sJ34jIHi/OcZ9HZE47nD9Mkvv
MaV44jTq7SquLKLihzhZXvF2A9T8FOSgtmFSyphEtgWpZdl1F2S37kb32TvP+PfucWrG/EjaLaie
9Zhuu0cEN2nKBp5FXmmEBFFkdmvDhXKusZjloiu3LVH/T9Pe3hJFCHw0tCpUNaYFojh+Mn2GBOSF
EQjVVDhqmNagQWg01i1D5pbQLqUsI6iFRplJdANTt22Gp9CVRaRHMTpNKxp0qftXZCfT/SQxOWku
WKzM2WBDyopQmAFbpScEayjgH+K9eckv6uznvFIlQEGrjJVUCtD+ue6oKh6hSdHa3/TUGNrfa+io
3Hi7sS9i/BcvbV2rMmmAuT22jIdHKiM2tIqzP4q+kaAAToudxY2iqIc7EDrZjSciZSXLJfDtkKd2
GIFwpUHenhlNeRlq8s4j0vbLlOryubuVW5zf7BiCEYCFGmmUjsA/OGDnC43E8OGNzgcccxlKGS3
UHf+hBCq0AtR3137EWEqx2HS/A8z/xPf+dvOvrV20Ljfa5mi+r+/hjl4btikqpWbRUU3cppVtF
6YQdibtbxahMHagEWDQENHy0EX0OPk7aMq9W8/JcrVWwHmBs9H9tTSXqpTrItKrcp+laiUxwHNxIx
obCkYfUdMwVS3kzQwKg0nqbpW5UmF11lhdZBl1DJn516qm15VTas2y0nsmell+rqv6TLipwq/yQn
1CHhuJ0VZulYUU+PotgET7Xynnwjwso7TwLamKT+XDZDXDSVDjOJd4/JIErQNchIZKsXRbA4Cxqe
kl5qAFVrrMchUOl0TqumICBftmWwOynmiuxWQ/LFaTOF3pkCymw7oNU9M8Eto6DpKBPVjizHHcQq
jFxnps+wYkoMp986sQg6TaP3luNZt0H+KgN+nb4qfMa7ArbpoRFLlNMs6mm4qa8Ej5zU3zNqyig
7NMreW9Q3pr8IEuABpmkaHsE4D0EEXq76xWk4MtSy01BTHzrKN5bM0qT/cvyInt35ehpcekXg5gz
VhZwjcs3SGG+1Q0y0X81KZgqFWU10kUftWVesn4KSuXjLMSj9LK5lIe/k7eKnjt+vUjwLa19dpIw
DBT60mrWKq03NqSu3UZYQsWjfnV2vwSJXNWpmi9AAYBs5otlMXBhc+uAe/kvTYZDyqqJURoK+VKR

JaY8qUBvBbcTxJSskaV6ltQTOL81dKr3pUmbQHewDaWiL8nQtZtcpUFI4VUgRnvDyRdTSBjzysB2
GBeazgf6ZRRrIuGAct8dEhh8Lb21urMOZ5rVbYc2zGcVv8+fNjDoiwd26ya+Iw78D0fij3fBBsa4d
w58+N5POa4LiMcYVJ3G9I/NbR5X6+xZW2KTOzNYmnK5kLKT/39Pyfv+Nq/uAXRg2kr11OnSFb7yX
p5JY/XLcPk92+o9dLIJqd3m5aW0DEAU4zvyWDnkuQ6TsKXMSnk/Pxc1ilIMhxOduHkaUdTGLUF9T
ZlNemmKqnUG7ba1B/WzMuswbxby/yqUqTYX0vIr8XQCVT1kJ++HnsVqnn4ZSbm83agLfczL2pwf
wgdkJBqKKWUWX6Apa3curfKzuHDytKjRmX0hwrP99sKaVmAYQ2gmPRyMoIlpGR24W0v2AjPP1X5L
DVesOvZwqJWkyMAuuYj3gGF7olYaOL51oqas4AkuOz5nkt0zRTMTZmpY/v35Z+vyeNsA9XpTVJi6
RX1YkDcFVYGHcyjaWVRy3Hb05B8xdpkr1YZukHLdu6h1Pg4B5H+YwhMw2mG5RarpUgzU0w6cKaa
pM02ebRfCi1GlrWd84Wza+axpeowHf7cZydOsagNXI4admpIm0T9rhYj1QbchlJGKpkp0UCguaOx
qkiaJIilgGVjTlZlNTNxrkpVRvZJ6tdYlGIXqvB/K6kGhw4X1u2g2tzbBIBXMZp2uqSv763QP7y
dzE5RSobKcrG4i5d5ULzHwVVOXZ+j1Yp1+SKjviLWpk2v1KFY3JOVsm8Vn+/bsI01LG6qwbRCLX
yuc1cdVmgV6aZez1RanTM9s5ro53KSAKPN5A6kv9QIztcLDXhRkTEIAGojr8/dqBeZ9eyuqv+bMO
kBPrvVuYrFb6T9ustlhMu9kP7B1Z5Xpj8nRXaI98HJdOZk04Zj5oblLnAuge8704b47y+ydIzzfY
ltEzjJOa7dkzjVn2PUT6bCRj0fdhMB1tVSL5C5wedFK10pP/l3/4y88w/+1W1RxBtiqwpW+/Wj05
cfh91UOSZb2hDdKVUL2kEAi4DIIPWUCOVqrQAC6Fgx2sm4zPVG+IcmChNNGmvyvdS44sAJZ18YN
j4d0aCK3kQYKIVV7+J7XFibJrKLfWJNdJrlK/KvBh1OkdDTvRlqf5AjSPberXnVbt3ygeLgaaHHS
jEeBrQ4Ykz6qoVQxx+xc4ivJ9ibByKTKFYgZ4q04bUx2P6FwAk0TXLSosv2OVjOaE2t7PRvdjkiF
kEwSNMQYMTLGzMETNyzQ4kS8sSNz25Vt20v5yNpEgLFrvPV2fi2aWWE87mIxmDqcxGt9lFgdP1fg
FIZ+/qK7CM5ziupWFZ5/j38/F6+VyZx+cAc4XHNZz6Y+1u9ztw6l8E//QZ+KoBfpqmeUF9rRqoG6
v6AKhlDVRel/78T9/zzVuuvhG2dDV3osndA7slvA6z2auKjAV+tTA2CwIoew9fi/k+jm2kxWB3CJ
vBg2Zb86SMAFPdkCkRzjItZVPkQMtF+l1tanQZrKFqA/tKgeBchwnXf1GpK2EK0KglSZbpGOPYX
Gy8W0ZpdUeV2tTLCExtb/GpoDiouDfUYpcWxNbBbPVj97s01Cf1tIIMoPHnCzgUoFQyWnD+LCBWM
2h2CHLDNvPUNN5XABYvjKN1xpcYAWq7qYgokjFeom06BvSr0oIBLAndfXAm9UVQqyE6Nu9aNB4X
d62rWzWtxXmh/gGNqNpuwMD3BFHClTjge5L9PR8RNFFl/GtWBKBuTdAPL90YaVUDPoDnb+Ij7l3H
bsHsC6xt89LCDHQavzVLPX/auwor+DBRAXIP3PqO8nAeobXfD/TUODy2hmxPdf9g1xLSPSXU7W5w
03aJuctulghTejRANCap1A0aL5QmJ229D3JKCzWOIkAS0sZJ11NfjzrBYpq8wHVpD+IASqsoxVNw
kT/NoCp/NZqelrsNbYEduzazExvyFhYOicGQaQtJqnqjWK1T9VS7kZTGyWY8XStrz6pOpmAAKQ0W
Rzo0pBQJubSqJiI/UiF/XJtJBcGFjRRGLS26lraeGrshA9YupZggfk7XIGcd3lEwJZeJZAahuKQ
kspt89lKgv38tIMBvrKTC+xPXM4rn68BRTmy7gix49hPMNJv+cSx7NsfDtAndl/XWju4u/rcQFvX
asfbXSnI1DOhHNJ1cavR249UZYmu4txsKYyi3S+N/iNP8TTmeG6/pf4Zz/AwfYZdHyGdvzn4FRTQ
HTGzgrDyDN1ZJ+mkDl1fvJv/unQIe/fQvWX7PAUpYaVR65levtpNGq5wVdl+CDDYTp/FxvcMqlUD
+IFoFdNef/Tnp9br1TR/neoPfvX9PC9zDZkPSCaoBFQKKlk1FyipTq4cWi7nMpyuZw9dlAQIjgn
rBqd/kuhtliELlYfb2j6Tfa9czavQbsdTX0xxhpQ5vTYMYxLJeHQReB6RmtayvzeOSHRDs1DZSP7
wCtVdlwnLDImJlBckqFi5WLJpgVRNHeVDLUK5HdnENQIulbsOr5V1YwuyrCBwLODy26LF2GrR/NS
90X73hDbgGMWdtvgSdIxnfyulnp3dxHYc630blW+KpNtBn6zks6G3ZufxWpeDtsNYuVqHwepFKce
06YCVtq6ralu9/KM/S1/HUR3AoZ0Y9Q/pFLFgTfAcT/Px/qyK/BwB/FQA6Mhg6/wyBKhdvq5/6e3
9a3vF1f20L118Tvm8rTbTz6f2G44d+kWxmGhRXUxFaLNHQsj/HaGkqY7qGhchSrUrKtGbWlFW0wO
+gtUYqccQxh32NBmugid3ioMkrajZNZ6rgzwJ36glHcQbgOHUkeNNR42yS/znuoLPxufrA7VZT2m
08eqDdZV0UUVmbtrhNa11NKWvazJgK/Vc9BQaCrHoRqGhtseDw5qTVWyyNmj4/0YL6LM42Cvy1vh
MDQZybetHMTqCqDGitvYJjn0mlyLS5ndVbKsBNOs+Rlgaj1lndZwsLuYTPORvf0hY4v7Wr7XI+3A
fO/5mMz1RFoj88lNXyVItKdKQGH3ndf1na3T3seq202wvaGlfLorEbeHuUhQSfzppFbj7uggYV8e
qLcRl/BMc8xaOlKXXTbJicf22Yr8DaPw+gzkg7NkBVL+BTAfXpr/80tajGg1XwjceJ3zRgZaBFTz
jsYeUaeSSlNDEjeuXQzWhpTPCwgiU86Pj6XTHUiAGzdaTjRtMgaoKqvOwfl1vAmHgZ211ARqDE
sRR5EE7R6otaen6ObmOydAGIzibBkNBWf41DdZUYkwKxQY63VUS4nSysPSMv1TixJaWv1j6Y0GEN
plxFmj2a1xQ2VWasYmkKslhT74XR1zmo9076jn88SSgJdhcYtFibEEjY6YrVaSlTZCxyo8qBX54
CpNJEmao0zlhbcJ2eQqCB54DDE6ww1IMX0B9+rM1/LfVmOT8Vg4QRF0JtFPbAl17acejj3pYdr1j
lM2UzFoMjG6cocIG92j3SgHlwHTDNpvSySghvnpHu845ZlX5e5odus/snsVwsA2v6z2u/1BzVQL
U8/B5ZtncdC+HLFS1wbVEvgArQ8iIWGvW9CCYZF/fIRfR9A9Q3mnV986RuGHENARdWaes3d+LVcl
Gyn9JrDnEZtVH9reGjuSqqzWFRtJSNKG4ddohA0WMVpHelFNQ24sSwPV8ihv/+gOtJA45JqVjqo
M9mnXbWyl10RbUm6iwrPgdu0YO6DYsc47LUFNCZIMlnAvXeJpI3ctlKZ9POwPh/WkJbU3PnJ9Z+
U1Fa7qaiWNQQPL9EH54HN+GOB8OrCajoRhUwM7lFgR09ykGTdFFyzs1M9N80TA4yUDbUBnrDKNU
Ke4dhYJ2xSzrR0tTTSTNiQ34eVDdVqz05v47UL+KZt1QsuVguZTk8kqWJd3MJGQ7r9Piy7I/Fqhu
vI6fCnG9+3rRVTBVwHvzG4C0qfGMwQW845jOW8SJmWvbwK7Dg3QDbmIH6rgDUMY79Pq7Ju6QWP/
8o9ZoqLdFwofYXQH1ti9wFQH8KVvUi5bYNMP3aY1VDq1WRjcxGOE3Wk9SkEK5KiSRq6fz2roKGIt
4BLGynPdSxjExrTE/va08lhbPj+2gt2Cqnc2c26Q4H1sTSKWYftqKZm44aWiZOoWMuMmGENKutKn
tWax0jQxcBgZ4pc5m4p+Kkko3NFMdmKWKk2r+MZ+nN6JgbvaZMo8qkvkZhaqCqHqAFCLn1kKm9nR
lp9XfFDdtKo3O2yVFZMJrreTBlxdwxj/Hc+nr4qt26rqvjmQzXlFpfrQf2qi7YrvWEGVvyChWSql

ZTZSo+FoJqz4Yfe6w2ioUWdSugr504nlkrWLR0DOa81loGG+FwqhWodBIv1cKbQciWv6umFGe4Nh
OsJg2jMuEqB7ej1fLzglb7JaA9xtfyfhzYXUPD6mZVlNV9nOhBXRtSucDobQD0pXd+3V+JXntPfB
z1/YT3zRvJur5pwEopTbfVT6LR7XtZvDg2bW/tuEaYJCSzBVgt3GwJ5UXEUm0jjkW0qaK/6QwpK5
3bpgEdU+q0jMWCiaCxmQguGytVaDqDqzetMQM5Hil4PpvJ87fu4Yr7OieHNJE6u7v9rjQ9WzWSO
ApMFks4GihfJaCmrIIIn7pFVPtXtfBcXAZ5VEvNAUm1tSCiLqSoazLKqi6waLZDCVqPKcXmyEgOSo
7iqUa1x+OpTFcZHvAn4X+fTlb4fQGfeaJiZd1GT9qhI/vDlhwOutLrgSJvNJeUxhulLnDqLzIgx
fY9r4mFaSbl4lwVEz2vpzlyrt+sqW0GV2Q1WsiqAYs8t5SOU2fYtjK4EDNJVxNcS7wW7sT43qn0hp
cFFwMLQZuiTD3DwupjWsvKstZes/sxnPKTeV7832AyvSLNQHFmPnJibGPFUMDGoo7wXUy0/vETbK
+N+hKYb2TAvnkCTi1h5fWPUuvWs/eyZPmyaTmzslx3QRNddnwwwGQAmCVoLQM5QdjS+lQq4DMwo4
OIKd4NKvjavCrVIUh9d9MgbCUROGaQXLfKycj+dkXbsv985kss7r3lGqDaTlSuTcn61z6oNt7vb
7ch9VpOJX02hlpeJb00Gk0HFV1COimMtBkMuq81uJ6y6S2kl3rL5Eos+MnzDQia5EOzsmExcT5qn
WPAMrZfCnnZyOZJiWiZUtuT8/llDFU7ty/q1bWdkbS6gzk/nImPhaC507Gcrk/kpv7Pbl5+Uha7b
Au0ZNqowlV9+WadbpXVfvDoCdxSqkbWE23peqhVLHIIk5m76obQSoewQ/mqI8sGkmjc6gtdfStvR
yWPTjQNr205Mzb5KrfaCam46/xWbZRFZcME6uWac+9Rvd9+KJuFUYRmo53VOT5K9T1xvFRZ7jcBJ
NMpcCf5dubanwGyWkj6X0jXZWP0tXiJozT7mJyauMGMDivSYuRamQcdChxF6SnOWjpleOblmtZrm
Ot/KG1JKAJ5H47kMduXAP162g+ldFd1gGw1/P4fCQfe+mu3F1VOiWcSon0h0uAPoW1ffigZ6DXZ5
NzSWD5+8MjVfn0GLWHdN/TtOs7rKhZiUkdNe/q9CZjt2wVYSM6GQ6RT3hqu5zpTWl8gL/TT94Dd
94PAKIDE+S0pIf++iH5Wy5lNFkBJB54nd2dTrd+ehExmenssTxrtJCZnEuy3ilY0Dqzh/RCiadMM
BJeWyWp1Yyelpl0jqOO45qTWAd02HAVw7lfaIKHAxrfQuAyPFdHGciYetQZmfPw91oa3eM424KMl
hJpQO9wCb8Nj6pcECB9+HKdOGf+thHHY7HQWkUTWDI74Jb87DtNfbBSu4Cm9OK079qH1ULgO/99L
+UYzwO3/GbPuH9cfB5XyLHP/PBT/gc/87nt2B9nbcP/5O/LGcf+Xe/6DF46wekjFYl6JYTL0aXsj
ThCt3ygtDkgGR2jKTrqQJAI7xM4UwnAB78O9OX+XymAR5aKlbjcc5Mv92Qt1y9WqvJV3W9YQS/dD
JdyHN3T+R+ZMgxLNdA0auvPlD75VLV6+pn8wpaqMptY/6YIiW3L77CqypL11YWRUYUZFoJZeC1pY
RuoGLhRmXUrV7UI9bmaAPov2rS0d6kJURre9M83USINymc+Rx/t2SN/X3k1iuyd+NtMgUlH+4N5N
bdc4BpJe9819Ogwq25ev0h6fYHcvtlF+FnLgVLmMxXaywut8XK19IMAx1t6Zj1PBubfrvOvMk0QJ
CkKx06RfKm/rsTyv37r2AxGKivW3D+Da5hGHZlvTyHi3IAKrzQjidaWFJsxgZsir1xCkE6Y/GK6/
ide5YuvJJaZLEaNFJ176XLuJMtPuJqOutF8dN1vDL4U7mu7ypNFpRUhFQ+kvPBN7wOwnwx4b2TAv
gKDY3NjWzIx77vrzE8kThh+5bX7v4sNX46vWHVDAKTJmKaRUXJcBNSwoSDqFgiuFonWmL4ajlUoS
kdvr7hufArnbqMkK0eRSrL+UI+9vwL8FGP5bkXXpHfDCYt3IS/82t+n7z9bW+XPJqBZjvvyNveJb
dHEZG9lvzG3/oloIigjM2hfPiFlznZDuC3YAFFVlEuk9VS0sLQelz6xDRbZX4hVlppPW7ttHIWa1
QPZlLFxAbutgfgmlqn040zuTcaylve8QF5y+NP4TxMufHo22W9ShTsT3/+b5RL167K15fvuR/2
v58q/43dLpBLLCQkVX4NLBVdm/fF2aoMQ0pjoOcjOPh3NnqUXFwg+KuvFactExcNkUYAycpvrirH
RyWiWSJ7Dec7EbXZU9LUDxz8YnmnM26ZdXdfmkG7Q0yGds2mlgwmFM8yBo7WRh2Bc37GgqSjNZHH
w9udcyDPPLAFQGM61Ltl+UPDwjJdgddKbG6xq5UpNR3BOqsD/EaobgB9anjtp7j30UViEE8v1c1
bqtlpNzVFSTYDC1AxHLOFzThdU4hetBaYwWFGkWjgA71XlPjng2Nx0vrBAGEV3vHn3Ll/R6qcWLF
B3MFTa2mp19IZrt3q6r3t3npXdjg/fMJeTu7fw945cfhRGR7ekOeff6meugYOy/LAM1YbFZVKou
asRGJdcpVtvkV3M1WirC2b1BTacpsK1oIR4Ggps2Umz334ozLcvy7Dgz1YtUCGg3258+LH5KmnHp
YbV/rysZ/7UTk/uS+Hlx+S45efl+d+6ofkUq8j73rrTfn8tz8qj968KoPesAYc6KoWRKhYuVfbV0
bFnVovmXNgLSoqcpIenm+EpMlp/fyFjhWLNIDIG8jy9duzhHZ3dkzGYSVYyql2oH8vJ7FyQqiI5xP
dqG14whltlNQMsLFrzw/QStNitgIcWC1gZV/iCjL3XAlXTNJ8kkHSxfapg0i/13ilYP9ON+rPwAQ
lOYW0rwapld6Xc+o/VxX0LieguDPXB65hbfIClsjzA1X3Y8mdYbhqRXJYI/qpSwpVa9N3rUnL3y
lgxvhwu+GrdakFuDZi2aDRTfx7Hzf5lCOBDJq+5hI/+P3/FFTvwuwcXgPNs9UK/bpnpnPHL1/bl5Y
9+SK5ee6t+Rq+3oz4mBxxfjMpgSmcJS0TB8fxCIA1+7X8W5OTAKqlV/jUHixvY4AjGeCmG38bTpl
w5eFhTSqvZVINGj7ztvbowcSE5PDyU8emxUtdbH/k5+fff94/kJqzsFz7zbnn6sevy5KOPyP7eob
TaA/j0jlpW1YzaSKVe5H3pKugUdgC63RlKujxTK9lq91T5gmWQ7HEt87pjKYylpY+u/cMZh331ZD
Eby/kZfOfJQs7OzmS1XNfSNFStMINGeq0H8PubeWmYwBRSznCd8VqrenPhfjct/uFEfwRg5IvBp
OHMT4Ti/pgKzX8HI8GG6+WpnzcNj9+VpLl2Q6++mG2HntO2NPxh5xJyim9VHRIcAcvNKiU1IGSsl
I93hL0WL/5qm4ba/rBg3GNOQceEyNmBcD1lOreuHJRqrMbu/uyC8896z8+x/4F9Ld3RW/2T1ZC
yz8bGY8KauHF2R+PxY7tBvbYVYBALN7bbhyrSeudJ3WZIMHKiLOW/m3GGmrD+wa4Gk5hA6gTaK
0v9ZamsNRj+JslrPDi/ESmd16UE3w+LVmERerkpI4GczRiv9sWZ3hJ3vOe90jbt+Xo6EA6WJQ4tJ
21zIxAk2YbnOPiSjdtbWb4bLSOTVh3TilOYNERR5a0ikWdj1Y5VSobWpWEzb42n2cp/HaDbrKuSq
AyF8t0EJZFyeAnUyo1Wq2kDR8abgsAnitNpiAA9ePEHYRVOFFWs1MsQLUvnFeuLKfnveH+we/Plz
xR42M4wOmnC8JfSwv65kzdXAIUbXzWi79Fs/vNdH76VGL6Dy9Obzut/beAap3oWarLriuIElYVYb
XmsCfRrptUrarpmJKu6tw6VR5c36mjM/qQuksMm7AoLH6wzFCMKJHdFqjZXgOW9GHQu6n2clpVLI
PQ17f//+y9B7Rk51Umuk+uUznd3LlbLbWktmVbtixsbJID9ozNYAbDM7PG5sEzPDBhHm/AngcGFm
EeMCzSEJ6HgWEYYAwM4EV2toJlyZKVU6tz9823cjnTnr72/9fdVvGHmyjUbDukWrdvvdWnTp17r
//nb79fYsnqVopUb0xJ6Exeo+Y8EkOLkofViJ6iE4BcG+qdpEMgstmlMrYmyldiVgVmBCMa3QgS1

ck/877K7SPQ+LKgavEIBE9KDDEROhUx902RdceE2Nz8yVBF/kcsIarVRGJdvl9XczPGgrbDPpScA
/LNBCq4pJukAj9IG9E2Op4AFPweXiFtRGKs3fPg1ycP0uA+wg5zCQSNv9hb0tek3LemuMUAZ67zb
lrFmUymhgAQcav6e5sC/yx1lzi63SEsWLU2+QNKPBQFESHbtBrU6e9KUEjZnXJKby5mjd+E3EHG6
DJxpp+oZ712WawX5bGatj2kxysWvHTVBZjb0m12DjU3L7weAFV0vFwh50SVKUCgbv1+y1hgTbdl
R3eJFBhxQzngbv2JT2Zbd3+H3y7L2kuAMWQoypsZmk8MIIX4GG9x3RS12eq9Bio8TGURKGOopAYK
AAH3GHUBYoIpgBMQAQg12BF/sIkDvgcBOFdFKFLJWPo7iEjoQo3mGIVhOwqUmcRDYOhNTCJMjXCo
MrWjF/jTkPBLIpIpfv0yh7lM6Xyfy592aPZ/k5AWA4gu21tGI7zeCUsklFlqZysWUIQEL1NBTCNC
ypNOyLEXqgfDESvkcfGkwkZpEK8/b2JuUKJQq629RjI6409onBsnXRqLsj+OB6hQ2215GHPWen2
XYWeXoYiRwzkYuL/fEL1UBcDmThqPr02hSQXGrXKlTfzSUKaB+d8duNg5+VTzuPcQXMHpOtx6/7I
vB6RU2m8JzulYaxXNB2GqMhx0b7HwYKAcA38mZIp6MiBIV4LEA7DXtB5sh5iy7rU2Vowq9ykiKTH
J+gCVSZSSGdXWKULE0dyVsFZ1UNky7VFDMEokv3keaoJkleRzYFYSVgX8fsQ6dyMQPvH3Oq0g7ia
Yqddn0z5fNKtSqiKKkMuB9wy6YTgfUOT+EopyJiV5JgS1NMP6pQpRUC/MK3OBxaA+qGduSyq7KQ7
XSHXt7TB2BhiXmUFciCgn/HYEEsR8X87AwctCPEm9Gvk99kIdneWGKBHwRAwGV+f20s3FWVPgwUD
Ae9aTIN0ELpzovqvI9ft6ouyVgilrzsAiBjXvbl0e/AfLbKOGuFZorHcv1zpfqc9C2ubrb2uKNEL
O3rka/ObzmlId2vlCv+oWEyM9ETvvl6Vk/z58ERaAoGNrsFQ+yHdw06PYKHodcAeriocHzBAjIf
qesNeSCrAojaeCy0WKCI8KBXAJVagdaScIrxJYkhJZ/FNVN4RiMV+MbajwGsPjQpdiguZUFaKE/z
dW4mYyMwoqBIDvIZYcjQ9BHRVPsc5o2PthvN4LakBckD+xPNlirgbhoxwOhECmkQ8NUbcTNHQYQ
PNFzSpmxqNQ1EMxgXjFGZG+WCuHulTX0UAGjmxoaUjCQmpKeeTvogewAcwRKqzqBLzPQKJucteEP
IeE9GHHXHovkiXzz0qc7LALEPDJklc3gxVRR5q7jBQEK/loYc00bZiD4zNCKbocWgsdlv4pQjV+x
OckgSWWzrr8PXni0XpTY8RKQiiZNhvXJGHPuLR3t9Vu1Tp/BABSBwI6fcmQulh4IOdzFqIRhc5
T9IyiBW+CtjQRiiLYELGIAYADUD4vU1syDYEjDPCcWpW0o9jyw4CNMTWUqRk3rwIv5rsdGyiEozm
FqMjNcDaZbQE+qxZztWfSJB54nCxe3kVemD76wI7C0oqXnPV1lYcVjydUKpoHV8++ijgWe0KbPb
cnAH9LyLgBZIBNRovFY6+K/itaH5gvRZ4HCKDMtpqeyosxzofcfUo3o98TnwP8UsIJJdcDTPNAaF
/AiAiNWNr/wPGEanHEHhTgE6Ca5vcdp3Q0UCJbQnDOoT88Nf8tMOUDpTuT70HeKyrRLTcnYBVcm3
jLKT7byctXCMmMg/aJaLzJYXMzm1852EW+DaPmDRgk4MVpgAWj/ULy0WebUZtfrgY6zU9FFzSaCO
M72Oan4TidjEdgw7vFL8+FyL2K9UUqVGpisOEK4hBYcSshQrVB/5mlws4PAWIYQpgM2csAojdt3O
tYUCJQiaOtKRGaIUyLgWNLS2Wg6YofkaVG4GAEEb8h4kZ7CERj/F5Fzt1A+wK2f1SghYhMyA8UZQ
ul6vPFxq4+E7ifDE2mBi+OCEA8v5GqsJw9NHJpjp6pIRVTwAxCI2OaStlO5F/ZAPieyb8hwYGSVY
opm0RTp3oCNZDNI1EhPB4IPywbCnc6vMHG4HvCyh9MoCtv06izrarbiSJec4tF2QSRX4P+BQYLIn
CE9kgdMCyBnrNAFF2FH0b1XpgeDfTRMTWVfVPLiQu1/UhXLsVRUHFLFUwE2c2p2ep3/QieResMA8T
C+iJX0uR57xvoUHZBsDERgaiTsBPCMsUgMJmJc8e8cuz5pQ/kStVWv3QF4q1EihKp+mKZIRSTCj
Aa8o5MbIpuSw6D5QDUQxoRxxgJqThNeV4eThikGidwTYaKTKeIzIHgc9B5h0ML8oMm+NRNimgXilf
hNxZCR18GboMjj2sAq5xRiiFTICEu4sprWqWQsakq3qamfZlGE2qeNtKe0hFqToDtbfFUpib7s5
RedjwVDBD5L24KbwJS+YWXtTmZOyqqzvbS0ZGS8uAiOqeq6VAHKVLa+q2juY6nkQjMtw7aydpW
rZEA26APwHAp+Uc6UKVAKdW3hjcFczOlqAghly+HDUESK7pN8TmhlDnpfQaOeMDV7n7uaF/d2dLe
qsnpylnkWB7RRrr+CTeNM18qIv1Gsaxud+7BnrUwVeyv7BgpUZUF2O2Xj449nBV791s1Rt3GG4pU
m/3aZOa4d63TYbgS2D2GAqGA06KtQT5v1EVT7Z8LEg0ee0Dcy1RkJFlulqqRKWUgaL3BRE0JE8zp
wx9pvsDYRmeHqdqKwCvYPwGV7RUZM88MaCj7VV+Hfln0w+I2apNbuh/Fs2nFTmW5UBKa8+LSQBeK
/C81jaTUKRSrGA89U5VEXbAOkCaHhBDJemeljekPcSmhShN7U1+bgr4TJYErOoK2gqU79GKQ8oji
hFysYm7teo017jD5+nls4Fqi0clTlHDB04fa71dM3SiLFAvyiGC+9aLM9J3xi9WGCKIXDxS0spkQ
cYGzHwn0Gao78juGLX8Sjs72CQ8BCH13P3/d4Pf1GW9qK3/5wmCn/yY89Ynypj/RyPf/Ac24tsI/
mE63ldFDGHO7Hwf4F+q6sgcbxj5/ySTK6QpbxhjH5hviJFGSzlPtBSyIXRnbdVCDxVBDdNleeiD4
mwUnK/bKIWseRdagMQYEMUi/E4Iipq6T6uYuxXj8N7RPx/Bj+T0PfNNWftUTlmMgtM/PJSceVgX
i8PpihgMBS1zXZVQSQC7C9VfV4bkYKMnCVvhs+EnBZkbCbNKFBRsu5FisQXOKG8P9pKpAbkRbEOUQ
vasRjN42CitXOZ8+cqe9hTlK80ZhVtr/JbtVkB9gkfdOMfyMLnDRQPEp1Z4J8tSsc9EdOy8yXKcR
ojg/ehIqerLB4UlfBg/FKQDFvVSXe74rn+FC21pImL1WY89Y306DXrUy4w0fshlnMex0MJA0Wqiml
guKmVwW8JEW0R+HfGuprAfKhkMnwJp82SCGJoRjgEPq0CoZE49oKWY4E3NGqhgeakiNUOYKfqsJM
z8KgpQESM8g4R4U9LCRALbZUmiq9eJglHKQo+nu5B48ytBicJ8qBFGwkU87cnairUQXhduGBuMVzw
QeJ9OYfDISznWNAVFLRvIUOEL+bds6f1WE3+JphYw8FYOGt1fQSEPaRb32Onu7Al+KR6uXTrODbd
CAfwYjRa6uRg8dCY0xdijskXw9frH07xhRqb4kXtT3KyrwBZRSpqSQ7+Zk88M9L1UqlPFGGo934v
lDJz7sVZp/VF46cikOR1/0XOvJt/20kuG84rFnrE/j8djf/AZHn8PznkOfLfbqbKS0mhZxLYH6FX
2L5molqhZyVCmVKZdzBeSfJqoPC2QqpkN6Q632DfZEW9EppbObL6JqKr3URBaieijvg16oGKSR6Y
hAesQBv0BQsjilUZyF5HvKUFw7Bbmtei9DwmhpxD2w3i2gaQzQEgyA0rMWjAShifyemH710LXFc
WyLNHhr14kspso8IeEy5Lou2LECI+B/pLPnCopEtXzVZ4V0pJhZggyLIpVoQ+TRsMgorW1VTa2vN
DlKGX3SCQ+sLEkbGy5UpMUjJGE2A4zxvlyTb5iGseC4DXoZKSvm0uFtiZoU3312ijsbce15aMD7A
K2a+9MgkHv+n/9c1/SELpS/tt97Bnr01owNrIkjtsCn9wffq55V2N+mcqVKpVKJSOUPZpvVgh5sU
ILjSitzc+xt/VlvZo02A0VYBw52NrWlnhKLMipJzKMHl7wKsrclM3OVD5pai4jAVZo4aTpgkxngl

aZCjE1GCFJwd000sihTDWT9WsVg4Sa+ElBM5ooQzOzJy8y8bCkNF/k/U1RUJ01ptV7eqoVY9g61N
Zs9lMRrGlIrb2qNGEz9oi211EU04/1+0yjqAzNnAEIJOZaoX4AT4zwFj3mS5fPUTgOBjMe9oyEzV
L5LVAwbvHPCzLFMy2SoRVUqC9Lvs0hD0X9bfn8pm0HfMCOpSow3I5fKsbF5tL9o9aFu8Lh4IxfqC
Rf6nK59q0/vte6eYyTlu0t/YxN4X8olWvbeTbIUj5HtXKR1lf20/JclRarOVpcqFodwyrbtQQg0e
93BdwP5od2L5BKs3gzHYoKSEDU3ZSZyiIjZcjCeJhqDVCyJEheqLww0z1WoqlOFRuVo4wCHoemrz
OVEQLOF6B+QP3iSBWHklQD+xORs5AHUFZCqqb0Uq1pb1Z7UgMYWoTEbn6X2V9LOqr+jwJjiKfWue
+sZSW6s5Y0y7U8iMTsKqdFu0vulXgibSTbLkqLaoRhf4S+ehmKmt+wL60bS/eZEFZLyMzpgleoU9
DfEFAFBjDRRffK83x5vJmh2ARK1wwF43U21gUadS7F9ZUTn3Zs9w/zc0f+Np1EW4f/2Q/+k6hdhB
xRPx76w/fuGevTXIca8wLbZk/51waFv5jLV0Psyj5aJiaIyUyhWSnyQ81kqkmWIS80AxSak5i2Oi
02XtlyEdYEVw+CK6kKES+nw01BG7mqv4ogMUt15TrVvUulGSqhMryTYP/0cLneCCDHI7y/tuY/Sh
Wvk/JgkgnrDUPzCgO7zDkcr3rFmZQqtbcpWXhmGrOQV8AWAB1I3uypPBXXS3pTGfeVSh2erzV75L
qkVaMkKEWKMLM5LzYvKYjx9bV5Y2uNxtTijS5MbWqPehQmSsEAPdXxGFjelAJ+jinACDZqKLMjj8
Xz+GuxeZAI5HXIYyXtQN84L6AOaOay3wbXqtwX1/ED/v2n0iS+jy/g4nXf9lPxp3XBXPetPyF94e
ljzlj/1/RyZqNkwqBwxUNFxNYgCUd/mqbGo46phrjBcACEpQEepWLPd6YyHG2IAQ16W9L7hN7NRm
tD1XjESKwZHajS6lU9UVN4kYBKimdVVRnYFsV0R0i8p3qtMALxLLww+SJE2HhaHQZlKWZCDeE8Us
atZlj5YVvkqNFULyVg4RhIqXiS0UwAPnOXTlgqxAW5A2wWFLFO3aBABWEq3ViliqdaOwQYk3lZyc1
NFCoa9qxErsh2mPEdVgdVGAmz02c0WPb7Roc3BkLZfTgmFLZKFCITzMeGw57QwA6hgM7vgeEC5P
TYgGSD4Xw339wvGrpZOFZcUIa659DIQainIh361abrx2kyMeMk2owBin6Kjppf9lMiBSpyoHvG+h
Ta6NQosXCmxvq52zo503b741H/dlc3t8PU8lORREB1kw0BJGfDIJDCkQAPgJ8VbiClAndxY00NuO
vwlUzVqM9SzQII2k6ZXLHFIEw9qC4ACaEjUcwJUMAhhrZCCBiNOzK1An5fMWT0NyG9IZ5VDX5LZR
bFMUefB+FxZkjuSthsMHqGKjKfI+MQEuG0KjDxNcEiBSWEpfq/tsr9DM3kIOhFRdlCup8rFWgtfi
UfKVMx90/DfvRhkTsrfR6lGYb8/ZnVNeqHIRU5chkOQhrzJYDdcDjqCvABEYsCmhDlex2RIULJK+
TZrqjwye6XBJQr1cnycpRqhFUEKQ+nRJMB57eFxmACQAs/fzIelnmDdaEm8FQeCfLvaM9Yn+oi0u
fts75sBLHyel02p3ffeTxRx85d+HScMS7PhutsB0CKQS8MHYXKlIWtIYx0gb/Npmq0MTOKPiYJ
YvCv7XNsXbiQFo7VTtyXUHsg0DiNr4FX1OzJtiwQN9BXfiyX3BZSyjawqPbEnImpM8k3Ke0KhK7j
mtEOOzx5mE0/DoKZ8riyJpbcisK+kQWlOKqpZROqsiS3g8XR6CZjJm4f3uqolV1AKlidxgmVluttG
Pj2aYTU3c8pvYAA+lsyIYie0NPOeXnYeIpkUpxrGQvM3VNqAdsb2/IQIWpI5EQkzuGGpLIFcoy+4
rCHtpVMeelguE2rB0Mpg8Gvdy5R++9Jo7jw2maOE/lrrx035Rptenfuv79oz1qbNVY7cV8T97mm
klMKhKpZS6lvcJDsvG3f4wG8Ng2bPAYKd4YZR2wCpoCeeQI2t1qzegbr8nJpBg8aDKC/U1LEyyrj
DSTIW8WjRZMLzoDzqKPDuz9GKNffs+SeHJ5NzZiY9X4a9AC9kyh0q6RyqiysAvg1kQhuLYWrdVRQ
YwCuivIkkGeIjBdlMKYvreJPGTWRLi9fWfMY8ptDa0Wrnq5s7aFeFhrN8lQ6AGFKdYcZKLnw79b3e
qKrAbuHobe0R7DZpaCRxkjfXx9ne62tHZQdHJsJesB0vWdzYtskJb8zHXLBAOXOa+hBKsBRQZ6Su
hnhQLmXZuG54/HsWkPY2PpzOMPfRXf60btv/49Tyns6KZ3/oo8ns7jeULybfXpVkfSRf08bPbVat
XS0SMHh8PR0BsHoYEFaAoJ4V3BXAggup2T0BrwQc/IC3k2hrAvb65Ssz4nFC/sqJWqG9Fs2kdEsL
TnMAytlG6p3FTgdTC8SEa3xYiNKCEHhS3xpnmyOXyKYVe4nxSYwZGhgQR8wlKRVXkdmASlBQMvpw
tAGCXLQAAeDqRPM6LNgp/ZClssVWwoypkKA22YyuAMreRuGEoIS77B0LtIW8a62GSq0i/CYFSpMT
KHYYthDRRC4CxtvNuekhkzP9DnsLeRrcjqkkoBlgjldwme+Jsyr+qU5DhZKHdaPqdfitsbE+QQf2H6
VaY47fljcyjmik+OUU2dDZa198SLiHhS7GzYwCmz3ts8cD6003mpdvinJzFft27f8Z/f9/o8GQX
ddNjKbz+9XGpIe4W907Te9+lm/iu3ng6HKOSuMJ3mQ2fwEyrqGWeI/GIR3g+Zc44HJ2iTsdrU4Y
djW/q17OE49Br106L0zWEzh3TxDaxwfnWDRr8qohhtmkiZlMMoU71w4WlAXD3rqWqDJWmGcsd0Db
TV7MiW2Vn0FN0cqrEcVPIiN3SIKyAJGIdImE89XSRTMmjvmHoPSvX7GFqNXQyLN5EY3tm1FUAfHg
reMAYRXKYkQwSnan6LgpK0wpvGitgfuooQWc58WTX00rBR1WmEX6rsNag0xdWpeeazxdEpNq3xw
LNFExxPBKuKcEim6plB08Pwh+1fod65x+ikDeuxx67j05c/yKqNxb4I7B3HWN2dyy5e615lPqdNd
7cKlApyGWWawwG25wr53LslY9evHzha9xc8bFAGNxbCNy0tHhEcmr8jU78ix96zqzKl3tj3aVesm
iXRM24on+TwYVe5p+tob7juU6uXqk8cf7M6footuwk7MsfNceLe5Iqfl4QCoVbm7x8icI0oBx7wN
WtDV5UmQZU43YhSiX5VqY2DJqhmjRtqKkkMQSVZFTiXBB9Eie4I9QtHVNADmkWksFmTAU1IaRm+S
v7VhR6ObxFOUiG2FHkkcJQqr2rIVy9Fn5fqEoLBKRvaWIJCgtGhqKXGIuEs47ymKQ95gzGmKkqsK
Y6nZG0SQacacBFJHzG8lqA9znM5bSZNre3pXKOW+X7iNncYW/InyegWrnKIEyIfL+gQnPkqPxyYL
KHwz2h0sH9wrjB+idNn5Hke3+8kGqlxOSIEeGQYKsbpUt5tjCSx5+YO0r0P3m3n/LnGJDRfvdzdz
/8optuWkv9dN0b7mQA+id8bz75a9+pSohiZbzl5kf64bf92LNYLT9/+qyGqS3X+Ox4GH8zNDC7V3
3d2/sWW0+xmLsjTiYblmm1xmOM2QXSWNDFi3GQhwiQh8TA9lqSRzq8YFJa21xTPVUB6cczXiQshk
Tv5JnuaUrPVV8LZkNlrE7CUiWMhQcOEGFDF3X7/CMUDBESJkoSQ96fQ1rOATPounJOHbDX2j71EO
08cj+NtzeKvSFACHigDJhzbppz1dwsqTE4AWJo+KBibdsGdG5sytbX4kyVHjnWH+ldLfwpmnchN
TNzGY5eo+vbTAAS2OVwmFbV7xNUQwARDLg+2QXazLOiJQCD2HMMAXhh0AKgAELjNRHu+m1W1Ipdg
sluX+RoKEsgYEKWYBqz+UaCytrc/MH2OA79mA83Me/fc19d9/zkm5o+9jYEGp/HoLwZ+0Sfh6BIR
LPFyRrWpaUnvjgb8N8hq5t7xw/evQPHTOxhF0vVGRfWap7nuzdcrzjx7xQfA6NZbFz+La+vTnrN6
qyaqKhgYyyy5YpBTbV0lDwQtXGUTOHUtjRUEUB2HN+HPa2KRix5yg0aNBalUKKPIKHUhiitwjvNe

FNJRmM2RM12YgdCrFWhVfYwCB4vixTKShGoTBlerlZ1VWKRMAdy5ieJkWXGxPPNppZZX1qtKZCUY
lhEmlARDoTDDf01BCM6/zaKhVKRY4+fIowJB7zNWNwn6/DyxXY0Y9oPBjyZmYp6tTZKFGiSM75mi
CvgXx6wtFEp98TnuHBzppuxVmQ7wnjjwSAYWLwnP260WjM01xzGclCMZ4klw3Hyeseve/eq7bHQp
Uh+OvPRRC+FwY//RWl7MmGmuq96UmeNROQg+rxZBJCsltkQ6kfPnIkl2bG3asb269ib8spnkuNal
WptWGWlavS5vqqFDuoUGTf5dNmpyv9UWkPpirPtYwpAEIYdlWOaShpSFKTsKrNI4j5VF4/NRBHeJ
sKusBlklDZlt5iNOVRyU9qZ2oiRvqvjaaihGlUyWJvZbKBCTaentBQ1y0hLobkASPUbY+pf5y2aQ
wz0UsDobGlveoVt1M2IUXRqnpPWmdSNWslgMEWBfnlrUlanm9QO05SjCiADTSeRMJgOOb8v1ht0J
CNEd65Pwxoro6KLnVUXEk2AdzFWnWRAG7xhxw57KyeoWNXH+dz7mej7UtrDM8CybdV8tmwu14uX0
wMJzdeaNyGjmMXLd59L1xcm49SesV2f3j69KOP7CSHltYajWaC6rJseNiDeSOJg/Gest7jtIU5lq
OLv5/tZUXLQWWwadJhj5B3ff8FR44d7fg5f2t1fXUHTTNT2gquKbOujPMXjG+fvRm8HOQXe1CL63
dpsdbg05vqPhLKwm2q/JOujIPC0Boy35qAlIwidVWocmJctfEILT07maZqvA2eADkr2PV5iQoiys
xyZBcK0gYSwu0sPwPdYwTPmEIMUzVrK2GqSHMOahjvB6eO+8Pum043GFPVphTHW0cFui8aQzyNGK4
+JgjeSYttA3omKMQLwIRsDqsCYtMnn8yoS4R0M/VXglfHVtHwJywdspIt2kz0vPxc0r+yNAXUYeK
OMRgNp1lk2RzqBST3OOYscyoIZESAIjCaGgSKzG/VaVtnyaqaVtYuuUTx65Ei70+qWur3RMD6AX7
++tXN2cXH+o2ykfc0lRL71SoLwvTD4GY95p9VgU0+R6MWXqUoqPIXydtmFJIn/gSPKWzjkXV9cXr
q32Zw7zXlQgkJQpVighblF0rB/ma675ghVSuxTPUeqnwmHbBfXLus+oAKxT2ddsft25dmm3yeCgE
KhRuFNIWhiG5qexdDvY9sxFbMfWARsrA4vYoyUCcLYBfiC81CwF1qObqHo/JEi7QVTlWcKdFBhkW
1wRDMq+grI4LS0NB2LE60bjYKlGj+sIpR0OqgWY+LQI3YAVWi2Cni7Ab9nf8CbGBtCDuTh700RaS
AvDaAZy9cWjPszwAgA+Jxfyp8FtDyl+hy5qE7HQ855axIOG55L584+xp56wt+3NXGaIYrpqfSmzd
grVrPMBomqWXQ8Z9s108HJ649zqGv5ju+eHA66r+912ke2d7btKGwZiFJAEa6whiI13/Osz4LImB
fHqEO33XG3cfPLTmZorWCnN9JAOWlZfOM0ST8SBpM72UBv4K28US4VjFbL/e7haPxK2/UcXyZbUt
q3vCBh6VonpI3V84JPvbixTSevAjO/GqJGziUhsCUkPapNk6nI2xSYQKyqu0AZae5fUw+KUWadJY
n0hI3tKQ4l0Ji6+aJSII9CyT1h0ArdY+sdgT9LFKuIAGaaxbMWi+qt0m74K4MG02IYPK0I13RDSa
mbCkYo9DMCeDLks0iPWFe+VW6uIolefVK6tHqJdrbWyEUeC090kKFMxYmPZsgcRWFCUWYLler09i
rZHJX4jqdgl5MxzC0v0XDU47vEaUgdGkD8c8+nLkcwrigVDGkMoDEAERLKRgZIA6Jg5KactPPf00
aw3ij4zupNN5xY7LS7zVEwusmwzbtzfvlcqLspfx3SyPOiDVB+J5nfRYcIvAr4r6Uu+POBwyZCI
lHogiOKqymYsn6/X4UhFE7To1PstF+yLacRw7s3/+LpVLpz3kfnliSY5I04ReaVar6plCGwsC2d1
ZpdXtTVNxxGsZUo2WiEppLMwoW8V2KDFBVRSh+QNZMvhGhqjklRLMUy4RA9oWH1leCwwBB0H+Ldk
4QKpZBy9QV21A9TMU+KFM8pGdNs0wXsqyZl5xtaTICNmWfiCSzQydpqxahlWKAkKuzXzbZPDIji
5C2TKu8Pj5clIQArtkoTyvSuY4xu02MP7Ds7b6Ia2ubdI25/u2V5b80ZDwOJR7mMv71Fjcr7YZ0v
LCCuV9DKZPOOqdCx46EW5gQ4pjQNoP+j1LKN15R+H74rCH9tiAy5bl1liuVemdpXf89KprcnOLs4
fjOJpvd/sW0gd48XyhyB852POsz3SzfUYFmO741KcfwsrMvfylJzEer61WW/Im6LpEUSxTMMhPK4Q
j5jxzyvXuZgsC6S/NzD4eTSYk391dHk0lulMVsxGxbVqzl+TY1cDlG7HPJdvHyE6t6K1QVYETkh
qZu30fmV700++VcScKlyucwZa05i2Vq1KmoYcKnxVbFmicgYyMxItaNiB30cX1pL0dikuGRjKpwY
Sp91QEa7vtmWRGvCb1XfBRAemEn0+nwQCA0KkDDBhGYcgo3G7qL3V19sDBJKIzZ86pMTa3QJ2tC0
J0FoyGVKutiIBAB2Tq4E7OoC2EiHbMPDCbLBo9fTbq1SpVYXzuMSG1DdzNBh2yags8mcEmCIUEv
Z4kKp0Ijc8A9IsNqQ84bhZNh6acZhBqsQ3bddqr10sXbiwbnOU8oLl5QMr+cRYG3e3xiv7DsZTgv
A9z/pMJrBQAcDilcodgUgYTb2BjJ3xooTRYi4VVC0wligMs3ASNiaTsMI79sJkEsxxrjXmhfX742
H3/xn2Ox2M0YWRMry8Z1Cx4JGfK9GftYvUC0ZCGDadppkWiaZfU835KyFjSnqIPJL+KYkCQKbxu1
N2Ble3izlvzdSMK0jWUs7dkiHneRwKpgLct3UPVFeYMY8KD6q5kIS3MFFgfhSf5Kupyc0wApjFMx
oapbGY0swazSmz4pPH/KazrZKno4jGz1/bWkFNrR2q1hY1PwXXMNICFIRwBJybIsqplktULBWE7B
teFe2biQwcpJSvLtI4GMim0Vg5RpvKGVZWVvhcqfAtscXJe0GJHr3YQBTpc4KAei4nTCGnmXCSR5
MQf/dcbefAKi8U7Hy+OGeb5jz/+dlxGJnJoGlogvA9z/pMZqrBaKBrsLsR8Wy30sWNKU0oG4EZx5
BvzYocnpb4+6p60Db/9g5elacchxNXmz3B2/m3hVHgjQZDA4um3qgKq3+vH9OFC+eoeuIkOyp9fg
k9UezRGfPdkXvDLjAGl8qg9RQ4obRxYCI2hrMNva1VPMUZ53i8wUD5fAqsYM9lujmZDSXMiNqKLt
QwdHtIHHReFnCa4qO7grFVvatUVXYz6fIqAxd+KOCWU8EKyWVkOmYXEJglM66ymVCiYcOWOGB8Dr
zD3ffdl1BAGCB4j0F61t5aJdNvCAAC00Mls6rYIKwiRwgTgr122ylyGz4bs0nlWkVC24Bz1jx7vW
LjABvuFm+oHpb86QJlW77Q8CDMJnamvEhjCMLPuvXVKy8xUqJxEcg5ZDby+Tx/5MC0rVnz852CmS
Yub3ZDWMdXt5OD+/YkLeco9Y30mol9DdvAPWpUox7AHy168lOkqJRNjSROEgiKf9kFvhe51jNYF9
EZflzm5+ZAKL/5p/7H6C9+4pu/2/L8ZWeSwCKJogoWth1P2GBrMkN6aWONrj16NRUKJV0dTVWZSN
QZwesVS86LXBmMiQILFLtQWGRp8qTQfFUMigY8UoAxtwk5hXUOKRsizwjisYx/Hq5uUDi5KL8Xgm
0QvNUbos1q5jwF6jcUtDDNxnoiRlWbI4SqooSXCv+xqUPLJ/HjyqaSkL4G7tgiUyx+EstKlGsEJ
3hSIwV14YBBY/fHxVg/G7U2yCTjR7Keb7vcrjbFLig4ZSptXGRQ+EJlfiLHI4WqN/bJL9Yo+E4Ir
9iUzTYIc+v0JFjJlW4HisAHjZOAvtazQ00zkvmLjFHoacEBNhvw8n//SmeF5NuVcw8q5bs71PG
s84vw2aBlpWDe6/QvZnrE+A4bKoY9SZ9Pom5tuvD771KcftFUoKugigxeQdE6kisnxL9sqG2p2gL
9vGNL7oMv8WEU3QRu2rOA3v/f98d//4jvfbNvRbY7jvCxG/qhnMhvNBv2WrtV2qIc51qmgN/VEH
yih/K27qeYvyX8DGTILMMVEliikeOxwGZib+OvWXCxmrfqmAV9lo0XG/LqBmGucEmgTxRsFhdk7

Mxdn1z2NmSAXUlp2Eo8IVrU+Kwn8M0DxsLFjome2SYHkCLTBuipXAI8JxCb3qFoU7vr7ZVYrOLK6
tbVbfc8QkKY/bMkx5Z1ZrM5eL+otpaZyMtcuiL90I118gCqtTrlGtdpoVmhRbml6hczLORNgAVpD
Ebvml5Ir2R9/NS1cZGMR94gNobF+TzeRjCdZOpTKN9I1EUtHM59AYM0cLUEiKPGD3dspzHxRgQau
dGnM7XS8awH80IwveM9elMxEXASS2oRAimjVkh54aTV2WqjyceK4WxarZ4/Ae0+YjPGEdVzEqP8z
kuWqYzQXok9A0//v7Zzvaf/Nb8Qf/wzu/wvXy9zgZvQDvF2OQnI2qUqnQ5c11Wlo5QJ7pSENDMk
ZDgHnC6YQdHyRsGRTNcX5eaKN213YubtDahbNUK9apXKiK2jg4IKAGgLYJQBP4XMgtB/aaKKslwA
vH0I0pEWqxXqXOnq2stG84vOaETnuRuYinqBDQOodzKTSW/uI+vr8jPmw7Ik+4Rq01FcQ7rPHhagD
LSGZULvk/RtwVkkY22zXn/gw88wBtXoIbRER2gIMaGkC/4nNPnKefZki97HKaif5pEQ7rqyGGan1
8W0eTJJOFUoq3mh2NsPNACSi3dEiKXBkYHCFahSEA4LahMwueKXH/icwZB2zovl8mIleQdh0Ixf
E5cI9570o8t5Akk3FsFaqpW6yknNfwRm3w+Tp7xvp0Ht1OS8klxPFu9TWbAnCm7HxXeGEs0oTXjk
FzvCavgsHygr2fv54H7S2mPFLdXvns4zX/128lf/3Tb39xEazvdDzvxXmnIrlXkRfpluYWtdrbtD
i/T6F+NBBDATOmNKF8bezp7FEoMo8oQxerTVrktZ6vzEnYF29e4vA1IYsNw7UyYUmwDTWHakMdyM
Jhn1fgsLjAz/cVpy4bqzu3rHiJABaAZgyp9o1Ra1BSKFDYvkzJpK+YKcxolrfL8DgKU+Yub6VYH0
gzXKhZV30fLYWCQhsG9+e2j/81hYHKU0v1fTQa7rABJfJtNwVOOSCX71FzYZk9LW8q7NURzcNURc
/RcNCjcRsshyFlux3J2UvlulScwdTR2lmlenM/TdCXlhlCzlGzkSxiKM7l/KIQhSecCiD3nIiItC
nQRZS085UF3gwv86ZRQy084Hs9tBw3cXKSx2Zxb5BZtrtnrE/nMRwM6R8OnGdXdnKepC0kYEDTrG
WZeYR/wx6STROxPpaw+0KoBu+LvBZV47/+mX9Fb3jPf33Smd/w7343+fOfeOuNwbj3p26ce7Pr5U
xoiKJV8/izJ6heb/Ki9AjJsJHtQh8F/ICZWAd8Too0DFGB5+SpcfAoh6dzNN7aYENiTwttGDY29B
kB28uGfXL9ApkFDu843Iu3L7GHKZBdrJDNxtlsLpJdW6D4ifugPkCcqHE4WCCjVKC0kFd6rdYiG2
tBGASnE0GJsDzE4sGlqAVQRqqkM+Ta4VFRqBKvmgpiMUvURnb6zCP0xBNPCPughzxy3BLuJ3g95N
LNWoXm2EiLeU9sPs/hcLm5T3qb7a1N2tle5/cHGV1HZlsdYYtgYyzVqN3uSP5pc2oh/EtCajdr6u
+OJeARIMwcl59x1GJMghe/ryhDA5DtsH0lbDLu80eMTJt3iTAIdyyrtBWzd7XtRpaBnYLTBxCE7x
nr056zfk6Fm90vs71WFJcsXUzJPDxbSfxf4uvncnnlVQTgrw3zswxlennDe/979qvfVfNb5haqv8
HO+Dtsjm/z7L2gpH5p9QIdOXhMwQM1cTZG3UgDhSyINoP825rIVI/jIegdsTfskFercxjXffxsHP
RUlZcNL1ebF4+Z9DbJqJfFs4oGS7/HMZ5DR9L0fYar78BmWzAKYD7RTbsakExKYp0JG8UNoR+EK
Krdo4tcCsF5ECPVQbdrxDBUqp3maoAq3smLaf+aEB3c64Ko2+wUSZ8HuCCi3xtjVqTFpYwQZRH9z
ihYrEoEpsQ1+7trNM259ZQns+AEMGDAC+VoLLQzuGIAIyHGEps98YCxCjyvYqQx2vFPxESwygdRL
Ghf8N/fm8rGJM05LTEgapgZoch+HL47+zGHOWO/ULxLEcgGxSHEUC6KdpDU4LwPWN9Rgz1ymLIrp
rck77XnjZNs03+oyH0nfDi7DiOmxpaeEYd6YyA+/Md7/rPn8x++R03f8/S8vyaOez9WNGUMVhwp8
6coqWFA5yv2WpTknIppqVAdmZlUURNF7SIUppYwLhhZj/PZntunSelgJNXjomoEJoEFzqGlbURkNw
/QBL0eoyF7WalzOhxRwoZiLy6zyZyp4felSnn039h7QCJRCIZDATAAAG/4saCgpDdLut0jfEu4Tl
vDqLUE5HTXSLONRkyFO+nBuz5BA/b28Jg5zhNtK6X9izWq1RdkOF9U1vmXeZ9zcI4MeulVNI5SbtK
CKNJGEhA65OCKZQqlEo1FfvKXNoe3mliqHsgUa7/Q4rG0qKUnMu05Q1BurMUf2tn6pzuFxpnbZu
abvoD8R6GTDfodz8chOfsSc0tst1TruNvm2Y0icddAKEzIQgv79szlqelwGQ5T1I732WImDrXJ3
vdKbEDP389y7LP0apVrZ3X/tvf+Uff+/t/55PJb3znK3/adtwemfYvjIPQhB7OPffeRjff9DVSzR
Tdc6EzgidjAxAwfaJyRkcBHsxMqXtD6dGucTjg84IfsnkFoeCJ47CjWfo2z8vFO+yBMafJao4Hi+
HcNTMDyspFqYaari3AfeUVDen3GloAy/R8AUXEwi0caQ4nxQmcaSoXVTDXFWA8B51T3jxQAV9dPU
utrXUqCoNGRziW9++/WmCRCHHjsEcebySo8IIRMmxv0JgzgU57nSbg7w5j0REqeKCx4Qe/vtCYo9
G4L/WsMhuvbXo0gUhWash8MQAcaM1gsCEIjXk+4/Z5ji/EMI6KABad8TWljuuY41EabBXvufHYMO
1ThVltft6L+qadS4NBN/NKjWxKEL5nrE9rNfizx+Cyf4yPVEu4fNZryPiSem7f/b5b4//0PV/9Hx
996P6ak8v9SjPyZtrWNp099ygdPnxMeJuUjkwqvSFDqtGZnmLJZtoyJsbmIFRlW6JdahQ4fE0KZM
mstxpbS1Gk0uz0Arh3ldJ4xsaKcNkUPL5TIY2EwV+Rk0nBC+0ZiSMxtYNcNSQFrdRarNObJLIY5q
wqJ2BGtJnYUDutDWqdfZg9ZlHGyxb37SeH/z0ebFDY3SiV6VSc14QWONBX4Ac/cFEvDDCZIO9Hw
ptyHs9NyeFI/R80YPNFRZOPBxQsVynHuex6XBCIUjPefOAYDYG2NNI6eKAdQKbgzD88+snjmVYqZ
H5xZKAs9TEchoZ0WTT8bw7eUM7xTlWEI87KQSwzFx5RhD+rF3XX54Z6xdvY7sh8T9kreQR4HU//L
tflPm+4z9+dDK/sPQzvI7+YPPSw5Fh5ujcuXPUGwwo0WGL4IEF25vNSNmmcolixFrdTSQZzUTYU5
y8Q041T16jxh63QnaRF2genpQ9ZyFHFhu06S15jV2uYlNhgDXtK0m2kUwEyd2KNVuGIi83Zvqjmv
3B1BX1qfBVrEjIUb0drj4hQwVoL+WqdYkcoI+KcbjmvuuoPLcP2ScNgURiQ9vaOE/tlgu2Q3DDY
fPo/RqoYcUrLKLJfzckCRIujJFssiGRKzQYF+Zczhs34DJYFwoobGE7SPMgXKiMIQAw4IhYT6BR
SVS/qao24nK4bjnuW693nl+duyJN4yE/bN8TgznVKmCilbN4z1ueCiV/hWac4Xix45Fi3/uYPfN
Hne/uvfDCYm5v/wUNHr729u3U6GfGcfetBu2k0SYT4SeFzSWvVqGpsNpWNnF6LCESZuwLNltJwnc
612mwoefaeFh76MDnN10+KISivGKvQFjBDzUBIQkyWzeps8GzTudup5k02ZYRM0E/dVVtP9OQQBr
1jzjsNr0hesUo5fgxa29Tt7ojHxchafzimdm9IF1c36NyZs3SZo4txyFuQU1N6NuyJS8U639+a6N
lg0Ait3UJtCugygRG0o4TP2RbxKxgseqqZSFdkwt0cTgK+llDybBTNWPyAMaX4lgFMG/Y7VjyJzG
p9MSiUyk9YpcUPc+TySHX/NUGURNzj7ewKgnACQfiesT7HjNbnPMkrFNI5bUxZXTLb3z/F3+D06
BTrTbeubDv2M7W+nkKOOS8685PsMFGSr/0CoCB4IZFqNmZfBOMqRZqpom79XNBx41wlGikYmLJYc
mBsQqXk6NG5TSAQdgTxTLVYLhsRsaTk3bl1RXo30in4lact7JHB9AjBmRPG2oEkeL+tryP4HEXAQ

Mxav6vvHCYw/CKMOr3OE9sdXu0021Tqzdgj5sJVzEZEaFSbkPYOQtF0ycYD4SdcAJML98FqB9sba
1Rt901DNFgaAKtrSIYMB84+eiAgwjQ67d6WxJDp3zy2yoeXh3Y9jaMPH7eqMZNfYOXHLcx/N4u
D22M534vbF9HMQhPPm4+8Z63PFSIH99fIFnTd+FnriSzje9ksfyZKwfbZWrf9Rc24xbm+e5hzSpw
c+czuNACAxDGZkJPuOROCaVfVepkYt5Wpoik9DK83BEwtPcCSFJSERhZq7o2Z01aSOfrfJV1SxFc
WMyi40dDEFxrmraDfNC5S2bKbhh4meFEowABD0JQwltASIYZd4E2IDrC1QfeUE9VqXKACOMWNUGM
7yvqPU4Lx1zK8L2TD8AueX7ImrjTpVKzUJoXGZBQ6BPc8RoH6vPxKh6gHnrJDYxJW5/Lt8zqX5uQ
XK511R+XN0bXrE6fCoAQ8kRSCH+9KYW4pKtfnVQqX2Cf6gH+qPovMb5x+PFUG4S+2NU6qfLLIih1
I32DPWZ7+hIjd1BAH0uetuX/ndv/wlnbvW3M+p2XCzXJ3LsLgs15RRvFOPPKATUp4wUYIu4lmV7O
OVVUmtuQqNVVRndRFFsUukqncLI9f0p8pYU6XjCg5zGZHB/ZMnujQu50yzma6NDMNRzqh41qGJhZ
9YQPrRSKnaSftHLWwYbSDiUgnN7b9GcMYLh17I3nKb8808G6BFBC6zF+bn6cC+FZrjXDsKe3yens
z/5iyDCp5BKytLVODn5zjPTPhix+OAPR5U01NBWLkyaJ6IfEg+n6Pmwj5+fUKVEufuSBNE9Q4RvE
vD7hZxPpoWCvmwUCysufnix51C/c/4nA+yoY6RDYTD7RlBOFQAQCWBacTbnrE+OyzyCvjhDH8oOR
ua4qgmYqTqc9Wf6B9Cor6g4473/SB1uh2q1eY8jolNz4H6WUzV+UPU3Vmny+eeoDBT09zGFWHqVF
TrypAVA9YZKaHhKTWoCvoYyEPCoKdq5YKMcjR7g2JSlN6k5KxTyLDNnC+jb0q8CrhaGSBIDfmKRy
Y9Vw6jOVsFRg6p4H2mGxsGA/ZoAlO+8VLKlascjpucP+dp/5EbqFywqVLik5sEZCdD8j001dmw6r
UV8dz97VMCOVxZXqGy77DHLXEYHAI MED15JNqskeTjwDgj7B1wnowUIWdhE8hTpcI5s8v/Zm/tuJ
6Q1zm2m/E1BLlc7oKbK/ydnSu+f9jeuvfcQ5/qfQqVZGm5nkGqM0Lo/SSC8GdGd/V53bp5zf/9n5
/58rI+Tj/2qG1b1tIkToqVQRHTCoaNjBeh7c6rIhZvDm5tiaLuDi/cCWnokGJD1BQNivxMkZTHbJ
giXcyLGARfBMJuDkGtxJpJcohtC6TR01QvjrSHppqrQD4Judi08izeNhUDn01OCjdaIgz+hozDGR
IBwHBj6b2qIfaYF3nz+MvIbR6kkMNxG4DBUpUyTiUmwy4NO5dkHE4KUJzHjgdtGuycYsPNU6M8T8
VKTYyabHAIZxw5jAJpDnngOmYPHuvidQJQBv+OX5mXgXTPL1B14YAQpTU5x071h8I6mUZBWihWB3
6hyPlG91HDzf8Nh94PbF1+olMsN5NGfpytb3TIYcP8un/7vufUurZp7/gnHb/67TcDuTT7/vd+4H
XClgeeJtMwXmd2XsTmcFM8GW8U8qVxqdQUWYiw35JV6JUqlGcvnzQPUEfxT5IHPnQugjUz1Kjbz
JrqhgvUEBStElq8FzC4ikAHx4QEz78WoR4U6Zk8aCkhmUAvpBRN4TKgu3V8heAQKLgA+EpeBcIbk
FxINHeV0SUVThucr6XssGgGJRbPET5xRM0AfaIc8+QnwnCN1SEcYUF9Dx76+LF4nEOBTP0M/N8fc
VSkT8ne12nzF5tItuMi49qpZLzom6QggEJA/dUQoA+5K2mTNqMh33y+fe50hyZbiDAjmg31bj5cp
fv/X382o+k5H6stXb+DH++XjzYiXP5Uma6E7JjVU1/rh17xvolHr/2Ha9gW7IyP5czfzf7XpOx9w
TQ3/Rz3iHTtBqmaXyGw80JG9Y856n7eHEUHDmcoZI6JkXwhcGAzto5MtwCtZ64lZqtTfZCFCHOSi
dfe5dhgU9rXTL5I2LHqYymiYLP1A0RhGaZgi9mGMBGglJXdxV88Qr2c8xuo4iGdqtpCpZXhuARhg
IkEGseJ3NK8h0rXWjN0igCxnYdxYM3iBqciRE9pBKZTxMTraJMzpsaiuwTVlncg3WElt+fQ1W3Qj
bno6pQB82rh19X/bIkr7ztWBGmLc7QSGV/IKih1LHHAUxxDlmdhsTwuRSglw/SdMwanmefzfvcH
8Txsmdw53Vi5zW9NPJkC6Wylk/7AsBW9DnNKR5YM9Yny9HoVAAdaV121aedTWxxZtaZc6nvp5z4D
GvwUfYoga84E+zN3o0i6030AXfLTgOoclByUhyMJBvb557hNZP3S1h4Lmz91KZw8irt1xLOUYTYI
ibQ01pp6SZ8qam4k8SJXZACA3lQ4U0LAP+OFMF4EQhoPB76NsolJPaKIB8gjFJ1pugf8rvwfmeOd
GM9LoKreQcDcmHxXmDWSIcCbFZ5diryct4IDnKRCoZqQVlSzroCP7CinjT8DP+bVURo3mgYalJaJ
2ELcEde+WmfKaYowJo8YAWxiqUtLCXIVM3fs6V7/P82ow/HHrNjtdYOJlfqhu8GWS9nbWe7Tr3WY
bxV3yeO6PJeJ13rGE47IVRMeh2Wp0MDIY2581uYtCL3/buPWN9vhzv+OW/F4KWD/z4Wwccmqamoj
HNceibY4+6wY85/v2N/HiQF00ly9Kh49qWUA5SORyLUPC+F76Kzt/9Edo69WkxstbWKgVsnI/ffS
eHkhYdv+ZaNYEZgbHEivl+qjmjCc4weSIUMJYKf1GwgASEcCvp50oLB7qswmOcKWEooIfQ1oljIR
izwkAxTzjWFQRvuqBkuZLLilgUe90Je9XyNV9NdrkuBRoT0y+OK5Q0JlupxZGCC2ecIBw3ZWA85f
xz1BpQa2eN6nNNzj2XObJQGwJ0ZgxMFMGLs8fdWb+Aa0rSycTw2IBz1eWk2Fgiyy9FKX/IeNxLeU
OMOVqJQW8a9trpsLV5ju/P33nF/F38nA3+XX8y6Af8mZJyuZoZk4wmHIpb9WVYZeA5ueb2jPWfeL
zpx/97+qGfe8cU/VPkBWYCunJvDeMhXvZtNpj95bL3rsEw+TM373+f43ZUHgkdUg7N4JEGJwmmxP
GYw0uvSftbW3Rgkb1Pqc4hXpm90UiG01MJSUkXmLxFPijUocJbqNsxnNumWoEc4HRSbY00VbKNMu
sjPVoOdcEomChhrSzdpCAR52pau7U1/t2YFztY8es3voncckOTi+fEQzuJjBeItGPmJbIB2K4rda
6EvWDOc8grDmjInrLV7cgmsW/ffo4c0EcZ06i3TZWLw+TGUVyozMW+4fQ51o45hWKUKzUgMz3meG
KITy/fdjppbo8DMHm7Lli9+TTRJS+XvylNk9UsMwaAO1kcG2fJJAuNIqcoE4rRH+ZQvXrwuj1jfb
4epqWkINKkriGrj8QwqnHKEHp8qvIi+s5JYt7GEZwD6OB2KJSZR+1L5/mRT15SIC8uGeOYRgJeL
3LoSSQSRF/parINudshkyaBYrU2xSiFYXpmIDhXrViDC0tiT4pwmEC0J0UUAL5HSq9ZqYEnxQyih
Tbfpqy4Uy/XgEqrFnbCAWx7s4Zarz8W8kuNhTfi63CU7wPvPAUBmk5HkfnlCPptCGJYTsPhXBVo
h8mD31uVOfofEklumiYldfTfninHjiUn0+iOP4Qntr+x7TdB5PI2ene/GhESpDnBtHSTwJMa/ICc
gkisYTKtYndmYkrpvrQ6IzHLAHHBdHXr6AGokktf2FFCirrQWudCtU5+bzHX/eOPWN9vh23/Op30V
e+6zcVpWiagg9kkRfsg7xq7zGUqvoil5R3JUKUh4PRb2dp+uo856mgOj1w8tW09sS9wiMk3MUuqE
MHnPO1JBxEUUXCTzY2UZbDYDUBg5Mrc+6nRKSibWGUstNSBOAa2zt1JoQRgWM3m6hZUglzpbcb6R
6bWm8qVhBF+anq9aICHUtlmDeP9jZ1t05T8+VvIa86TwaulbQ1JFLxL5muqdgSgU22YvG4qCinXk
HENVEiy0UNqvP77Fx4nPzf9WK6/daPUXNhhYzL25SNO3TVVSfi0vyBbm3u4K1G5rw/TuPHJ6NO6N
q2kbB9Go5t8fWY0XhgXNhPLJsJ4p3E8csxb0QBb2RxlDhy04TM1cU31YMEMBMB9xeVzWC5zC0YM

9Y/wmHGCpKMHEI0rWEd/w1NoG2oeBDL+f19PpoEtzJK+RTbBiv8fKlw/3tS4Jz9SocRrKXqTSXlb
GmHEpCSNnwBX20s7PDax3D2HlS2rBshAiZiW4xXZu9V04EkoU6lDRV+8ZQquGmxhaL8U5bPYZiUz
LT3SEFGckTfO9UhHgirVecUzC3/Bh12wLJm3/5vyB/8ShZecVYL+fLtBgVWBPZlYPFAf9GFRqap6
DXlmq1o3q2js8hN29AcXNA5y6e47w5T/1BSMXAoE/f+RBdbEXx15eLW/sOXX9Xrdj4zKS31RuvBW
yoscmhhTHsdy0zXzKTZGIYpmewgabxZJjG8U7iF6tpqbmUublcwr9LMSOcqylyDm9S69LjEuYDPQ
Ye4z1jfr4fnOvx/xMrTZIz4BHkH30tL+VjbAnv40UN+bYF/tnBvO/fOB70ZBh8k70q2h/VwlHe8X
Nksetx2hzy2gp9BCB8v9eiuf11Dn0jJVIMCigMMgIZ3EAXS6A9AughNHCsnPQoZZYOXR+HNMrJEM
HnDNig8KSgMenVGF6WttFckaZKJSlaJWHI788edfMCzd3wtVQ8cC050PMxNdk4zmtobDEpqtUpZH
KqLGdAaZ2fn6RsKMGYco4r86jgO+52+2LQnKZTHtIlqUUPPnCeVhaao6Xl41tueXHUefxTiVeoYX
gdu5BRKFWi7fULqIpR008Y2GiK84eyUn0hW7j6pZxNWFky6mZoNyGHH7cvci7co/bqOel709Um/+
zynrE+n4/XvueP6a9//J8laRq32WBzvP4e4B//Ha/sBi/cBTbaY4Vy/WXxJKhE4ZC8POduoza5nk
++75NTNiOHF1eh2yXAEaNSyZw2JNQc36/RjPQbhhraS+JdoqMzk2ndFSLJpO5VYAbLCXbinQWgH
et/mbFqoWjRKgiAdyrTSeS2VDIUYC0bNjZpMrxm6h8/EZywU1kKgiqDvqHFi8tJaCNKW7pKCICI
UNSWEPtCoKyD4Hr8zh7LcVmdEo9FIKEcXV5Zp36Gj106NacjBRKu15jUbB80APSUUn7rbWXX5CH
hlqVCZz1L2qgUOvV0ASmpL4FXKEN4ejJ/DYhq31ijobstsLCrXUEbPl5QsBiKGPWP9Mj1WP/Trho
SHpk9LX/fOzws7ZGNBoArGLrARC0d9Rkadl/Y+ly98Je/5L+Ycs4TQUiQdeHE1lw7LLKjNnoaCkT
AilCpbZK6tiiFibCzkXE6mUrKpUpuaqMk0telUWEoKQvC0GBQwzNlsqtKj0bKMgt1jT4fvoXeTaI
EqkfFIBDAfYT503KMxbxSlw9dT84avIp8XOhgKDeEBTmgGk8Z7pkpXlrSXFwk10Jec2ZI+sCV9XS
dXpMm4z7m5I2p0j546S9fe8BIKLxKtbe6wF9Qr9W2W61C/fy5i8cvXN6u3fCG7w70/v2vpSg+ca
wrfVLk562trYxDXfa4Y9p89A4JcZEvWxLquvy5MAA/5Mi5xAZbpXx+IrjjoL9JX/Gu/7RnrF9ux8
aH/z9zmm6Zhf3XZbml67MZhcnOV7/Yx+Q33/gPa9RbIhpluR/XJ0rVb7B93IvTJK00u/36zCqUn
WFXI60K80VDi+vpygIBa0D9v5CoSDMCqDyNMtLjYSD19AiyKJKLl9V3xOsE45AEWOpDqdRptTqAL
Q3FAIKHMDyb4AhEpWzQZoDbPUYgheBKzY6LHBQeI45BC/ML1HzRa8lrzrHUBm3q3dDioFCpOtIOc
qkHOaCoCabylySUNStoDox1Lgh4H2mmkvd6bQ4YllygtYWDQcBtfnvR94FJP92Ln1+srC3M2FSu
W+n3/393zsR3/t9wfVan1275eez12HPbN88tG556+MzVt+z41GnWJx3/VO9cTXkVdbzvJzB/9RMP
8Hf+Ytlu24gCFavKBfzGHf1+a83JHMcon269KyosIO0beMafnEjVQ5/EIZ6XJyOfKg/cJhHUbOLm
+CZV8ZJ8JeFJowWWNopTYBLYBBwnSF7HvaF82m2jSpogeVohKYKOKAPfLESM9BIYOcVBgVwKkbBI
LDDQZ9moz4Kz+cYoWan75R6E5RyEK/1TSsKyhvMlWFBverJoaRa5syvRqKEKx9XnAbA05oOnmZbr
HcPD18/32ixN5oVmlhgb11EBNvZowF89Tr9YoPn738wq12701uoXLyl977ntzeqtzzrJ/zSAdy8
7X5uLeZi3q7mx2H7u9axRKEY1Dql731TODPffr3zNouG0EiZFhkgR9SNfLgyZwHzuca8vF6lvZW0
51ljfeWT3zCohFAerWKC/QsZe/kexiTYHuEwVe8HyMzuXpoUceY48HsUJHIHUp2PLZC6c2DEDtlw
Aza6sWS0bagAXDq9TgIg5J4U219ymyHyga4bmRefQwKoAkGxXamHNlgDJQJUXPF9XexZvfRIWF/W
RxTi158Yw3LrvCYPXXRCmgpzE2gJHksdDugaGi56qEhQyFvuINZnVlnWqNOoWpSQ/ddy8dvvoE+d
W6fM6V/fto4zLZ6+vrjVK1/Kq11a32cHxx9NPv+ZFH/t3P/Ptwz1j3jicdiZnaWX/zSH7pxMnx6v
2XHC9/2rTsl1NdHm3f+T+SzM3x2owtGq77RnE5c5KYHdbET6OwZHvewXIS3hhPxq+IErOrpYa5sX
rqFaAoWd5/FTWWDtHSiZeTC35bNkQYXTyBuvhYPJhlj+ns2XNUsByqVEpKc2am5apY723LevIsrq
XlBQyF90WRyBQK0UjA+1mi9FYT0V01BHSfSplJvTfU3WCsMRurxbnz0ivfTIXlw8pQVSVJGyYKSa
aIP8WpwguDNzycDquDdynKdFEJm4dq10ixC31fCMzwdzxqVvp/gfOUr6coxfddDOH+QH12y01BX
v+AtVqNb4uj2+yfSBOjK/v9frx3bff+sfvevv/9uj3/psfCK5+wcuyPWPdO9QNqSxbw3N3lw3Le2
nmlb56tPHIRWPr9GZmmZ1kPB7Z9f3JaOUBldXCkY28bPJUGnG4VE7ncyn8bjqeIUR+5VKMhoeGA
wHHbj8qFr6cC1L6PqynFyK/OC6snCUKhS5D0BfnBNXvBbVPBdJZFR8MXrZtmuELopVegsQ8+3Zp
qqVED2libezqSZCl4iwbNJeJOM+WoiI3DITxNQusR8DZESc0Z00f/S11C+uV8KQ8bMUBXCSbSpDK
ViB5JukJulwpA4HatTo3rwoMBJC80MNg2Jwx16c52i+6570GCrM+5i5doa32LvuWt30gPP36Gw+
AJ9dodSngDQT90MApyw2H/0Hg8/udJZnj2TN/8ee//9v3ffVr1/ov+7o3p3vGuneAlc/KH3xJPD
z3aT8Lei/OLOumJB4G6bgPGNwk3j4H+J7L7sAJg1EuisKCW2i4WZa6nH9ytupm496ON+KwGIwLh1
/wnBw9CQVa3PkNfdJrpYihMW85hScICrme9FirRR9qjbmKRx22DjnFNUK/94SIWXQmijVNLOGyX
ULBAwkIcV6aAOYkMYzMS4l/2FQBGhhpMbnEke+ZZILq3NkVDv5SiosHiQ757NX9ZRXh/xkpuZelZ
6h2kMzoS8YPopUuh0iU7G2GgYQwSqwVghqTYiZf/wjF00D0QcOaDa/DI9+ugj9MlP30+Hjx4S/u
BL5y9JfxS5dMgWze9TNCzrmMcJPHvsxuOnzvllp/V7n7zr9ls2v+fHfiHaM99nA+ThKhTy+/HjVzc
w0SiY5Dh3RoLNS00kyEDO4eQ4PcxQGQ0tQPuwlJ9EFyy8to98nWqRWeZHM99bA+cYh8gsosuTJr8
wp7C5oPiNdqDclTBQCQRXu4ElLlTL1ti9RYY4Nlg07r4nMhGM4VWLFouc9CyZHihXeWwmjPWRoj
ON412+X9G5UBQvCYwUrwPneED9/PzqsRdR7dhJcgtVxZYoo3QWv1wZp/RkMz3ELiF5osNhzQ1lqC
KTDMnbirFCzoGCMu3I7zGocNutt9L6xjY/RSGnmgvztLq+IwiuCacKCyv75T7sbK1LAatSq/IGZu
Qm4+EBzoULk3xucRyZh9fWtz/8Q9/+zU/kipXBT/3K+9I9Y32eHvuuf2Xv4UuPrmRucb9tgDLItC
b9TYsc3zITXyhAY3soizpXnuevKreEYpnBeahTnKdcfUVYC410TA7nfoXGPlm4MMY4UYUWtFmMSA
s8papIBJTRzuZFKuVrYpSxNmYYLJXYMuKWCjOhwuEqlTclqC2OT7opKAgFangdfVSYdaLaKKISl6
oh9NhUPN/FA8epev3L+LM0RL9VEEemYnxBzjs12ClvVapJlOBLNrR6HNxuqnvAln6eMnglwgxitQ

sp30/3PXSO6nOLgo6aBCEVSxVau3yBDh65mnrDCY3PrYkOK/rKIEzLgVoVbBB85MvlhVKpVBgG4c
poFB4PjMPuv3g7v/9m9681mw2g//3N3873TPW5/Bxx60fdzfW125u72xcvbf6+aQRDQ+P+v3F1Y
VmvUCjw4eufQE9eM+dyaC7nRw9fGRSn9vX2964NMyGm3T1C15SdvxyEXxDAflkskcFltepz0mPEb
mbJ5bEZlFucp5VlH6kV1lQbx5PZNEV68tiqMgJodiaomqreXlBhSkTNJnqh4o6t7D0J5KLokIr8A
rO+WAU4FnK7Ex6qjA0y7Z2q9jaQITtUAWN8Yk54u0EtzUSysqF68+L6CHPH+1XV8po2v2fkl91Z
mVWruM5ilwRZwoD4+815Br1YPxhlJMn07tpBEkLnwZ57uVvSpy1f6FVWrWSlSvlWXYAKgmTBqBrA
5s+OgxT/h6JzE4lSbUmKvT6uXLlHRTe2trCzCqXLVarXW73aOmad5p2/Yto9Howf/zW79xc/+hA+
G7f/aX0j1jfy4cP/+TP/qyc2dOv5Hzylfbjv1K27CtixcvCiLnqmOHpDKZdx2oHdGZRx+mhYUla/
nAUVBzcc5p+r6XS5aP3sxOwaSd7TVr7fJ5Cx509eIpgfJhJrJcrVAQxXTThiYfpDd/yXVQuL6nBb7
+kmRhiUR/32XDFowlWN9H5YboLrM/UgPdU5gKyEKNBn/x8VRFsJwpVBP1WdShvKcrp4GdCzmspgM
M095WqsHi/KeWoghKaiWLTv2AFzm3zBWq+7HVUaixK5Zc0Hako2SA8BldTpnq2FmmZDan6KsJvcC
ZJ4Up7UkAeETGAxM3UvV9RqOPvgXu+4+O3y3VN+N7s9EfUZ8951aFl8ngT7LS3pNUEZNYkjGVjCr
2EBoMt/r1Lhw4fo8sXL/D3AahwcqNxmB5b4X/REcz79ha2vjrtFocJvj2I98/796y9bivv0w2m
TPWJ+Fx1/++Z+6d9728bc//vAj38/h1bWupzRb2u0dyvOO3Wu3qFAqU8Qh3aA7oMHwIudIQ15oAd
WKPl3LRlwu+OSsDq2Vg4esOOzTE4+do94oZW8woXjzHHV2dtgDhHTk+DX08JnzwhD4lre/m5ycCm
Fdv6ZIrzMVj1YX9onor3KyoRiZkrpSHk5EpSSn82SWVUSLZWTNUSLFULCKZy0bGCdkHgxe1KaIHq
s5UozMZqThhPJ6NVurPLL2PDSCTYM5LxgkGDjn7/xtVRZPCB4WhGGylR7JgXNWayGyaQCTdNqtN
KSyaTyG2H6TBGraaGrXUNFJTmV3BYQRQCJN1cv0rlLlZUnsSH3MeGwfjgcky+V75IM3U+iRISiAw
6RW/0BdVo94anK0nNCxoa5WFKht83nKRocX/Pr6vycq3hzeGm73b7n4OEjd6yurj7wg9/+tvUXvO
AFwTt+4IezPWN9lhw/95PvPXnnJ2/9b3nLOQlDgd5LyAbW667Lip+fX+TwKpYFMg4iWr+8SqNgSI
lGQ42EsWfb2dmkaukGHTnxYor4tR/86G3UGiF0dGjYb1G1nKeVg0fEYG6/5e/o5le9jr76jf9SGP
GB8wUeVzwqvAmv8mJzUQwCI2QIIo2pWjiHxjBoVHVNW4HvQQkq+V2c6fz00zOpigkURgevozynMi
BD690IO3+WaQyRwkQYur1jikSjucv6AAM2VWFr7oZXU+3wCc63vRnIQ1PTdKocNpQfE+oYKXK8J
NYDZlLnjolcxM1AFsNE2hRarSSUpoga5fP+rEPf4S6bJgwpDwXchyCGzYsCYEtDu0dP0+5oiM913
KlRpvb2xyxlOWe94c9qmRlQXTlEedhz96GEls23SKfiGf4889b3vuifPnz78iHI8/4/n+radPPf
qXpKag9ozlmt5+9r0/+v23fPhDP8uL3h+nA9q3/yBdPH9JaCrnlpy5VAlojY2zVm9Sayeg9dU1KY
QsrxygkMPZSrFir7rpbB3at0zNxSxaXFunj992G40Sn7Z3tqmlvk5Hrz5GhXKDLp89DbJO+o7vfT
c1ltgJwWhy+VkiC8wtFhFaNMLnqxkF0bIQhUZTsShgmBsJJ0JlTM8Afa4j1uXgGauo8s+GhMEq5L
Rl4coYW6KHy7W+KjYEkbNACmlkM6A/WjowmJimUpI2lU/eTMlJ2V8DcacCkdxJsLF0ywVea6iGJ
8aagZlgAg7SlV/VTR5dNUY92A6iWPKMK/iKsZQAT/z4x+/RRGzaT1XiSrsWGREWlfbfzfn90s5/i1
dsbm6+/id/8J2325b1B+/5hV8P9ozlGTp+9Rd//kc+ffttP9vv9ahYKgmEbjAeUBAOZMokDHPShM
HiALMePctaLditwQ+0PF+iV7zwBF178npZZH/7oY/QPfc+RKXqEnVbpyUH/YqvfZ30P7dWz9D+g0
fp9W/+JmGc99BLFGW3XenBI98rlgTwQIkIXBKCnyQKak4NTEDjyve0ZFQmrRedM5TnVum5eX9wM
equdWMZlKQqMoqI7WVQUyBv+ZUbiOe6eXAtxoQS7ZMQSiJE+b3rVzzUmocfyG1bKjTPcGYKstpxR
sRWE4TTSyUk8CplpZACKzh9Wbq8ShEGUoIy9Bka6IGh0/PX3e2NujUqSd0iK+iB4ev0+bnDTnCQS
8XraBgEvHnHlCXN8lrrnshbaxeomuPH6Vz5y/yz4cUYridFOoLIToy/VHAq9frsvGlouKe2Lxx2P
mc1+Rdpn5pWptXv/1URD+H//6n3/9add3f+h97/+LtT1jfrQPP/r9/3LD7R/76E/1Wm0xRoHKcX
h3gXPJer0qi3pjY4sKnAvhuHzpHP+8yYukRBEBE2opLzh+kE6+6AYhJvurv/0onblwWZTYet0WHT
x6XCqxn/nkLTQ/16TXfv0b6YUvfQVNgq6wxMP92Y4vEy8wONsxJaeiNBCZhwwVlclOvQxFws00bY
riNQqUsjhaQODYJlPUj0nHr7207rn949rHKdCDGAsowBxi4ZkkUW7UhsyHqfetCnLqcqVeHMPZH
GIWb/25VReOSqhr2rF6F7PTKDK1EoAqFjFipxNvHMq9xg6qpTtjsGRrvGKI4WlBtzl1f13pRnUZx6
duvZ29ImZuObuOeaOx9PCgpXRpZfgcI4N8Qef5b1DijTc+fZq5QL1e21aXmzS/pUFmeXF/Gs44Q
B7okLx3vYWDbsddS+zWEJph99rMjFsrDHbly9XecMoLi7tW+CN/CVW7L7xbW96/WX2yt/LYfInfv
2//GGyZ6z/i4+PfehDP8a7sBXGCKwQc2iGBQMP2u8Pqc75qON0xRByhZzSNjUEd0TVnE1veM3N9K
IXv4QefPQx+oM/+mPyiw06fOQ450pFETt+5IF75byvfN1L6U3f/DbOkwoyqmZYvhiqWvCWVlR7V
RGN1SKZn5JFpbi3CXpO6ZT1kA9ZWYzEcWvIgs/mEK4R2LpZp4VAAfLMvYlXjEaBty3MwRozGMRK
rAwgSBB3JGzJDqim9iJpIX+odPUMHfNeTxaqLnKgUh8AoJnzXMGUY/1ZiYsrllIiYpmUcNuEkwxX
Gmp9+uWLPicjIE+yuy6qZY968t+6o7bZegD90Z6xxnJCiRUR4huszeGbi02Bivb6vXp+1WQ0KP2
p5snNVvj8T4VFyXVeOTYtFFTFkOozHjVUqChyUkISddg91KBV9ZJm9zfkvNpjhsF/h8LjSt/t/yV
+73/LG15zhc367nys88Vt/+P50z1lifckP9oDvo9l6fceaSS/6kjMAU3iCvE61fusz8Zi0BDrbm3
Tt1VfTmHdlzxjTN77ha+nYNVfTn/zJn9Dtn7qfcsUS1ZtN+eOfOn2W+p2uLLx3fOd30Ute+nJenK
EUUVWADDha3FmSyLBWigljBSC2l5cqeHDhf0pVUpet6hecDgslQrRBFLGhK+BYDzN7aFC6kYNjhRW
nLiKadV4lGcVhp5xUToUAPFYBfxH+1nKMYlRa3Mvn1pcUjVLv6Rgp51UacnxfMprpPILAVq/BaCy
iDonTWctJ9VTWXSmo2V7CFphp2hxfVvE9T8AM4mxCaW6JIZ+kKtko/7r/7PsUNxZsaJ5SSe0aRQj
slhv6bhbH0Ygcc9cQapWWkXRoHAalttIQOxrANEBZyeaPM5TxBSmXwzBEGoib8XoGEyaat8mbZjC
TPTqnb6Uu4nHD+KxQ5iZnn98jz+lKkX6M7B4NB71++/mvuYsP9IX6c+50/+UC2Z6xPwXfx9eKBIA

j8ELOYQNFYpshNYPdMJawzJbza2lwXycEjx45RIe9Sndf6a1711bSwvEQf+Nsp0aWtMR04fh0Vyh
U6d/YsRv6+KAZ7ZKVB7/qBn6R9R46qciwbnyCBrtCQy7RGapoEkre6BZ9cNnpUWg14p0gpu6XZVL
Ftt2qUCfBA04iaiml/PBoQn4BG/R2prvpYkK5WOECRB5y8/z97XwJt2VmVuc9w77tvqCGpSqUqc0
IIPDKQoAtFUEREY6OgtLpaBAUVFIwCYUEnC8WBOLVTu7BFRWjtdliCtiyW7QCiiIzkHitzyFippM
ZXb7rTmfr/vr33f86t0L1W9zL0SoUKtap479W9557z7/n7vh2urU/kk1GQCS908bOUEX5+xyymdM
5F4c8zSa+D4RURK7Kxthpq9vvklHMukIZaIpiZRCMYzwOryvegm3EiXS+ncSVk0nQAF8ab9b9nv
8VcMeyUJZP+Hx7brouGGSms9tKx1VoCiX1METajBj1YlqTeF+ZY4MBglmkFEAFaMDh4EJ10/k0OJ
ahQRyFpYteeY8dYkWCpXb/242B6FmgCchnl3CfsxRpASz11vDCW7KsPj2cp1cF4z8SUuXPhkzgA8
Fwn/r9P/vL5ivG+v/46/D+AyD6BNV1hTqHRFewrllTs03YpFlbWw7p1BbpN5vkG155qZx0yk75m7
/9tNz32H7C2zaGpew/eAdX3W9dWpRvenlL5Ed+9HLO+nTDqeJ2HRmko5mU1DBucQsHBYAk1Hxknj
RjPfwMQtbyPLXJNemDXq9MK6KQVMOopqES9xtebzoakxg+HyIJRkCMmrUupNKUTFRZFGtWURw4PM
G54HC2XfwK2XTK2WpIKRUXwz0ZMbrh2tdDhvFkOKgnnXYOpU98ZWT0QYnORuvE9uHUyYmfpZzrlp
a90PkwW7AIhiabtNrD3LKe991Auv2220JNantisUoSguLBqYDXC3AEYIeTkoI2HQog9+RaSstoz9
wj4rRM21jNE/t01bEkTLWpMUVkVTWTbcbXDKfrIVvp6cy6zFgm4GGryuD/rkq3hDR/SzEpfzDPi+
8ZpuPVN3zHt94TsQV3zs3NP/iRP/94+RVj/b8x1kPLKwUecKORCXFY/CGLpXPTSa2SnOHQDMcjed
6ZZ8kZIS3+zKf/ST5344OyHOqiKRslKV0o551+slx++Y/LhRddom2d4NWxQ7WsFIdL5XrUkfd4mQ
LaxUYjTD1BHKdifqo7ZcQWHIOWRtX7hEybdLBIORYcaAiIQb2B2UHjzaaxjNAwAepHdDRT2zLhHq
JMqNmUedNTnaPw9AbbTiEHtRdqZWgw4X0B21NsLgx7Twe64b2P7HtMVg49KWdf9LWq6wSGD1DMkH
hJmjhrdVoeZWLqql26bGkvFW6AG8ZyZ46KdGcr692QXWDEgp+951+ukbn5BSnXR0p2D5kBI5w1rd
iMorNVQ61N/X+OZACTo7O01ZNI7yEzXqQTVm0s5aFhd5wPnatlAmmTU7sp6bF+ILeX+GpND3RubI
AVzR6yXrjOLEh9toR/f3r487o02xh+72WvXg5//52QHfz3kM2tfeiP/rj5irH+H34dObR/pZmWjE
RcKwgd2rJkrYL6bjIeyuKmbQ11iNTjibzwvDPk61/+1WR83HH/XgajjWDAC/OL3D7+zS/7Knnne9
5rC3vVi4OUjddP0rUnoiijnhgDTFzr7gxRLH/VTMIpyvE6qWINuM5g1KaRul3iWVW0QpWLBOWtDG1
tI76DQgOZJo2il4fKYHN2/19QKJY500MKpWUmyIQUlCaT/g5D+n/L1r5deuHbuoQ11eRVSv3RF1Y
GF64ZsTDDy8WiNMMHJ+oo8cOu/yrkXf40MFjbR2dVJu8DZTIBZxf92CXiTRkMMHs5SNPfOo+XVc68
En94WS5QDpdhrlCn5GUPi4vqMgbKJkDUvvh9EFOT/u5a18jQE2tLmktT9+Hq/H+hZoggu2JAJE/N
9w9SW4wkYZVNKBSuuUtbAnUFqpUtUqdt7rJ0yp3VFSd7mut2RNSmVaTXeFUuQ3Q2b3s8MQBd70um
+/MWRh73/eeS948H0/98FnPOo+6zSYhmvr72STj1IhmvauryhyWTYub9SFUIGdLSGuX5ISTC/LSF5
8njz26X6677UE5uj6WtdWjcurOk+X0k5bkZ/7jT8p7r7pKBgPdPUo9o6o2RftUSdrLWBE6ENVGPb
m4SRa3nihLJ24PxrpeLxc0e9AY6gxjBB+ULBymi6rakA/UmJA+Y70j0vjJxhQVBELVw3rDEA0f3n
NDiCRzgmVMvVz3r3K0mal0KAES7HLqDtVt174q1Mub6BxwENeWn5JssCB7770xvP6qPHrnNfHQw6
nhd5YvyGS4IQ/fdr0Mg+HWnqo3tXaZReLbt+GO/plRNY0RDh221OCNXht6Ywn089abbg6JRSzRS2
nC2t6UWgjOaJPdE5WhU0MFbWfu0NMU2NwGRloYiyHlRW2gBALFD8Ngq6R93cacK94PURXosliOiD
pIX0yN51w0EmtaonlA3YMjwPUXALNULcySXyvQHct606rePhmNTx+Px69fWV2+7u49tz125U+8/f
q7br01+4qx2q/3Xf6OU4vx5J2V3TzL1oSGICzx7c2RegUvDF7pmbtCLbftZLn1rkfk4KF1Cmdv3r
wkX3PJOXL1r/yqvOwbv9mUFGpiWxsbs6TW8SWyCF1ZpIw9lQCFN0dqJ7mOL8hndVEcurTo0iKS9R
cWKb3Zn1tkTQsGThb+XiY9Rn4cFERpHCAN3KWsLx8kXrkPAe6ksfpQ2H3VFE0PJbeNhzp0sOsM2X
T6ucodZd04ie7mhJ2ncUnzt1P04riHRhYOOupdXE8/n2dNu/euW2T18H41RBbfeujTug2pnnKmlv
QBtDIx2CDVJHRbs+oFI1VLdKZ6W6hXAU/0g1+A6I7UHfU/HE9ay8LcQE7cslkWF+YYUXsc5ft0/S
3lRroOOVN1/ujcGs0wNBjxJxxsXVWVZ2ujKrV6H2VSBZcdQEeeI3gmIpClTZ8Kx+wycr9bVT6Nd
NUnBMH+3wQFOAZe43U4TNUgQ0nuw7u3feSP/zdD3/2njvvSL5irOHX2urqz42mk3mvMSrz2LjxAI
UDEA61hclESdIX794te/bsYbo72hjKQP7KG77nu+Rtl18pm7ZsZYeVqQ70j+qMKSSiJz02tqAhGv
XnLZrkPKwbolWij7gqIvw8IiciKldGAN8aUlJQ3TI0goAkynocUYyDQU83lrXezXvsYroiA4z38K
NfZLcXXdI5pK7WYCHg14yHs9BaeAPbdr80pGzKloGjwmwzD/UyjHbTtjPD30P0MmUKjfQKV+yHOn
NurhcygiWexv0P7pEjTz1G44hjoMZmqKlBCjONpExDo9xLHXs+NIJKU9y6l1LE7c4779Qo5wguOL
bMRjel6hWztX2+vIk4tx0nniAnbt7EiKoGl+r75ZkR3xv+O7xeYfhk/MmZcaZNrjgicyft9BTtlb
RlrGZMZzshiGZpmpFZhE6s4VaJtRlFtZfpuGz3rcFBct34PQ73+cDhA1/3Jx/76FXPeWP9we/+zo
UC0b2ZnrEyLmXqNUfwtfCFScGFvOAJ79+4NqW105MveQ4dlLaTJFzzvLPmpD/yUyPrbX6cljBYpOj
PkCkXlFDINLqzFj1TPQRDBaO69e4986INXscva2KLiYtTUgkhn2eltDK0iR6oFmy3QBb6vHqaei
qY2FJjLKPCe06LkSw/+UUAUy8YXb+nIH9Q69JGjbg2xg8+/9zmbTJ30k4pRhu66mKqZ2OKFE0/ma0
NmtAoRH8aTWzoNUAL+3qNsYwIJDxkdSV/2P/KALO9/ohVhS7XzzGhVar2axE5wK9YmndTXIXa0pY
7s3y9PPPEki2RlhXkGg5QSjZ2UTeyEdaPXpWgkDYLzWsA29DyPo6A4IrL3QqSeTetvBHHDtPUpD
GnkBndT8sFFKPsxFeNRds0dp85amqsJxHufWXD76jgmJTkHzsJn918VOq1xFm0y+LQaCfT3r59e9
/9m7/wwe3P6QbTxurGL0GPl40KayMgdY1/hgeFcQ27nOEbLoQIVYeUanjogHzTt7xGXNhZ65ge4/
A5oF4XIGM4Hx58SIGSVGsidF3TxpcRJ/J4OHh//zeflk/99V+ze7zt139Tfux97wOcgSk1ZU8Sle
TEvlHUT5jPTLZXZDRaYWSYw9jMKOtP8L8YbfNjhoK/ufYz16GJ/SSVPjPkCBQhIrwD9B8I44kfTyZ
iKmAvRHPTbWNdFBK9FXC5TNjgCbjjPmC2kaRFq4UWpsorXWXG+qdh2pKv7HriLc8ztp57dRhw07h

oQARTs33TqWVeNQDOp5hY7XQ8Z3IPcdP1lwRAzRqMcXW4bfbEZRS6usYEaPffj+frrqownGOqfZAU
KaltR8Duhyg6UjNprJ2zDK12U/GaCTJre5rTo4NiBTUrFhm6nOrguSBFKKBtRJQXqivldtvfDWWd
RJ3ZFhFWnVMzL+G22Yhc9YFNsfuOfetWIA95w01n9/2be8fDwavSbn0qZMpU5s9QM9rCiSCWnSFO
nuYE6RQuWqvOVt75Dnv/jlCgDAInCrNcHjhLekTEpFyL2mdEjlbJTwwAMPyKf/8XPY2X/8vAzH6C
hqB/K/fvx/yPYd2+R73vIj7BLjNRpGMA1BJ1h7sbain5j6uX3+HDmpUwVyCFUJtWGEFYgQ9c5BDk
h0hgiPD0cCA4B+ApJqaieBNDBYIeXyg5wPuE+lDbOSmGYZhyo6xpbB5lDqUJiPhicBfSaMh2doB
YLYe6Je++SYbjuU8+9UDOAcl+RVovzWistL8jMrexPUOfQrUWECY6qCD+G+arrRqEEaHu0tY1K2g
wj80Szo4ph/6HHFDize0GgxRQ1ojrFAUdkaUwQfcTEKIf/Kl+epXWtp8E0rKmWT0BD5ZEL3OPLMW
XONJVPDb+cJUaWaCTuFMrEVpk4n9dqbNTB66P1C6/4oR/a9Rsf+9iTzzljnQxHVxP0gJod61y16o
wV8EiZClZ6LMDK6KcDOF+S18vC9jPoKSuCy+eoC8R0CQ84qYlnRaSehod7YN8T8sWHHPQH7xfnt
j7lNx/3yNyNKSxG0NbhZjoJnCc3w99+A9l84nb5NWveY3Mb9psjqPi+onh8iFtjot6leeAUGWTCw
cs7+c8SA0fdh68+7qsh3iSGUHu6V9Z2DY3RGBqBvIQ9VJN4QbbdnFOqygeZf+AbO/4YxXvLhgZLf
fmjJmQhagFw8GcZhXvW5EUMcWEUsVjd98pTz32sLzoFf8uBOcea3EpxTq+2sTxyJqwjOixOUQZRa
O142233mg/3wL/U9/V411hj1geHC1qewMqMckawAybAmXOiKkmsc0w4DjaUmK+tTFMzULTVRhqhf
UdecOMBR30qRkqG1d48iRQWJlKvamC8FFgt8HptSvujJI0rU5snSYcVe6Zg3Wlw9+3rG2s/jbizH
PKWL/zml552XQ8eUVBGpSOZQjeTprY+i/qmkgZPCJ8f9eOrfJD73g3lz5JqjImNZ8mhuNT1rT79j
4u9913nzz2+JOEGu4LqejOnttlvLEuCH4oJiwztUSHXFGFblTNyqQvRGcx8/+wq/KZz7zGbn83T
8hL7zgxTJcPSqj1SMhAx6wI8pIlyh+WEKtiMgO8TCaIIAA6oUTsrrvMXZA0QBKyG0tyHNFLVaSWp
fYtBN6+gXXQy6eclZ4reCQ5jfTeEAYQPR2pQqltFUGGsentIGGPaoi6dHRav1WI8IWmmCgv4BDXhu
vyd3/1SfncF26Qt19xFxmj5NyaqmHiKXKqMqXs7JY6okIk3vfIg7L3qRVesy5jrmPTJpO2Rm1MMd
HRR90VIHnu+OssYoW1ZKjZE0gMHJFZ82tqNTO1kBMH8CeM6FnW4pinoZ7OLcPJKEZet+spjbjgk4
C0ERuJtYqnGj3rCB/tItuIvMpbAkO4ppde8aNvzX7j9z5SPWeMdTgcXu0PjLxRo30hDS1QD3Hmpo
ihhp26TN531fvkohd/tWysHJDD990tt91+gzzy+H65+549IWIeUBQMVtdvQJhL1fs2bt7GmnJx6z
YOyJ94/AnKZ3KBU6oa9ogchM0BdlqHFDHUT5//wm3yvLP/VnbuOp1REukpRzfACAOTGV5jMjxKg4
ChqgGJONZmtHJQ+oNeqH0HynoREyRj8LcxgS0sTtK+7Lzk62Tr2bu5QwbGv3Z0f3Au25h+FSuHZX
7zyVGLycEIVHUGyyaJTS6+Pg+yGgicIbKChx7eK7c+uI+/H310rlz+nvfK88/frfe8h7S8ijDFxh
c0A93VVwLDPXvuieQAysDAnpqkjZzS6k8l0o594tY7J/NjTgrcr3XsES0VlZR0iO6N1Y66vhKvxX
GM1argHmfUPg5Zw2gsOWR2uFy6cU10mh3xzQ0fb7wOVq1NFaOpNIRQsJwoTGYHNEJ3Pl2xOnMuC+
Fanh++de9zwli/45Wv/K5xMf0qb2Aofrlpwez0cSGVNHVA3DZ0hX/1135LFuchsm1R5KUvuTSCktg
/L48H4II9y5jnPIxxxGmq8pU3CmenK0cOytDAVcwt9WR1XsveJ/eHhBiNemGNTw+guSksjTC2Nhw
Re/ROf/JSMKgiXzfJ9b3ijzC0msRZOe3PUDC7WlySZS3RVRI0rGt2AgmL6yxFFRSMj/BBi80oZLe
YqmE2c9nWXhSC9xDppvLYsg8UTJAufdf2JR0K9viquDTTtiulqXKr/i6CBqC1vKmzLTCwLxGj5jMw
119hgq+PLZa2+WARVnKcmu+leue/HL5cr3nW5vOrbXqsHFK9j0jKKTvQ6HZtf4Wt33H47a9XSm0
BOVesgkmbGK04kSFBd9mZGLKnJVZWFprYshJBCgXGhy53aJgLDAJNAUSu4BSfbG4RY4AxUVI/k+E
ZcjoOZWaIMJzxHIiIziZpXqrLhOzUrwjTYK+kgqzxp96ctNFQuLYtZTl9f/ixNx33o5vv/rZXZ2
VZXg3heUTOgVKKXuqrBW/k8NAZVw+Hg/DX878jRdXnk0Sf13kcOyjXX38ltayedCL3x7B5j7WheH
DBaPbvfyhYVCz0xYN9KhgqSM7zS/McBXmN4i16YmiNgqX494QKBx//07+Uj/3JX8r73nMfa99yqs
qEGOsgefexv2hxVIDhgsnRpLkRhboRL6ojCUVhPrusqXJs3/JoPNfJg0xbjxVak7yHalCORrB49IJ
t3ncMGk2oRd5Yd28GqTTeJXXBbwIw0FvNmgeLU3XOfHDyyElM5fMLDRzfk5z/46/IHv/Mh9gLECe
3WVIHMC8tsiACE93zgvvvZreVunY4xdqNmTCs76SVnwOHaERkz2+heSbs7lKAGS4cxkkoT5/pW/D
lkO0VduQePiCVccxMiLCYBOi90O3RabSjh3+eJKlm0tEN9vs1MuuvnwNP2JCp6tNhprZnxXL7sOT
Fnnayuv6Eop7vZ6TWlv8ogS6ndpMoedy0mXlBbakfPju1rudx5/yPywCP7pDfYyt2n0+l6iEBR9J
kgm5993m7ZG1Lfe/bcG+rXg7IxmlCtgGwV25qzmZ2MHlsPXGODdpNkQXRHqj4t5Pa7H5QrfvIKef
D+u51WjkmD0z56KPx5iMbDfwWAQq51IBBPTMsNhaWHF6l3E1c7Onh905nPyjYb7vk5/IaTeQz04kx
VtBjnHNKrl20JlwyfTYfjIS1TBH+D6jeGa3Hj7Hda40wMtxlvFffjP/24vP8975anQoaim+8wOj
F5G2aVSXiN1ZA6P87X7WGS2gh/dyNq/FrjihMpu79jOANL071bXLEpNGa9qim0Np3gYOva561NJH
P4vYIRw/ABAjnze8w+m7SkAP2tpQJ+Hs9YBe6sSw3HYOAuyt0wktfxuftncSfhRq51LwNK/t9+/3
ft49pYv/0bvj4LHVEDUHooO3KaHBOQnaGP31XjOYowQevBXIiIhuPEoqP5hRNCvOpIrrn2dtlZ98
OS97bIj127KE06CjXnU3v3yj333yvjELFRG28JutGihQMUEgTvKx1M+aXtYWKapOvE49wRSJgHHn
tCLv/xdkn/uzPQ4q6SIQUonttTZ/KRQ8tGwAlSbAXcWdkNuhcugb3m/70RE0HWBJCMQXq61Ou+
Abmc4pOtAk9cN7IvWvrIZjJKJ6S6rRObwv0t/JeCy33n6fHF0ba0Or0aXNOByAP+IflWUq19y0R9
7+Iz8sf/8/P6WjkrQ2eKF2q++942YZV03sGEfQiUcqJyak6Uwa3BvMMTIDBYSSgMJWYAASi6sZU0
3hNvM3IZuoTG7VexmentLIOeZRQ6USYUirPZrYZXZD00CcMBvQDrfMADJaTqzVsibZGpleks6ANh
zrbOSB+fXVtfnjumYNnv5Nk6o+VxsLx8q3KsON6bM6FkdXx5vLMHx5oJxjYInpmctls6GFAjLkM
bL63I4/AakD80HPzjbt59C7whNYW24VCRB4yETi4sDV2k0TRXXxr9oa99aJ+Fns8Y6lujyrk/kw3
/wh9TQ/f63/CgPCQ6210uow4EN1jSqXVFBMgJHFD7qqDluAkFg29kX8BAnVdtUAWy4IkZPT3fL0s
YQlTWYah/jh/fs6IF7S0JBSKEPHz4ot9z1QBzwr6CB1WOZRXnc0wOHV+UXP/jLsue22+XH3nWFLI

CaZ3LiX3zwUSPW19wCkHS6vHp9FTvLbUMmPK/+HNNcQEIHg0Gs/dgswggL+2MdgN8kfHa9TCOXc5
l1/w9ExXXOPQ10OI+GqiJw3NNjCDBGyrqJkMqynvKZJnG7uzewDJTdtJ8/7YoQNLqSRLnK2hDes+
slPf/svZDtbAGm57iMrD/93vdk4UO+H1EINR9V6esibi6jAkA9iyZBvYpadH5uTh9eoSBvztyCh5
2ElHZ+YYFaSmBvAnLUWzcZad+BAwfk6NGjbDpBvhShT9MYjXSZTgSUjlUTWCi5eVo3tDIYUGUbW
2sAfmSj370T+TWG77ALqLXrGLNpTJpjgQCVTzkPpOEs/D6bQeACj0F7Yt1Vx0/mlrql3QgieRylp
XtuGki6ssxvRXJ7iOykK6/6U5ZWdl0GSgdp5EYsshXYGdC0YRo/Vef+nu58l0/KfsP7ONMGnf+2K
MPU0CuMf2J2lF0lnInpmOMLIkRKlcv/5WVNQN5uLGJxQ11cNaoulGzkwpXshGH/3Iaa1srkuMb/o
RT1tfpzTiK+Kfp07fRsIx1qn+N9WvdmPhbGiOoj5A8M8iyufilpNMx8N5A1ds9IwcP7jpu0+C1tb
U3hQ99Lg4X0tKKH17TnLLsYFDjnl9t3+NYLi4u6N5Q0xLizc9Tinyth3QP0pdYe1GFaLu2MQo10p
R/NlQx0BUQXWhZJD1Hg0q1+aiVjaY7NiJhimlNkzppebDrwVl86i/+Qg+8SXV2my7laCmaJA9zBy
9L52ANoZN3X6pLp7z/5ARsu64WtFASDOERp3kaKvWJAcAsQxlw394n5YY771XAPRyfKOger+Pvo3
WrzW3jfclyN//EW98qt918bXhWjTz00ENESdX2GmJsFzqwpDUQdHRZP/ZyGhcfm/w0F6nQgyPer
Jc12F7acN//Z6m307oartfyFzwSRA94Szk9eexEdbXe0bBirrRwAgktnXCiOGkDBQTWOTSt+jbS
qRRFDYImo4kqy9T5GxVSj+/PCBg6cel8b6tje+Idv3+ONXKqKLSAR2ALHxDB7WaxOvF2IUqPShoC
+Ih8tmjyvQN7UBux0rU5KATiBAbCCFw5Jr/dX9lXDPy6xxOXqlawQehavY5u+kpOFZXn/DtbJ8aH
+km3mrhRG0MmidpYmtglCYG5g/nAOfrfVqEgEFbSRwrKx69opjiBIk9WbaRhe7tMpmkZPhkFH18z
fcTM1erfCo6qED3l0qxdNARx4pFK/Hz/bEOVW58sqfkb/4+CfkgYcelwL3F+OWRuVTVda4remTzq
wX1xScM3WytTfgYmtKrZsYzt1ZN9z5avKsxPbaqkyCJ4LzQQOwnyJN7ptEqjKaa4h5Bzg+y1T1lp
mFlc1Mt7dqlJbnZ83PHUZIPstlSm29hcI7KR3QB74fAkTvuKxZn3z8sTdNp+ULiC7BoJ6H10S46N
FKDuLnkDKFnlVKKU3wY0cDBboiWkA6OrBCIK3pWEmbsdKaw003EzNd1lRBWlYlFgtNE40piIN7
XN0oicm4GdlaIjiqTKiSCA8DrJxOnbeTyv9aSiTJWY+uJ9JuumeqizQ35GzDBTTSPzcFE7XngHQ
ZizkUzYVXLf4s6kZeLJlZR6kYAp6/VdYvng/MbjxhVH374Ybn/4cf1+7VDF9OZVM9npIgYcIIPZ7
Q6X3Z1xOG4kt/5Lx/hoU8ZWRtb6mxjI4NDcB5r6TQiIQxVMchZTH8ZtcqK8E/NpHSumlqnV39OJU
iRTqccPSnhHGMWYJ45s7b7i3+f2mtr4yqJTr4wWCKlgCpVHVHAIg/eaz9DWVoDiuoUGadtng3EUR
BBcmkcSqfMDZ9h53FnrK//1ldlK+sbVyJ9cH2gXHcAs3Ez5ewuZW2aI7JkuUU9lFJBqjMdvRHkz7
THxixOEXCMAW40pi68ucaNGr67IoiYlXSImpUyGs+OE2ttQsMkYqqlhrRUUkejblpPMEmOU05X7
j7wlb/12SoqIq/UsaY3o2HfNhQLYAqH+rjOk2tgaS5z66LvpbGic+hN6Woc8rGuq7swMZSodRVjN
2uZ0OVJNbwo/G6rK4sy+euu4XKgYopTtssAdeZK6mbJmyHVsHrdm1N5J4YeiiJiCvtI9RxxMZ9sO
En+5gd1xA6yzRyTnSm2rNGm88zveRhVKtUNwlb/YDv7Y5+/DemBhBh89TXv8/nllV6Tf4LkVHF1m
vTemp8LhrfV0H5w5iAamah6TUcBK6bDSzrT6hzahQJlcyMqDa002OdjqZvCjXHC6DwMC6mHMF10V
MAfDjgGZklWgdOwo2aJ6IpIehacZ5i+FBNi5Vx0dOWuzV0akMgqeKCGiF4naosMTFIInpKj1FHRxI
6zm0aq5teMhHjPSQeapsSAXqIsEY2pFR3txRdewFQbndfB0maxrceMFk04EKSNOYCb2AVG9EKftb
htu2w940KjkIFvm7QUwaaJouglgqs+dqRW1LAr4o7L4Bw2jq7K/fffL488sV+jcWXxr/FassfX8T
oM0d5Aeor5JeVMI2VSKX2P0a6srByRqEAOJqOS00w0U5Dw+tF13kvWk53qg88wGqq7m0p6fd8r1B
A8USveMTieIfW3YPBiMjPqECvuNcqMmK6GI+wwK/Df72Ea+xygChr01D5CEvsJHDFZqt7FAYt1kJ
GN8TlABrIfR1Nip9aOK2N9+1t+IAtpyHtJAgfDYrzBfSipUbHycPNKahhVWhONSsE2lYyyloH7c1
jcWxk4Ivwm106rPKa/SW1euCLEMx5g1DhEvowLHQWkHoUy0yNSzVtG1bRvCn5V7KrSO9tAnDSCRu
IMVSwi4bld/KKvYoSFpAtEzoQInHBWJmNGfygoZKUOQBBB6srS6FCj7TjvRSbtkhi6qZqVJ0XGQL
qgRwUd8EsHZROH96FM2FhfZVS95qY7eGC1k5m2kp7Gi2WEypQuWHs9C8OFogauWRRoEaOc3wefZN
GJ1TxiCRX+oVKh+sZAFRHnnQ84ivE6r5VKgaKjzU6BOEqVSAeGDyIYyhjco2mhuGf1LKWDFsxsbe
rNtlb3uImosLiexHbMYolmjJFb3o9jH98m0BU3byl3iYrAhXvBMoZw0bb7HL731HF1rMVotrxw+
7KWv5oJq2trcr84hLHK9QkYiMoYzo4aaaSFQkjaOwODx2dQAD5a9vajUNN1fq0ZxxR1EsaeevaVy
Q2No808njtHliXHuOfsqzR2hZGGzu5ipJq6ibiT7WebDgTTD0qh//OPH2n7Dj1ZBprvz9okTxQI9
xQ9X/ODXptfkMELkkru16Rk15wIUH/yCxxQ4c8PXUCN65VO+dqqImPTNxgARLAGAYz5o2hPPDgI/
LEk0e0xky0RheLQpGonWWmBoiyA0gng3N6A6UuZ0YgmpFo1E/MCLQGVnhmEyvXNDyroUav3KRFjR
mD1ZwkJFSWijqQIa1jhGvioFMV/wF8iOobVr4kNud0Q3XKnBsYyxQXcWv0HCAY6/hIzFAVteS1tN
emeL3c0Wfx/6eddLlFSZFvl6UHjytj3RiuvldB7OEmpZqKjKYT8k9Tazo0iQ7Eq0IBCpNws+cTTa
Fy83aVadVz0xLOLYKpZGbKZlNiY4SebQDXTd+H20DCAwCrp6hLjvY1nctMuSGzw6n90MrV9g2xgu
iPqe/lpSaxtcSNXHLRxfRaoXxgspw6P0+4oa5Wzot+n2oRp5pMEHfRCz+/5dQX6Gdri9RoqCpMrU
AMn0uyH+R0MLuTWX8rp4QUIqp+/gvXx06rd2kzi6jsa6RpNFTpdL55TxJ1FqaTpg0z637ywsUghB
ZitaQqo9ChqbTCe9rP8nZXtINxnw8hkFFDuNKZNpZ68ZnlBoJA+SgthU6tjsyj2B0jP+GptcnS6H
irsmXQU1PIID4cUMd+L5IIXGzNO3JNpxPuB1oa26Y0vrA34Rzp5plgnJnWHSVhJBH+8H/43leEWu
6lupgoiRsRMjR+kjy+VA/jWHAeyJOUY8AgoYoWZQTrmM4dqboKQ4eBnHCEMTSJfKXQ1n3jjPypk
cFdZ1cNQ2a6Z7r2FpnWPexJmT5VAY0SBmB1Zu6hq5768q88cUX7TYthNRWWphsC65zvMYfTnOtu7
PaIH0JzlaXduy0reQmoJ01NkutbLFy1daqGNWA5NB0xlqVdrIdADFeXZU9d94re/cfjoeWmgpbVO
ePkivNDIzOsdil9Q28aVNHB6Md50SSDn/VOScidDU8N+XRTQECk0Xb7WgkG1wIBNLbRWV1Na0NM
JMYvMK0RLpb5Y1kenSWHcaDSfM1kFtpNFXyftl+sta5+hzc/NcU4LPC5yxG1gmLR46lVlYZJe72i

UgeJ2LUVJmUR2BB+uXwo+tHzfGurG6diVSWALnIS2Z23gFKnZNIX2xRU9VxpRYGyAqZwIM8FLwup
Pp6OkvbLtAa0tz5BjUkxqsNoG0G2rcVRPBooYwamhSo2prCCURjR4bSzk7fMxHBniAm+YHcv4LL4
iIIIp/26ZzAhDQHEG9ihWOkJsBiTtV0a8sHM6lk05rxclqayzFxlG7OoSYPpWpH86AILSWKqlTve
6U1j9fd2OILJmlyarWV5rMZ1d/NzVVCq9lOzzNmMrTgTTJlRsrl9ZEhjZ+8iR/TQ6B9srXldaaV
Ckv4VitCM7qa39Vok/PC/0GJyrmuc96x3YZvakMgogiPWVSY5WPFuooQYhk4GcC2R0ACt1h8Jswc
ZILfCmu/Sr6ZybWdCMNv9U3CAzzDozviQdhdechBegiDe+9jsuCUZ6Ge4Dd5+YNAgK+saIy3iAmG
c6TG1q3U4dqkOWozYQvA7PMxtB9NIOraLb9bdeYHYaydWY3q64+CI1EHeYsN4q7+yuGiqnnglXX
A4HtjzzzlDBgvzlP1IjBkRwfr4WSy2yueMXWmskES/C3PetOPUzpxURz9iWYRHVO/m0lA78Ls6Nk
9K4qOHw3W57fa75NDKGutNprxZEoEfXbUGJ4V75u0pX/fQRkikIax8XBHxxKIG5jDC9bUNyAbT4H
Rc06N6RzxTwtAKrduTqiUvpB3YJe49BAFG5Vg1kNMOWbx2xYuKf0f9CzUPRHu9ZrLZNNggRvW+/
Ma2R2c311f2VLdsjjOaQXMq9l7QNJAEnmxDoaIoIssK//zx/6oPi6MdW119crl0TAYaqEHmZlqxc
YRONLhM6MGIM5rVtkTiXtoqTdLzwwNijc4COWXOFebDxz2O+Ln7IOHoRNO89QwLnpGTvW24mlor
gfrHnhz0qV/V1jOLGfQVQt1tdl5egR+cLNdxIW2CSldXZL/XxRC3jW4GobUWikyil/ta7rfv114H
V3kl1cc2pNK/z/5VCrcvSRt19zEEbsVjuYi1WCe27fx2gtMaBDOZrIwmBe5kBusI5t1XFckCmF4x
9PJqxP4ZDm5xdZLkHLGZ+HHeWymklvnTHjxtjlsHbT39gBtnoZRo8udW0puOOJ7WdG/5b28v8tDX
7HD77p3PXhxdT8NkOMregmEclv7GsWcf0JsGL9jIjoIcbNw2pTa+ReVKrpnH9X2OND6R1rj0bwd
vcLp6q4FajoxHp8j7tKuhtdYWPrggcEG4Tz+IDBWqJQAHPGLFFkzYtErn4gudHkjYiQZLMKt171C
44JlHkKgAaSL96i/OycOKp7WIog/1VcaMc3YnqCXNloo1sbGTkAPhyolH11lsRVVtieaKMaTbGnO
WSdiRvVb4jUslqWPYCuPcnb6NwrSUGu8dWT4o10FJTdhjiqQO2NEzhk91sBCAYABrGG+OZ5VQZ77
mw4afprhLTkyNsuEQM78VUA/YiWExhGsq47/MDNXKfj3bvVZPh0XeobOzk6DN1HBMz5HZMk3UVJt
LWqVZJFTHHzjcoZVR4rMnfHBfGOhwO311VdTaajInnbGdzNr+CwnyuGklIexbyeT1M4bYwNs7V4y
rIv9FhdKPr6lNDm+hDMQESLq4qOjBB85gY1tcaWWqyd1Ld4JlqrM5qPdJr20rT6vLYw/GvIernlq
xsWexzmzqGuzXXV/TaJg6ziJRMEXjmpFB+KuFx3EAnsnnX2UxCp3RatY2YZNbTS2OKMxVT4cYWKV
Um64LDPR5tsAN87U23WAQx6l9jEplmiDyIlQIZPHKUhgRTBFI39WtiI8pJ6l3cLd0OZ6qZrIb0dz
weyXwP4t2p6SK3h92VJyDH468PcdHev+qF+2LEcAnO0vCkk5qjHtVsSrmvkIvFGGYBzSdjJfG17B
khXSV2HN3lRGGTfj26TT2ZqVOjZKql7v73eK2iTqWfaNr6jl5a5L30j5/1xvrW7/++LeHGvxfmEj
jW0ob+kdlilDhge/GASsLkpppaie4KrZzLWmc22qi0prSdJTiIaaIHuvJGECJrJTYTtNV/RgCw6o
uRIXcRA8LV6qg/0tXk8bleBKY3EqMovrb7vHNCbTSgkRHSaCgrjeY6PpLEIXnSQSRpCrjplDMZLb
LK1j/Yz/MT1k5ely5oTYjh1EZCugakpMLChEuobgpr9cjKBp1J+EDam+ZCqTqyaIx5F5UbirKMeG
OxFR9+jVWqWQjZqz668W13NiiUTUBE2+XlZa0/8zTySx3q6WssIm7bmjclumZAKGGu38tmVAMxqy
YSqbBRFVhUjLgVXzPvLYT6NOFYyafEagRAROV7cyRyihtcky9mmiTr5fE9L5L3/Mxjn6Wirxmr9
kt4/6GbGAY/s2Dz3pjXT+68rbRaLQETwWPh9WH8IA9Y0k0nJ8GQyot/QnebzidRCQJ66hSU7iJTK
MMZW0kbYp3YbVGqg9L9XOTqCukRldHr9kqf6QxpYWhsFvpDJlUubT+/6t0BijTIabr+7zo/PMZOW
uT4JxJHTs1b+PKjJEDoyiphZPOsLpU1fygWUz8LtknjYqr1UYO9ENmKCDtnan+00aoV6+76TaFYS
Z9pa/hfbiDtlAyvNfinYPOuQJ9QG88eSUQfxcAdBA7amg2IKoJJYx+w8dDg6ipvGmeRbxtT0uYJ
7tqMbUONUNfn54Akef7YTh+igYam7XqqwcrNacTQuL8LrECj/jKh50LLhGz4BqxZUTmw0jLdiWyz
SlsdvrUqVJMqtq8TSDfnf8orP98LpzMuC5Qb0aHM3wWW2sV737ndl4PH47xjW4KZh1rY4nqv0rAA
no+kVazErWKqYd29RxcxzGEQtz8/TiM905p0+JRTKrxVw93W9um8K0cDTdwKZzTAX3J7o3NZnt8n
pqRNU+7jhrUGKdGMOJ8LdwVy960SV6OI0EECVavG5KJY6JMFZQQLiBwkGU37xd03A0oIxY3RhWWW
esen3czWN7e6iRXOnu2jpeM9FqMNTrbpGDR1dJto8Dfhx4NfiyjoSmI3TKsuXl+jY3ywjgoPJOFM
qa11BBQiaLzubYwhdD/YH3OjxDjEs4tqkzfnZC9Jq6ZTpV/q4qJ+sGn6UthJBkb3JJJxHthEsEoG
EBDSR2pes4loPsDIazRWm43fAaa9hLVCsirnj9pbytrVtlibaByZlrkkbVw+5Z6BETrn+fSwz8kg
GVlq394oc+0jyrjfxAk0++JtzkszfXpaqc1RTciM19yC20jA801RSEclh4aXaJK6KWOLlhHk14eE
WtNklED15bupjF9DRV7ic8veM7PT9Kswj2TkzBUGzREeCoIPaUcUlaVTwKqZmnj9vHwvM6dcdW2X
XaLlEKRkp199k9KWIG6suRK/1NJFMdououbT7VOqNMOzUhnYeD0Dm20cXP8XUbreGAvz16ZFn+5f
qb4wiC1+JR3Oo3NdTcl0s7js74w10WS6qoByXSWyNPEu+Ga6ZBcP7cQJ586imTjRU2lnoOiI+c3j
ZST71wWnHA6icTqkylJ5r1M4YbT8dSUHYqqb4xGE44MjppSchYPHToKpAWvEqBKcBIOK2QEDz11j4T
P9StfVqWnH46icV3R5kuq/JN6EnSue91ZPKWgzfZGk07aKrGiAyKxl8ER/Pz/9a282N031rx19bcVlu
FErYHDB0QLDLAorTcv5MzKrmcq3Wq8iQh34JVCrlLVta67cw5p03aRExt4wgt200yn3ZkLlKxO
DxtD0ieOAZyYEV3RsJRkTW8Zk2p5paVfKpzJVatG50iRM+z+7zzqP8ibiMSWdHigMM2AzKM9tPA5
DiRDeuhXuzuP2UzhglmwHk43MUvOZSgQ0GsVTplYxrKBFVJ5NVufn2O+ToxshKhCam2z4fbXz+jD
13U8VuNmhdvNmQBH4uR63A6jOLzezJRK76ixlQjRcW9sIBjW2iKRfs21VImfjTSersSaZrT0gBR
CNKBDILTnl4wdNI4x2MDedUuw95cY5VeJQZ1aTmBCuPfiaqS1UdibRQshWkk56rUfFOMjW6FIHaX
lmgyw6w0Yi/8fS6lpFDXj99oy02cb3Wakp+ae1cb6jjf/wNnFdHpZV+kBAHOkRkCUwENTQHYS4t
nglVRqrLgBrFcLFQNDpEQneFqOVQYyaUhUZ8SsK6ri++oK32dacLeLIZHSVVOXwt0dch3ZJHJME
LOssHaGm3ysPmEEqrxua0uFs5Rr168m/NEp6aBAaSJfMxYROMjXnZOqtFqnBPiPRZPPi16G6bWJM

a3OF5dpQiE0STq/fAgmcwIFAvXltfkmhtvaQ1Vsrh7xhtKSURiVZrSWocXTRLum2lsV2pH3oa1rx
l1Y1vsSIHrabd00BnLwcOHolOA4fXzDrBetAsrnYjNe1MbNBFlaubcVJVqGfQGMqlLRlQYaj944w
FWV2LZFppLVREbVojAJHVAWrSnNWyrQuHpdGZz1obLoUtrpuH5FXaPuivMu2R8ovPrhMy6zbK2QZ
qzKdYbbdly4tFntbGura6+rR5NMgedxw9flyrO1U9l0PRlGFLehgcSB7Mk+97nXHGNCTzfpI6JL
OURnmohPVZPQnGBveBNjqiaawRRGND6oWuMGlNnRkOM9v1WR/TpW0MzN9+nWqLvpQIrxheeCk4iv
N3X9gaZKWUrBJzu3rJpE5UHoUrtZyoEfi+zxBtF7aZxhaI8FiTWNcRvK8mX4mqm5caoXlmyfmgAv
ZvuOk2ObS8artmkjjjVaV71fKKeq8MZidxsXBHFBxkCFG2kxMIIkHc0mLolZNNnVHKHDx0xNQIFY
WFFBiOqxfLjORp7+MkfxcRoGiAqDGg/Mnme1IMp4yWmFfDUCmjOqltTUDpnw/Puid4YXFxy0wdqg
ltGmtOcKRJdzWGBFFBxeyOEufzaYDU7TjHF4G1HYyOvjcyDNbe83NXN//W9vnlQzD92i9dnU3Wh2
+eUGiqJJKGyBNRAq8+TDU01BNws6puWHKkjIO0iWjhwiCtzTiKsejiY5XcIF+ZdO5XR8NWkU1GA0
NUsY5m3HjdaPolhS1KJD63rAyZVcOzv59/9mmytGWzws+Sdikt0lN2MquOo8rIx6Kndjjf/Ak7w5
fn7fXacY4bqqeznInaCovEBvdoDoF+hj21/3LtdRZJ2mvz1K9KdJ2kGBE/abefaURNlEHDtM6aL6
qO0Yq09TppqGyhL4GwPr6xa91cHZugvwLioiJge0/W1WlHJ7bpzFcharSdzcnhVqbDPZtVGcHZsVG
G3bqWafxgyF5ZZxES2Nb+0yJS3m7qyLicSTBT6pY26UmMXRayvQS+zelavKc62PVXOWseDZ5A2Ld
c16+XDpflNP/1M2NCXLbLef889rw43eCd5lwZA963T7Z5VVWEg9G5OW+eAj5WmCxuNNTwURbtUXG
AbgQktJlONN/GUzWhuhtpBQwWNGcwisAakUAaxHSlVeIhJqlxVHChVWzd9HWreVjPwMxgzOaD28M
4//wUGgseByHRLI9eYVqxtPcvalVmALEproJUhuq5sNVOobJGQuB2je20opqoworkibxqSzwsKh9
10/Q1ydHVi9aaRyTMTcRPOdjNLqcuompCLS+jIzD4XJeJonZpZLZ9ZZqCb1XU7/MbaOmtK3pvEI1
pm+N52419ihHbcipz3V8cwVPfIKhMNV9wuI2muCoj4xWXYuFfiG+ariBteXJzn992RqnRLEkkCyr
BqIvEisWXPsbML6VsfXyFC2ny2K/B27NjNRZ2sYwlnHJzJcGHT/N89q411uL7x5onJZygeXVXW2d
GtratI3R5tAUE0C7XpXF9hZtPJNKY0vX5O+CB3CdVaP0pn+VH04uF7vSyZNDAl5QjdAePgee8Itv
qwhsAJaV1h4IzEDo8P/2dmbmkSGSoLc41cdNFFjKpQKBVftXHAi1CLa5PMZFjY5NFuLKUvS05+Zf
Ou01UbqNF6m13IdrpLOl/Jzm7dwgRxLdgoEO7R4UPL8s9fuJXaSlx7aF5fkm4K2qpCJfXlORDuyC
zFWUhmVlK1j1IrQdJOryMoh+sbjYaysbFhavW6PYGJbKo4XAcW4PX6wsI0qmHBedBxpyprW8dj16
2Eeq3FQWlD4wuNsL60ZRHqUTSa+tancKplVHZItCZltEe54cqEuK+TJJIXsKAhHULYM3J5J909dm
TThS7q5EKNFr25/vJVv//y8FlrrL/327+1NNzYeC29F4w0aaOLjtfUYGHMmRGcY+qGNRF5L/Id8c
BAH5uMJ7bqwbS6bPaZpG0t4ewZUuVqRTAlvh+Hc1KteWuTYODms1QbTFU9MzxwGmU8faFyfD02Ya
pYy51y4mY57cyzIjE889YVZDuL0tTbW9QPjJTKCEURqW6LO87STCNT8ndC8nul4Ajvdt6xT2Sww
HFAxc0pMBIf5fXlois6q5SrIjMMgdgxfEs7ZsEXNctx6VBXKEzdYDvMnLSjewVD1WhlIw8DnK
TGvDmWmdRN54mlLmtx8QHry3H0xJHeaMxOMBxRZfNgyH/0he8BBKGcVjXwsjBoBeIj9so0nMuIX2
HXtpe3kMhEm5fMoKxT3DVCb1J3v9Z11hGBpYSMop9nf/1M2dGXxVhvu/6m105H43ndp1p36gGkV6
VG0loPPY230Fii+1hUbQBdtqanuRg9YVPG/ZmNM2SQImepedmKXlpb7TaSqKpIicvSvopUu7vtuj
6xJVGnQUgktQuuNdb9tcgCnHKiRovE6bzzpV+iAC1qohFIEbONRwGa7OU1ZrPwTGVNESAGQbbTy
ZhQZk1M06c2GGFs2lUZblQ2kpH0ZSWQujBGRxdXpMvhBQYnwtlqX/uRIHlOt7CtffYCB9EA6jguo
vGULwbwyA6RQydeC09WgFMkCoXWA3VYI98kLrdD/NGdIglk47sSiuu3e23uuayLwHdd6dFGRd14b
mB5+qia4iW6PKqCqbS40rb4wpHyiylUlBz04xjNzjCHVODmfcMzMAGVnnMzFa75AlfUtVl6Tgbqb
az2Ye+c95bCff56melsa6urbwGzaLKdHuYkjk7A2ss6iJKhJAiFZdNqChyQeKmis7WOP+ydj+6vb
5BG4cE+1I0tVVD5JawWiF93BJWK+A/SR3P2rgIX8SoeioMY+vrFkkDSTQzNLLK2DvY+Iax007dL1
TZj6LseF89HJzfhS+ZuxQp9W4ratROJ6OQtK7lhK3bI7uFURBSJaVG+8brei6/UgCFmEfHQZ4WI7
nx5ptkZTjVDM2HTJ64flPnunlgxQyTKWqbwrbi4XVnc3obWTJrSm2sb6jEiTXoEpNQwSspoKEi6u
hLcYkjCwYghFojtRpnzfQdpIhSMxPqQVnRWCz1TaJDnowrOgWimeoyquLj+zk3zgl19PQg4Wky6m
XIY/g8Ff/GBby9gxwXb9XVDILN56pp501SyWJtcfOm5Wetsb7+m181Hw71ZUWlCHrdw6JemQyXpL
B6wJEijaoaQgupsd0FMDinwXGRb9mheSXwfcxYG2EjWUtVUk0iDq8xbqgrG6JXim6hKr/WhFEXQL
cWU4+DMTTXEU1laW3lLqYz1sh7bh705Oxznx+hZ013glplbCKk7gsLBm4Q28421U5w+D0IxqpjIG
F0TNBA8ihkKcNcAd3pGYRwCn2lEOH+9dqBVFaoK55FppA3Q+z1WdeL3pd2x2hqCC3NJW3F1Do8Cn
cnEYGEearSE7O4ZZ1FS6YCa7moEkSXNugqCo2hpYgNr3RVxVw2b4uegsFMR4z8UAqB8aQGulTCem
7Iq9K2oYc/x3qvwV1lWl2Pitj1TyUKWxVHNN11HU1Hff9YHHDROskeu0BZu+nYtpcUvbn5T3zgV3
69eaZs6Rkf3YRD9JLxdHpiZYglLlXyFjpS2TqLqk4RP/St3nU4bxDVeMNG0OmJcDclWHSQZyK8tmS
SMZ9yU25nVs3hgB35nks4M610QjVlei8xlrUJrrD8zY+/XM5KUSObPOMWHbN95Slys2+Z4tl4xHD
7Uln1Glmv5YjaKgzLYchLHVZVDDxv157ZziGYRtYP+bfRQjQu55ebb5clDR5701+ymoB0YHDUX4Y
CaVsIl656Gur0n8aCawePwApureZEPdsJgjdeZd+pBb8B1GzKZybQ4aT/xlSTWOBqNlduKLjCj4l
w/rm7kjLeeksOKOlAyBS3I1qlbYzc4z1KbJkqNOp1tBj6ZttDCWSZNZ4ds1s78/LE4cXbOheyglZ
Ca/8IzaUvPeGQNH+ilo+G64ih9ZiXcLKQymDyUJpAleHC8kcFqNbHRhzRxxYPOvBqrZxNCzxBRVU
V+SjU8NYqyjRTSCj3r9xJd7VjrGkTO1jId0GdikT+VqGbIzLFu4szFpUuQEVBcv7v0xRcQOVUW0z
aNTC2NR/YAGMNk3FLJkAKjy102KhIW3m+weYsCWHFvBz9qTa5vwUdFGVWp2ntLLHXBFPiz//Q5Ja
J3DCuVvpaErB8HInQU/TG7LKYq29rdkp4kx+yFSBQOxXOcjsZx1ylwvWT/WMc9yyp9bq5an6RR38
n35tBh+za3NDEoohof+KXTsogQTaCZMmvy+fLkoqhlvj/PmSpq16fhrm0c2BImnD1TKa3QzhLFwh

```

sHnHQICqjp62bGyXQZWsc6HzuiuSxf/pXf/v3VZ9KWnvHIOlgcvBwEclfar0wriAZpagK1EZJpxE
iVUati5UXTij+jEzqXqYRDitroce0gAL8HIQViQ8EN0VZGRHrXjMe0jrCPGkJOitGBI33coIioqX
TjaNJ9YJ1VjYhem5bmZfeFF7cdY2t8JV4oVvoZUJv65jlQ8HDQi/A1Yl8XNqmoENNKu+iSZXjunm
YdBoTALFr7XEowv+vOPfL4U4dmUjTppuKouTrspC4wwbZTxLFU4zW9dLqiPq8Ml+UzT/2stfQa2x
GUKr6acAmTA4V0qHNIxWBN0tBF1vFW0tQ2d58W47aGBEAmbYXhcJ94pgYLSrBpiVzXbg06A2JIZn
sQ/1lYYxsZqKraTn53juqqD12BuFj/J8c4NXzwNC/y3txPPdO29IxH1oWFpUsG82uytjamxAg4md
T59UXBidZvmvL5HFL5qI4ASm38AQ8LVk5tC5Y4KA8PbMkWCs+kKK12kbM0t46uRfnM2udsMUrWLT
GZkf0dCSUDhoh3MrstfLF9Ouc/7yzZvuPkjgq+1UL+poancGHphDVXqFWHazIBnjcY7pZt5+i272
DASY1VldoQmawfDhF7nhrKXnvWZCalbDRhdeM//MO12kVPtcvbyP+bp/MvxRylobo407VGk3ZKJa
pUJUk6M6oA+yUyfxpfwKSZEI4Sxk2MgqI0MbRbU2OySIQs6lLjJpLnCwqx615bLEIuo1xskmmvgM
4yg5EuxVTXWVXdDW9eH9e+Xd3mz00iMxey/lmpl1Temct2Db+LVOoafPdrWJsyl/cO9frZJ59pW3
rGI2sxHJ+6NL+gW617yqgZDSdx1QMXGosq0pX17LDZxz81vVmlDbdvrnQeji/N5gRMosDb8fbJq
oL5J62tu6y73D1jeMu5xFhd/Z8Shsp+GihSzqGM0En+NJLLw3GNODD99EA436tIm+sTdm8bUcCAD
Zg7Wix1RoVeGDUq9PJmuRz87K+fFD2PXC9OcXZeXAYzoPrV0mVLuk6ALfe89dcs/DD+nnsUlnGj
G149x0D5kZqkeM+FlrbQ410gqlld190qnOwOMKxk5nlwZiCCexDrcDDdqmXdvMkQ7/s7ZGWWO8XP
xbOATfKseF2uhIh3MTnD43B/prA4c8sVqfdMuqisqCXSkW/z3br8gMO14/zQhnz189Kz/abRy2GQ
oRXXf+pw9/dPqsNtY3ftfr+sPxSLVz8gGXTUGVHg+fLfJSWSVV1RyTwhhYn9uym2gc2JIdCctZxn
oFy5J9hQFurjat9O95kn6JZcLyJeuOmcZW22OJf854ZG+thAh5wuK8nHfB+cZnbNUCGzuknlZqaL
bInWmTpbGxBA7lwrYd4aCuS96b1w7o3ILMbT6BQP7x2jKWALWHOxE7pBP5zD9+Xue3EQndNkrIlq
lq05dqItjTfjTb19R+pGiScpu/q8ZlnZpKdvM70/tp4X7P2nsJYdE9MHjV4u4bAn8nTCDaITZ3URc
oKjmx6HcqZ7rHJ41b4wlg2cc2HY3JdID7T63C9JOecdq8hmUFh5Xwuvmy5mzk5Eq6rGdxtLplDWA
le5K1fjhHo/xJgAHavyI3ijlo8AAAAAE1FTkSuQmCC',
height: 150, width: 180, top:290, left:
50}}]"></e-cell>
    <e-cell [index]="2" value="Gender"></e-cell>
    <e-cell [index]="3" value=": Female"></e-cell>
  </e-cells>
</e-row>
<e-row [index]="13">
  <e-cells>
    <e-cell [index]="2" value="Contact
Preference"></e-cell>
    <e-cell [index]="3" value=": Phone"></e-cell>
  </e-cells>
</e-row>
<e-row [index]="14">
  <e-cells>
    <e-cell [index]="2" value="Email"></e-cell>
    <e-cell [index]="3" value=":
mary@gmail.com"></e-cell>
  </e-cells>
</e-row>
<e-row [index]="15">
  <e-cells>
    <e-cell [index]="2" value="Date of Birth"></e-
cell>
    <e-cell [index]="3" value=": Dec 8, 1989"></e-
cell>
  </e-cells>
</e-row>
<e-row [index]="16">
  <e-cells>
    <e-cell [index]="2" value="Department"></e-cell>
    <e-cell [index]="3" value=": HR"></e-cell>
  </e-cells>
</e-row>
<e-row [index]="17" [height]="40" >

```

```

        <e-cells>
          <e-cell [index]="2" value="IsActive" [style]="{
verticalAlign: 'top' }"></e-cell>
          <e-cell [index]="3" value=": True" [style]="{
verticalAlign: 'top' }"></e-cell>
        </e-cells>
      </e-row>
    </e-rows>
  </e-columns>
    <e-column [width]=20></e-column>
    <e-column [width]=280></e-column>
    <e-column [width]=172></e-column>
    <e-column [width]=160></e-column>
  </e-columns>
</e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    created() {
      this.spreadsheetObj!.merge('B2:D2');
      this.spreadsheetObj!.merge('B11:D11');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold',
verticalAlign: 'middle', backgroundColor: '#1167b1', color: 'ffffff' },
'B2');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold',
verticalAlign: 'middle', backgroundColor: '#1167b1', color: 'ffffff' },
'B11');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold' },
'C3:C9');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold' },
'C12:C18');
      this.spreadsheetObj!.setBorder({ border: '1px solid #1167b1' },
'B2:D9', 'Outer');
      this.spreadsheetObj!.setBorder({ border: '1px solid #1167b1' },
'B11:D18', 'Outer');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Image

The following features have some limitations in Image:

- Corner resizing option in the image element.
- Copy and paste the external image.

Chart

A chart is a graphical representation of data, that organizes and represents a set of numerical or qualitative data. It mostly displays the selected range of data in terms of **x-axis** and **y-axis**. You can use the [allowChart](#) property to enable or disable the chart functionality.

* The default value for the [allowChart](#) property is `true`.

Types of chart

The following types of charts are available in the Spreadsheet.

- * Column Chart
- * Bar Chart
- * Area Chart
- * Line Chart
- * Pie Chart
- * Scatter Chart

Insert Chart

You can insert the chart by using one of the following ways,

- Select the chart icon in the Ribbon toolbar under the Insert Tab.
- Use the [insertChart\(\)](#) method programmatically.

The available parameter in the [insertChart\(\)](#) method is,

| Parameter | Type | Description |
|-----------|-------------------------|---|
| chart | <code>ChartModel</code> | Specifies the options to insert a chart in the spreadsheet. |

The available arguments in the `ChartModel` are:

- `type`: Specifies the type of chart.
- `theme`: Specifies the theme of a chart.
- `isSeriesInRows`: Specifies to switch the row or a column.
- `range`: Specifies the selected range or specified range.
- `id`: Specifies the chart element id.
- `markerSettings`: Specifies the marker settings. The marker is used to provide information about the data points in the series and is currently only applicable to the line chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getFormatFromType, ChartModel } from
'@syncfusion/ej2-angular-spreadsheet';
import { chartData } from './datasource';
@Component({
```



```

imports: [

    SpreadsheetAllModule
],
standalone: true,
selector: 'app-container',
template: `<ejs-spreadsheet #spreadsheet (created)="created()">
    <e-sheets>
        <e-sheet name="Book Sales">
            <e-rows>
                <e-row [height]=30>
                    <e-cells>
                        <e-cell value="Book Sales 2016-2020" [style]="{
background-color: '#357cd2', color: '#fff', font-weight: 'bold', text-align:
'center', vertical-align: 'middle' }"></e-cell>
                    </e-cells>
                </e-row>
            </e-rows>
            <e-ranges>
                <e-range [dataSource]="data" startCell="A3"></e-range>
            </e-ranges>
            <e-columns>
                <e-column [width]=110></e-column>
                <e-column [width]=100></e-column>
                <e-column [width]=100></e-column>
                <e-column [width]=100></e-column>
                <e-column [width]=100></e-column>
                <e-column [width]=100></e-column>
            </e-columns>
        </e-sheet>
    </e-sheets>
</ejs-spreadsheet>`
))
export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    public chart: ChartModel[] = [{ type: "Column", range: "A3:F8" }];
    data: Object[] = chartData;
    created() {
        this.spreadsheetObj!.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
        this.spreadsheetObj!.numberFormat(getFormatFromType('Currency'),
'B4:F8');
        this.spreadsheetObj!.merge('A1:F1');
        //Render Column chart
        this.spreadsheetObj!.insertChart([{type: 'Column', theme:
'Bootstrap5Dark', range: 'A3:B6', id: 'column-chart' }]);
        //Render Line chart with Marker
        this.spreadsheetObj!.insertChart([{type: 'Line', range: 'A3:B6',
markerSettings: {visible: true, shape: 'Circle', isFilled: false, size: 10,
border: {width: 2, color: '#3cb371'}}}, {id: 'line-chart'}]);
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Delete Chart

- If you want to delete the chart, just select the chart, and then press the Delete key.
- Use the [deleteChart\(\)](#) method programmatically.

The available parameter in the [deleteChart\(\)](#) method is,

| Parameter | Type | Description |
|-----------|--------|--|
| id | string | Specifies the id of the chart element to be deleted. |

Chart Customization

Chart feature allows you to view and insert a chart in a spreadsheet, and you can change the height and width of the chart by resizing and moving it to another position.

- You can change the height and width of the chart by resizing.
- You can change the position of the chart by drag and drop.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getFormatFromType, ChartModel } from
'@syncfusion/ej2-angular-spreadsheet';
import { chartData } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()">
    <e-sheets>
      <e-sheet name="Book Sales">
        <e-rows>
          <e-row [height]=30>
            <e-cells>
              <e-cell value="Book Sales 2016-2020" [style]="{
backgroundColor: '#357cd2', color: '#fff', fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }"></e-cell>
            </e-cells>
          </e-row>
          <e-row>
            <e-cells>
              <e-cell [index]="7" [chart]="chart"></e-cell>
```



```

        </e-cells>
      </e-row>
    </e-rows>
    <e-ranges>
      <e-range [dataSource]="data" startCell="A3"></e-range>
    </e-ranges>
    <e-columns>
      <e-column [width]=110></e-column>
      <e-column [width]=100></e-column>
      <e-column [width]=100></e-column>
      <e-column [width]=100></e-column>
      <e-column [width]=100></e-column>
      <e-column [width]=100></e-column>
    </e-columns>
  </e-sheet>
</e-sheets>
</ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    public chart: ChartModel[] = [{ type: "Column", range: "A3:F8" }];
    data: Object[] = chartData;
    created() {
      this.spreadsheetObj!.cellFormat({ backgroundColor: '#357cd2', color:
'#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
      this.spreadsheetObj!.numberFormat(getFormatFromType('Currency'),
'B4:F8');
      this.spreadsheetObj!.merge('A1:F1');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization of line chart markers

Using the [actionBegin](#) event, you can change the shape, size, fill color, and border of the line chart marker. In the following example, you can see the modified marker appearance, such as shape and size, while creating the line chart with UI interaction.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getFormatFromType, ChartModel,
BeforeChartEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
import { chartData } from './datasource';
@Component({
  imports: [

```

```

        SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-spreadsheet #spreadsheet (created)="created()"
(actionBegin)="onActionBegin($event)">
        <e-sheets>
            <e-sheet name="Book Sales">
                <e-rows>
                    <e-row [height]=30>
                        <e-cells>
                            <e-cell value="Book Sales 2016-2020" [style]="{
backgroundColor: '#357cd2', color: '#fff', fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }"></e-cell>
                        </e-cells>
                    </e-row>
                </e-rows>
                <e-ranges>
                    <e-range [dataSource]="data" startCell="A3"></e-range>
                </e-ranges>
                <e-columns>
                    <e-column [width]=110></e-column>
                    <e-column [width]=100></e-column>
                    <e-column [width]=100></e-column>
                    <e-column [width]=100></e-column>
                    <e-column [width]=100></e-column>
                    <e-column [width]=100></e-column>
                </e-columns>
            </e-sheet>
        </e-sheets>
    </ejs-spreadsheet>`
    })
    export class AppComponent {
        @ViewChild('spreadsheet')
        spreadsheetObj: SpreadsheetComponent | undefined;
        data: Object[] = chartData;
        created() {
            this.spreadsheetObj!.cellFormat({ backgroundColor: '#357cd2', color:
            '#fff', fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
            this.spreadsheetObj!.numberFormat(getFormatFromType('Currency'),
            'B4:F8');
            this.spreadsheetObj!.merge('A1:F1');
        }
        onActionBegin(args: BeforeChartEventArgs | any){
            if (args.action === 'beforeInsertChart' &&
            args.args.eventArgs.type.includes('Line')) {
                args.args.eventArgs.markerSettings.shape = 'Triangle';
                args.args.eventArgs.markerSettings.isFilled = false;
                args.args.eventArgs.markerSettings.size = 10;
            }
        }
    }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limitations of Chart

The following features have some limitations in the Chart:

- Insert row/delete row between the chart data source will not reflect the chart.
- Copy/paste into the chart data source will not reflect the chart.
- Corner resizing option in chart element.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Formatting](#)
- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)

Rows and columns in Angular Spreadsheet component

Spreadsheet is a tabular format consisting of rows and columns. The intersection point of rows and columns are called as cells. The list of operations that you can perform in rows and columns are,

- Insert
- Delete
- Show and Hide

Insert

You can insert rows or columns anywhere in a spreadsheet. Use the [allowInsert](#) property to enable or disable the insert option in Spreadsheet.

Row

The rows can be inserted in the following ways,

- Using [insertRow](#) method, you can insert the rows once the component is loaded.
- Using context menu, insert the empty rows in the desired position.

The following code example shows the options for inserting rows in the spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet';
import { Component, ViewChild } from '@angular/core';
```

```

import { SpreadsheetComponent, RowModel } from '@syncfusion/ej2-angular-spreadsheet';
import { dataSource } from '../datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false"
    [showSheetTabs]="false" [showRibbon]="false">
      <e-sheets>
        <e-sheet>
          <e-ranges>
            <e-range [dataSource]="data" startCell="B1"></e-range>
          </e-ranges>
          <e-columns>
            <e-column [width]=20></e-column>
            <e-column [width]=90></e-column>
            <e-column [width]=220></e-column>
            <e-column [width]=90></e-column>
            <e-column [width]=140></e-column>
            <e-column [width]=90></e-column>
            <e-column [width]=100></e-column>
            <e-column [width]=100></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;
  // Rows model that is going to insert dynamically
  rowsModel: RowModel[] = [
    {
      index: 9, // Need to specify the index for the first row
      collection, the specified rows will be inserted in this index.
      cells: [{ value: '' }, { value: '8' }, { value: 'Northwoods
Cranberry Sauce' }, { value: '3' }, { value: '12 - 12 oz jars' },
        { value: '40.00' }, { value: '6' }, { value: 'false' } ]
    },
    {
      cells: [{ value: '' }, { value: '9' }, { value: 'Mishi Kobe
Niku' }, { value: '4' }, { value: '18 - 500 g pkgs.' },
        { value: '97.00' }, { value: '29' }, { value: 'true' } ]
    }
  ];
  created() {
    // Applies style formatting before inserting the rows
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'B1:H1');
    // inserting a empty row at 0th index
    this.spreadsheetObj!.insertRow();
  }
}

```

```

        // inserting 2 rows at the 9th index with data
        this.spreadsheetObj!.insertRow(this.rowsModel);
        // Applies style formatting after the rows are inserted
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'B3:B12');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D3:D12');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'F3:H12');
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Column

The columns can be inserted in the following ways,

- Using [insertColumn](#) method, you can insert the columns once the component is loaded.
- Using context menu, insert the empty columns in the desired position.

The following code example shows the options for inserting columns in the spreadsheet.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, CellModel, getCellAddress } from
 '@syncfusion/ej2-angular-spreadsheet';
import { dataSource } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()" "
[showFormulaBar]="false" [showSheetTabs]="false"
[showRibbon]="false">
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range [dataSource]="data" startCell="A2"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=90></e-column>
          <e-column [width]=220></e-column>
          <e-column [width]=90></e-column>
          <e-column [width]=140></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  `
})

```

```

        </e-columns>
    </e-sheet>
</e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    data: object[] = dataSource;
    // Cells model that you are going to update in the inserted 5th column
    dynamically
    cellsModel: CellModel[] = [{ value: 'Unit Price', style: { fontWeight:
'bold', textAlign: 'center' } }, { value: '18.00' },
    { value: '19.00' }, { value: '10.00' }, { value: '22.00' }, { value:
'21.35' }, { value: '25.00' }, { value: '30.00' },
    { value: '21.00' }, { value: '40.00' }, { value: '97.00' }];
    created() {
        // Applies style formatting before inserting the column
        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A2:G2');
        // inserting a empty column at 0th index
        this.spreadsheetObj!.insertColumn();
        // inserting 1 column at the 5th index with column model
        this.spreadsheetObj!.insertColumn([{ index: 5, width: 90 }]);
        let rowIndex = 1;
        // Updating the 5th column data
        this.cellsModel.forEach((cell: CellModel): void => {
            this.spreadsheetObj!.updateCell(cell, getCellAddress(rowIndex,
5)); rowIndex++;
        });
        // Applies style formatting after the columns are inserted
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'B3:B12');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D3:D12');
        this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'F3:H12');
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Delete

Delete support provides an option for deleting the rows and columns in the spreadsheet. Use [allowDelete](#) property to enable or disable the delete option in Spreadsheet.

The rows and columns can be deleted dynamically in the following ways,

- Using [delete](#) method, you can delete the loaded rows and columns.
- Using context menu, you can delete the selected rows and columns.

The following code example shows the delete operation of rows and columns in the spreadsheet.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { DataSource } from './datasource';
@Component({
  imports: [
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false" [allowDelete]="true"
[showRibbon]="false">
  <e-sheets>
    <e-sheet name="Sheet1">
      <e-ranges>
        <e-range [dataSource]="data"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=90></e-column>
        <e-column [width]=220></e-column>
        <e-column [width]=90></e-column>
        <e-column [width]=140></e-column>
        <e-column [width]=90></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
    <e-sheet name="Sheet2">
      <e-ranges>
        <e-range [dataSource]="data"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=90></e-column>
        <e-column [width]=220></e-column>
        <e-column [width]=90></e-column>
        <e-column [width]=140></e-column>
        <e-column [width]=90></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
  }
}

```

```

// deleting the rows from 8th to 10th index. To delete row, the
third argument of enum type is passed as 'Row', the last argument specifies
the sheet name or index in which the delete operation will perform. By
default, active sheet will be considered. It is applicable only for model
type Row and Column.
    this.spreadsheetObj!.delete(8, 10, 'Row', 0); // startIndex,
endIndex, Row, sheet index
    // deleting the 2nd and 5th indexed columns
    this.spreadsheetObj!.delete(2, 2, 'Column', 'Sheet2');
    this.spreadsheetObj!.delete(5, 5, 'Column');
    this.spreadsheetObj!.delete(0, 0, "Sheet"); // delete the first
sheet. sheet index starts from 0
    // Applies style formatting after deleted the rows and columns
    this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'A2:A8');
    this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:G8');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations

The following features have some limitations in Insert/Delete:

- Insert row/column between the formatting applied cells.
- Insert row/column between the data validation.
- Insert row/column between the conditional formatting applied cells.
- Insert/delete row/column between the filter applied cells.

Hide and show

You can show or hide the rows and columns in the spreadsheet through property binding, method, and context menu.

Row

The rows can be hidden or shown through the following ways,

- Using `hidden` property in row, you can hide/show the rows at initial load.
- Using `hideRow` method, you can hide the rows by specifying the start and end row index, set the last argument `hide` as `false` to unhide the hidden rows.
- Right-click on the row header and select the desired option from context menu

Column

The columns can be hidden or shown through following ways,

- Using `hidden` property in columns, you can hide/show the columns at initial load.

- Using `hideColumn` method, you can hide the columns by specifying the start and end column index, set the last argument `hide` as `false` to unhide the hidden columns.
- Right-click on the column header and select the desired option from context menu

The following code example shows the hide/show rows and columns operation in the spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { dataSource } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[showFormulaBar]="false"
    [showSheetTabs]="false" [showRibbon]="false">
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <!-- Hiding the 1st and 2nd column index through
property binding -->
          <e-column [width]=100 [hidden]="true"></e-column>
          <e-column [width]=100 [hidden]="true"></e-column>
          <e-column [width]=80></e-column>
          <e-column [width]=80></e-column>
          <e-column [width]=80></e-column>
          <e-column [width]=80></e-column>
          <e-column [width]=80></e-column>
        </e-columns>
        <e-rows>
          <e-row [index]=2 [hidden]="true"></e-row>
          <e-row [hidden]="true"></e-row>
        </e-rows>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
  }
}
```

```

    // Unhide the 2nd index hidden column
    this.spreadsheetObj!.hideColumn(1, 1, false);
    // Unhide the 3rd index hidden row
    this.spreadsheetObj!.hideRow(3, 3, false);
    // Hiding the 6th index column
    this.spreadsheetObj!.hideColumn(6);
    // Hiding the 8th and 9th index row
    this.spreadsheetObj!.hideRow(8, 9);
    this.spreadsheetObj!.cellFormat({ textAlign: 'center' }, 'D2:H11');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Changing text in column headers

Using the [beforeCellRender](#) event, you can change the text in the column headers. In that event, you can use the `e-header-cell` class to identify the header cell element and update its text value.

The following code example shows how to change the text in the column headers.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet';
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, CellRenderEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet
(beforeCellRender)="beforeCellRender($event)"></ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  beforeCellRender(args: CellRenderEventArgs) {
    // Condition to check whether the rendered element is header cell.
    if (
      args.colIndex >= 0 &&
      args.colIndex <= 10 &&
      args.element.classList.contains('e-header-cell')
    ) {
      let text = 'custom header ' + args.colIndex.toString();
      // Add the custom text to the innerText of the element.
      args.element.innerText = text;
    }
  }
}

```

```
}  
}  
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';  
import { AppComponent } from './app.component';  
import 'zone.js';  
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

Filter in Angular Spreadsheet component

Filtering helps you to view specific rows in the spreadsheet by hiding the other rows. You can use the [allowFiltering](#) property to enable or disable filtering functionality.

* The default value for `allowFiltering` property is `true`.

By default, the `Filter` module is injected internally into Spreadsheet to perform filtering.

Apply filter on UI

In the active Spreadsheet, select a range of cells to filter by value of the cell. The filtering can be done by any of the following ways:

- Select the filter item in the Ribbon toolbar.
- Right-click the sheet, select the filter item in the context menu.
- Use the [applyFilter\(\)](#) method programmatically.
- Use `Ctrl + Shift + L` keyboard shortcut to apply the filter.

* Use `Alt + Up/Down` keyboard shortcut to open the filter dialog.

Filter by criteria

The [applyFilter\(\)](#) method will apply the filter UI, based on the predicate and range given in the arguments.

* The [beforeFilter](#) event will be triggered before filtering the specified range.

* The [filterComplete](#) event will be triggered after the filter action is completed successfully.

The following code example shows `filter` functionality in the Spreadsheet control.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { tradeData } from './datasource';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (dataBound)='dataBound()'> <e-
sheets> <e-sheet> <e-ranges> <e-range [dataSource]='tradeData'></e-
range></e-ranges><e-columns><e-column [width]=100></e-column><e-column
[width]=120></e-column><e-column [width]=96></e-column></e-columns></e-
sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public tradeData?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.tradeData = tradeData;
  }
  dataBound() {
    if (this.spreadsheetObj!.activeSheetIndex === 0) {
      let departments: string[] = ['Sweden', 'Canada', 'UK'];
      let predicateList: any[] = []
      departments.forEach((department: string) => {
        predicateList.push({ field: 'D', predicate: 'or', operator: 'equal', value:
        department });
      })
      this.spreadsheetObj!.applyFilter(predicateList);
    }
  };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Filter by cell value

To apply a filter for a cell value, right-click the cell and choose filter -> **Filter By Selected Cell's Value** option from the menu. It applies the filter based on the value of the selected cell in the current sheet.

Clear filter

After applying filter to a certain column, you may want to clear it to make all filtered rows visible again. It can be done in the following ways,

- Choose **Clear** option in ribbon toolbar under **Filter and Sort**. It clears the filters applied in the spreadsheet for all fields.

- Use the `clearFilter()` method programmatically, to clear the applied filters in spreadsheet for all fields.

Clear filter on a field

After filtering, you can clear/reset the filter for a field alone. It can be done in the following ways,

- Click filter icon in the column's header and then choose **Clear Filter** option from the filter dialog.
- You can right-click on a filtered column cell and choose **Clear Filter from** . option from the context menu.
- Use the `clearFilter(field)` method programmatically, to clear the filter in a particular column.

Reapply filter

When you want to reapply the filter after some changes happened in the rows. It can be done in the following ways,

- You can choose **Reapply** option in ribbon toolbar under **Filter and Sort** to reapply the filtered columns again.
- You can right-click on a filtered cell and choose **Reapply** option from the context menu. It reapplies the filters again in the Spreadsheet for all the fields.

Known error validations

The following errors have been handled for filtering,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message **Select a cell or range inside the used range and try again** will be displayed. No filter will be performed if the range is invalid.

Limitations

The following features have some limitations in Filter:

- Insert/delete row/column between the filter applied cells.
- Merge cells with filter.
- Copy/cut paste the filter applied cells.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sorting](#)
- [Hyperlink](#)
- [Undo Redo](#)

Sort in Angular Spreadsheet component

Sorting helps arranging the data to a specific order in a selected range of cells. You can use the [allowSorting](#) property to enable or disable sorting functionality.

* The default value for `allowSorting` property is `true`.

By default, the `sort` module is injected internally into Spreadsheet to perform sorting.

Sort by cell value

In the active Spreadsheet, select a range of cells to sort by cell value. The range sort can be done by any of the following ways:

- Select the sort item in the Ribbon toolbar and choose the ascending or descending item.
- Right-click the sheet, select the sort item in the context menu and choose the ascending/descending item.
- Use the [sort\(\)](#) method programmatically.

The cell values can be sorted in the following orders:

- Ascending
- Descending

* Ascending is the default order for sorting.

The `sort()` method with empty arguments will sort the selected range by active cell's column as sort column in ascending order.

* The [beforeSort](#) event will be triggered before sorting the specified range.

* The [sortComplete](#) event will be triggered after the sort action is completed successfully.

The following code example shows `sort` functionality in the Spreadsheet control.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { defaultData } from './datasource';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (dataBound)='dataBound()' '
(sortComplete)='sortComplete($event) ' > <e-sheets> <e-sheet> <e-ranges> <e-
range [dataSource]='defaultData'></e-range></e-ranges><e-columns><e-column
[width]=130></e-column><e-column [width]=92></e-column><e-column
[width]=96></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
```

```

public defaultData?: object[];
@ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
ngOnInit(): void {
    this.defaultData = defaultData;
}
dataBound(){
    if (this.spreadsheetObj!.activeSheetIndex === 0) {
        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold' },
        'A1:H1');
        this.spreadsheetObj!.sort({ containsHeader: true }, 'A1:H11');
    }
};
sortComplete (args : any) {
    this.spreadsheetObj!.selectRange(args.range);
    // code here.
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Data contains header

You can specify whether the selected range of cells contains header. To specify, you need to set the [containsHeader](#) property to `true` and pass it as `sortOption` arguments of the `sort()` method.

* If the `containsHeader` property is not set and active cell column's first cell value type is differed from the second cell value type, the first row data in the range are marked as column headers.

You can also enable or disable this property using `beforeSort` event arguments,

`typescript

```

beforeSort (args) {
    args.sortOptions.containsHeader = true;
}

```

In the custom sort dialog, the `Data contains header` checkbox is checked on load. Thus, the default value for `containsHeader` is `true` in custom sort dialog.

Case sensitive sort

The default sort functionality of Spreadsheet is a case insensitive sorting. When you want to perform sorting with case sensitive, you need to set the [caseSensitive](#) property to `true` and pass it as `sortOption` arguments of the `sort()` method.

Case sensitive sorting is applicable only for cells with alphabets. In ascending order sorting with case sensitive enabled, the cells with lower case text will be placed above the cells with upper case text.

* The default value for the `caseSensitive` property is `false`.

You can also enable or disable this property using `beforeSort` event arguments,

```
`typescript
beforeSort (args) {
  args.sortOptions.caseSensitive = true;
}
`
```

In the custom sort dialog, the `Case sensitive` checkbox is unchecked on load as the default value is `false`.

Sort multiple columns

When you want to perform sorting on multiple columns, it can be done by any of the following ways:

- Select the `Custom sort...` menu item from the Ribbon toolbar item or context menu item.
- Use the `sort()` method programmatically by providing sort criteria.

* The current sorting functionality supports sorting based on cell values only.

Custom sort dialog

The custom sort dialog helps sorting multiple columns in the selected range by utilizing the rich UI. This dialog will be appeared while choosing the `Custom sort...` from the Ribbon item or context menu item. By default, sort criteria with the first column name from the selected range will be appeared in the dialog on initial load and it cannot be removed.

You can add multiple criteria using the `Add Column` button at the bottom of the dialog. Thus, multiple columns can be specified with different sort order. The newly added sort criteria items can be removed using the `delete` icons at the end of each items.

You can refer to the [Data contains header](#) topic to learn more about `Data contains header` checkbox. To learn more about `Case sensitive` checkbox, you can refer to [Case sensitive sort](#) topic.

Passing sort criteria manually

The multi-column sorting can also be performed manually by passing sort options to the `sort()` method programmatically. The `sortOption` have the following arguments:

- [sortDescriptors](#) – Sort criteria collection that holds the collection of field name, sort order, and [sortComparer](#).
- `containsHeader` – Boolean argument that specifies whether the range has headers in it.
- `caseSensitive` – Boolean argument that specifies whether the range needs to consider case.

* All the arguments are optional.

* When a `sortDescriptor` is specified without field, the field of the first `sortDescriptor` from the collection will be assigned from active cell's column name and others will be ignored. Hence, it will act as single column sorting.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
```



```

import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { tradeData } from './datasource';
import { SpreadsheetComponent, SortDescriptor } from '@syncfusion/ej2-
angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (dataBound)='dataBound()' '
(sortComplete)='sortComplete($event)'> <e-sheets> <e-sheet> <e-ranges> <e-
range [dataSource]='tradeData'></e-range></e-ranges><e-columns><e-column
[width]=100></e-column><e-column [width]=120></e-column><e-column
[width]=96></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public tradeData?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.tradeData = tradeData;
  }
  dataBound() {
    let sortDescriptors: SortDescriptor[] = [
      {
        field: 'F',
        order: 'Ascending'
      },
      {
        field: 'E',
        order: 'Ascending'
      },
      {
        field: 'C',
        order: 'Descending'
      }
    ];
    if (this.spreadsheetObj!.activeSheetIndex === 0) {
      this.spreadsheetObj!.sort({ sortDescriptors:
sortDescriptors, containsHeader: true }, 'A1:H30');
    }
  };
  sortComplete (args : any) {
    this.spreadsheetObj!.selectRange(args.range);
    // code here.
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom sort comparer

The [sortDescriptor](#) holds the [sortComparer](#) property, which is a function and it is used to customize the sort comparer for specific sort criteria. Each [sortDescriptor](#) can be customized using the custom sort comparer function.

By customizing sort comparer, you can define the sort action as desired.

* The [sortComparer](#) is an optional property of [sortDescriptor](#).

For custom sort comparer example, refer to the [Sort a range by custom list](#) in the [how-to](#) section.

Known error validations

The following errors have been handled for sorting,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message [Select a cell or range inside the used range and try again](#) will be displayed. No sort will be performed if the range is invalid.
- *Empty field validation:* When the sort criteria does not have a column selected (empty) in the custom sort dialog, it will become invalid, and an error message [Sort criteria column should not be empty](#) will be displayed on [OK](#) button click.
- *Duplicate field validation:* When the column names of added sort criteria are repeated more than once in the custom sort dialog, it will become invalid and an error message [is mentioned more than once. Duplicate columns must be removed](#) will be displayed on [OK](#) button click.

Limitations

- Sorting is not supported with formula contained cells.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sort a range by custom list](#)
- [Hyperlink](#)
- [Filtering](#)
- [Undo Redo](#)

Link in Angular Spreadsheet component

Hyperlink is used to navigate to web links or cell reference within the sheet or to other sheets in Spreadsheet. You can use the [allowHyperlink](#) property to enable or disable hyperlink functionality.

* The default value for [allowHyperlink](#) property is [true](#).

Insert Link

You can insert a hyperlink in a worksheet cell for quick access to related information.

User Interface:

In the active spreadsheet, click the cell where you want to create a hyperlink. Insert hyperlink can be done by any of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the **Link** item.
- Right-click the cell and then click Hyperlink item in the context menu, or you can press Ctrl+K.
- Use the [addHyperlink\(\)](#) method programmatically.

Edit Hyperlink

You can change an existing hyperlink in your workbook by changing its destination or the text that is used to represent it.

User Interface:

In the active spreadsheet, Select the cell that contains the hyperlink that you want to change. Editing the hyperlink while opening the dialog can be done by any one of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the **Link** item.
- Right-click the cell and then click Edit Hyperlink item in the context menu, or you can press Ctrl+K.

In the Edit Link dialog box, make the changes that you want, and click UPDATE.

Remove Hyperlink

Performing this operation remove a single hyperlink without losing the display text.

User Interface:

In the active spreadsheet, click the cell where you want to remove a hyperlink. remove hyperlink can be done by any of the following ways:

- Right-click the cell and then click Remove Hyperlink item in the context menu.
- Use the [removeHyperlink\(\)](#) method programmatically.

How to change target attribute

There is an event named **beforeHyperlinkClick** which triggers only on clicking hyperlink. You can customize where to open the hyperlink by using the **target** property in the arguments of that event.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
```

```

    ],
    standalone: true,
    selector: 'app-container',
    template: "<ejs-spreadsheet #spreadsheet
    (beforeHyperlinkClick)='beforeHyperlinkClick($event)'> <e-sheets> <e-sheet
    name='Monthly Budget' selectedRange='D13'> <e-rows> <e-row> <e-cells>
    <e-cell value='Item Name'></e-cell> <e-cell value='Quantity'></e-cell>
    <e-cell value='Price'></e-cell> <e-cell value='Amount'></e-cell> <e-cell
    value='Stock Detail'></e-cell> <e-cell value='Website'></e-cell> </e-
    cells> </e-row> <e-row> <e-cells> <e-cell value='Casual
    Shoes'></e-cell> <e-cell value='10'></e-cell> <e-cell value='20'></e-
    cell> <e-cell value='200'></e-cell> <e-cell value='OUT OF STOCK'></e-cell>
    <e-cell value='Amazon' hyperlink='https://www.amazon.com/'></e-cell> </e-
    cells> </e-row> <e-row> <e-cells> <e-cell value='Sports Shoes'></e-cell>
    <e-cell value='20'></e-cell> <e-cell value='30'></e-cell> <e-cell
    value='600'></e-cell> <e-cell value='IN STOCK' hyperlink='Stock!A2:B2'></e-
    cell> <e-cell value='Overstack' hyperlink='https://www.overstock.com/'></e-
    cell> </e-cells> </e-row> <e-row> <e-cells> <e-cell value='Formal
    Shoes'></e-cell> <e-cell value='20'></e-cell> <e-cell value='15'></e-cell>
    <e-cell value='300'></e-cell> <e-cell value='IN STOCK'
    hyperlink='Stock!A3:B3'></e-cell> <e-cell value='Aliexpress'
    hyperlink='https://www.aliexpress.com/'></e-cell> </e-cells> </e-row> <e-
    row> <e-cells> <e-cell value='Sandals & Floaters'></e-cell> <e-cell
    value='15'></e-cell> <e-cell value='20'></e-cell> <e-cell
    value='300'></e-cell> <e-cell value='OUT OF STOCK'></e-cell> <e-cell
    value='Alibaba' hyperlink='http://www.alibaba.com/'></e-cell> </e-cells>
    </e-row> <e-row> <e-cells> <e-cell value='Flip-Flops & Slippers'></e-
    cell> <e-cell value='30'></e-cell> <e-cell value='10'></e-cell> <e-cell
    value='300'></e-cell> <e-cell value='IN STOCK'
    hyperlink='Stock!A4:B4'></e-cell> <e-cell value='Taobao'
    hyperlink='https://taobao.com/'></e-cell> </e-cells> </e-row> </e-rows>
    <e-columns> <e-column [width]=110></e-column> <e-column
    [width]=115></e-column> <e-column [width]=110></e-column> <e-column
    [width]=100></e-column> <e-column [width]=120></e-column> <e-column
    [width]=140></e-column> </e-columns> </e-sheet> <e-sheet name='Stock'
    selectedRange='D13'> <e-rows> <e-row> <e-cells> <e-cell value='Item
    Name'></e-cell> <e-cell value='Available Count'></e-cell> </e-cells>
    </e-row> <e-row> <e-cells> <e-cell value='Casual Shoes'></e-cell> <e-cell
    value='10'></e-cell> </e-cells> </e-row> <e-row> <e-cells> <e-cell
    value='Sports Shoes'></e-cell> <e-cell value='20'></e-cell> </e-cells>
    </e-row> <e-row> <e-cells> <e-cell value='Formal Shoes'></e-cell> <e-cell
    value='20'></e-cell></e-cells></e-row> <e-row> <e-cells> <e-cell
    value='Sandals & Floaters'></e-cell><e-cell value='15'></e-cell> </e-cells>
    </e-row><e-row> <e-cells> <e-cell value='Flip-Flops & Slippers'></e-cell>
    <e-cell value='30'></e-cell> </e-cells> </e-row> </e-rows> <e-columns> <e-
    column [width]=110></e-column><e-column [width]=115></e-column></e-
    columns></e-sheet></e-sheets></ejs-spreadsheet>"
  })
  export class AppComponent implements OnInit {
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    ngOnInit(): void {
    }
    beforeHyperlinkClick (args : any) {
      args.target = '_self'; //change target attribute
    }
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limitations

- Inserting hyperlink not supported for multiple ranges.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

Clipboard in Angular Spreadsheet component

The Spreadsheet provides support for the clipboard operations (cut, copy, and paste). Clipboard operations can be enabled or disabled by setting the [enableClipboard](#) property in Spreadsheet.

By default, the `enableClipboard` property is true.

Cut

It is used to cut the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Cut can be done in one of the following ways.

- Using Cut button in the Ribbon's HOME tab to perform cut operation.
- Using Cut option in the Context Menu.
- Using `Ctrl + X` | `Command + X` keyboard shortcut.
- Using the [cut](#) method.

Copy

It is used to copy the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

User Interface:

Copy can be done in one of the following ways.

- Using Copy button in the Ribbon's HOME tab to perform copy operation.

- Using Copy option in the Context Menu.
- Using **Ctrl + C** | **Command + C** keyboard shortcut.
- Using the [copy](#) method.

Paste

It is used to paste the clipboard data to the selected range, rows or columns. You have the following options in Paste,

- **Paste Special** - You can paste the values with formatting.
- **Paste** - You can paste only the values without formatting.

It also performs for external clipboard operation. If you perform cut and paste, clipboard data will be cleared, whereas in copy and paste the clipboard contents will be maintained. If you perform paste inside the copied range, the clipboard data will be cleared.

User Interface:

Paste can be done in one of the following ways.

- Using Paste button in the Ribbon's HOME tab to perform paste operation.
- Using Paste option in the Context Menu.
- Using **Ctrl + V** | **Command + V** keyboard shortcut.
- Using the [paste](#) method.

If you use the Keyboard shortcut key for cut (**Ctrl + X**) | copy (**Ctrl + C**) from other sources, you should use **Ctrl + V** shortcut while pasting into the spreadsheet.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from './datasource';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
enableRipple(true);
@Component({
  imports: [
    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<button ej2-dropdownbutton [items]='items'
content='Clipboard' (select)='itemSelect($event)'></button>
<ej2-spreadsheet #spreadsheet (created)="created()"
[enableClipboard]="true">
<e-sheets>
```

```

        <e-sheet name="Price Details">
          <e-ranges>
            <e-range [dataSource]="priceData"></e-range>
          </e-ranges>
          <e-columns>
            <e-column [width]=130></e-column>
            <e-column [width]=92></e-column>
            <e-column [width]=96></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    priceData: object[] = dataSource1;
    public items: ItemModel[] = [
      {
        text: "Copy"
      },
      {
        text: "Cut"
      },
      {
        text: "Paste"
      }
    ];
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
    }
    public itemSelect(args: MenuEventArgs) {
      if (args.item.text === 'Copy')
        this.spreadsheetObj!.copy();
      if (args.item.text === 'Cut')
        this.spreadsheetObj!.cut();
      if (args.item.text === 'Paste')
        this.spreadsheetObj!.paste();
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Prevent the paste functionality

The following example shows, how to prevent the paste action in spreadsheet. In [actionBegin](#) event, you can set `cancel` argument as false in paste request type.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from './datasource';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-splitbuttons';
enableRipple(true);
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<button ej2-dropdownbutton [items]='items'
content='Clipboard' (select)='itemSelect($event)'></button>
<ejs-spreadsheet #spreadsheet (created)="created()"
(actionBegin)="actionBeginHandler($event)" [enableClipboard]="true">
  <e-sheets>
    <e-sheet name="Price Details">
      <e-ranges>
        <e-range [dataSource]="priceData"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=130></e-column>
        <e-column [width]=92></e-column>
        <e-column [width]=96></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  priceData: object[] = dataSource1;
  public items: ItemModel[] = [
    {
      text: "Copy"
    },
    {
      text: "Cut"
    },
    {
      text: "Paste"
    }
  ];
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
  }
  // Triggers before the action begins.
  actionBeginHandler(pasteArgs: { args: { eventArgs: { requestType:
string; cancel: boolean; }; }; }) {

```



```
// To cancel the paste action.
    if (pasteArgs.args.eventArgs.requestType === 'paste') {
        pasteArgs.args.eventArgs.cancel = true;
    }
}

public itemSelect(args: MenuEventArgs) {
    if (args.item.text === 'Copy')
        this.spreadsheetObj!.copy();
    if (args.item.text === 'Cut')
        this.spreadsheetObj!.cut();
    if (args.item.text === 'Paste')
        this.spreadsheetObj!.paste();
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Limitations

- External clipboard is not fully supported while copying data from another source and pasting into a spreadsheet, it only works with basic supports (Values, Number, cell, and Text formatting).
- If you copy =SUM(A2,B2) and paste, the formula reference will change depending on the pasted cell address but we don't have support for nested formula(formula reference will be same).
- Clipboard is not supported with conditional formatting (values only pasting).
- We have limitation while copying the whole sheet data and pasting it into another sheet.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

Selection in Angular Spreadsheet component

Selection provides interactive support to highlight the cell, row, or column that you select. Selection can be done through Mouse, Touch, or Keyboard interaction. To enable selection, set `mode` as `Single` | `Multiple` in [selectionSettings](#). If you set `mode` to `None`, it disables the UI selection.

* The default value for `mode` in `selectionSettings` is `Multiple`.

You have the following options in Selection,

- Cell selection
- Row selection
- Column selection

Cell selection

Cell selection is used to select a single or multiple cells. It can be performed using the [selectRange](#) method.

User Interface:

- Click on a cell to select it (or) use the **arrow** keys to navigate to it and select it.
- To select a range, select a cell, then use the left mouse button to select and drag over to other cells (or) use the **Shift + arrow** keys to select the range.
- To select non-adjacent cells and cell ranges, hold **Ctrl** and select the cells.

You can quickly locate and select specific cells or ranges by entering their names or cell references in the Name box, which is located to the left of the formula bar, and also select named or unnamed cells or ranges by using the Go To (**Ctrl+G**) command.

Row selection

Row selection is used to select a single or multiple rows.

User Interface:

You can perform row selection in any of the following ways,

- By clicking the row header.
- To select multiple rows, select a row header with the left mouse button and drag over to other row headers (or) use the **Shift + arrow** keys to select multiple rows.
- To select non-adjacent rows, hold **Ctrl** and select the row header.
- You can also use the **selectRange** method for row selection.

The following sample shows the row selection in the spreadsheet, here selecting the 5th row using the **selectRange** method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getRangeAddress } from '@syncfusion/ej2-
angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
(cellEdit)="cellEdit($event)" [selectionSettings]="{ mode: 'Multiple' }">
    <e-sheets>
      <e-sheet name="Price Details">
```

```

        <e-ranges>
          <e-range [dataSource]="priceData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=130></e-column>
          <e-column [width]=92></e-column>
          <e-column [width]=96></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    priceData: object[] = dataSource1;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
      var colCount = this.spreadsheetObj!.getActiveSheet().colCount;
      this.spreadsheetObj!.selectRange(getRangeAddress([4, 0, 4, colCount
as any]));
    }
    cellEdit(args : any) {
      args.cancel = true;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Column selection

Column selection is used to select a single or multiple columns.

User Interface:

You can perform column selection in any of the following ways,

- By clicking the column header.
- To select multiple columns, select a column header with the left mouse button and drag over to other column headers (or) use the **Shift + arrow** keys to select the multiple columns.
- To select non-adjacent columns, hold **Ctrl** and select the column header.
- You can also use the **selectRange** method for row selection.

The following sample shows the column selection in the spreadsheet, here selecting the 3rd column using the **selectRange** method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, getRangeAddress } from '@syncfusion/ej2-
angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[selectionSettings]="{ mode: 'Multiple' }">
    <e-sheets>
      <e-sheet name="Price Details">
        <e-ranges>
          <e-range [dataSource]="priceData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=130></e-column>
          <e-column [width]=92></e-column>
          <e-column [width]=96></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  priceData: object[] = dataSource1;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
    var rowCount = this.spreadsheetObj!.getActiveSheet().rowCount;
    this.spreadsheetObj!.selectRange(getRangeAddress([0, 2, rowCount as
any, 2]));
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

How to remove selection in the spreadsheet

The following sample shows, how to remove the selection in the spreadsheet. Here changing the **mode** as **None** in [selectionSettings](#) to disable's the UI selection.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from '../datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[selectionSettings]="{ mode: 'None' }">
    <e-sheets>
      <e-sheet name="Price Details">
        <e-ranges>
          <e-range [dataSource]="priceData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=130></e-column>
          <e-column [width]=92></e-column>
          <e-column [width]=96></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  priceData: object[] = dataSource1;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations

- We have a limitation while performing the Select All (ctrl + A). You can do this only by clicking the Select All button at the top left corner.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

Scrolling in Angular Spreadsheet component

Scrolling helps you to move quickly to different areas of the worksheet. It moves faster if we use horizontal and vertical scroll bars. Scrolling can be enabled by setting the [allowScrolling](#) as true.

By default, the `allowScrolling` property is true.

You have the following options in Scrolling by using [scrollSettings](#).

- Finite scrolling.
- Virtual scrolling.

Finite Scrolling

Finite scrolling supports two type of modes in scrolling. You can use the [isFinite](#) property in [scrollSettings](#) to specify the mode of scrolling.

- Finite - This mode does not create a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as `true`.
- Infinite - This mode creates a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as `false`.

By Default, the `isFinite` property is `false`.

Virtual Scrolling

- Virtual scrolling allows you to load data that you require (load data based on viewport size) without buffering the entire huge database. You can set the `enableVirtualization` property in `scrollSettings` as `true`.

In virtual scrolling `enableVirtualization` is set to true means, it allows you to load the spreadsheet data while scrolling.

By Default, the `enableVirtualization` property is `true`.

User Interface:

You can scroll through the worksheet using one of the following ways,

- Using the `arrow` keys.
- Using the Horizontal and Vertical `scroll` bars.
- Using the `mouse` wheel.

Finite scrolling with defined rows and columns

If you want to perform scrolling with defined rows and columns, you must define `rowCount` and `colCount` in the `sheets` property and set `isFinite` as true and `enableVirtualization` as false in `scrollSettings`.

The following code example shows the finite scrolling with defined rows and columns in the spreadsheet. Here, we used `rowCount` as 20 and `colCount` as 20, after reaching the 20th row or 20th column you can't able to scroll.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
[allowScrolling]="true" [scrollSettings]="{ isFinite: true }">
    <e-sheets>
      <e-sheet name="Price Details" rowCount="9" colCount="7">
        <e-ranges>
          <e-range [dataSource]="priceData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=130></e-column>
          <e-column [width]=92></e-column>
          <e-column [width]=96></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=92></e-column>
          <e-column [width]=96></e-column>
          <e-column [width]=96></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  priceData: object[] = dataSource1;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

Protect sheet in Angular Spreadsheet component

Sheet protection helps you to prevent the users from modifying the data in the spreadsheet.

Protect Sheet

Protect sheet feature helps you to prevent the unknown users from accidentally changing, editing, moving, or deleting data in a spreadsheet. And you can also protect the sheet with password.

You can use the [isProtected](#) property to enable or disable the Protecting functionality.

* The default value for `isProtected` property is `false`.

By default in protected sheet, selecting, formatting, inserting, deleting functionalities are disabled. To enable some of the above said functionalities the `protectSettings` options are used in a protected spreadsheet.

The available `protectSettings` options in spreadsheet are,

| Options | Uses |

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

| `Select Cells` | Used to perform Cell Selection. |

| `Format Cells` | Used to perform Cell formatting. |

| `Format Rows` | Used to perform Row formatting. |

| `Format Columns` | Used to perform Column formatting. |

| `Insert Link` | Used to perform Hyperlink Insertions. |

* The default value for all `protectSettings` options are `false`.

By default, the `Protect Sheet` module is injected internally into the Spreadsheet to perform sheet protection function.

User Interface:

In the active Spreadsheet, the sheet protection can be done by any of the following ways:

- Select the Protect Sheet item in the Ribbon toolbar under the Data Tab, and then select your desired options.
- Right-click the sheet tab, select the Protect Sheet item in the context menu, and then select your desired options.

- Use the [protectSheet\(\)](#) method programmatically.

The following example shows **Protect Sheet** functionality with password in the Spreadsheet control.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1, dataSource2 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()">
    <e-sheets>
      <e-sheet name="Budget" [isProtected]="true"
[protectSettings]="{ selectCells: true }">
        <e-ranges>
          <e-range [dataSource]="budgetData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
        </e-columns>
      </e-sheet>
      <e-sheet name="Salary">
        <e-ranges>
          <e-range [dataSource]="salaryData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  budgetData: object[] = dataSource1;
  salaryData: object[] = dataSource2;
  created() {
    this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
  }
}
```

```

        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Salary!A1:D1');
        this.spreadsheetObj!.protectSheet(1, { selectCells: false},
"syncfusion"); // protect sheet with password
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations of Protect sheet

- Password encryption is not supported

Unprotect Sheet

Unprotect sheet is used to enable all the functionalities that are already disabled in a protected spreadsheet.

User Interface:

In the active Spreadsheet, the sheet Unprotection can be done by any of the following ways:

- Select the **Unprotect Sheet** item in the Ribbon toolbar under the Data Tab.
- Right-click the sheet tab, select the **Unprotect Sheet** item in the context menu.
- Use the [unprotectSheet\(\)](#) method programmatically.

Unlock the particular cells in the protected sheet

In protected spreadsheet, to make some particular cell or range of cells are editable, you can use [lockCells\(\)](#) method, with the parameter **range** and **isLocked** property as false.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { Dialog } from '@syncfusion/ej2-popups';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1, dataSource2 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<button class="e-btn" style="margin: 5px 0;"
(click)="btnClick()">
    Unlock cells</button>
    <div id="dialog"></div>
    <ejs-spreadsheet #spreadsheet id="spreadsheet"
(created)="created()">
    <e-sheets>
    <e-sheet name="Budget" [isProtected]="true"
[protectSettings]="{ selectCells: true }">
    <e-ranges>
    <e-range [dataSource]="budgetData"></e-range>
    </e-ranges>
    <e-columns>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    </e-columns>
    </e-sheet>
    <e-sheet name="Salary">
    <e-ranges>
    <e-range [dataSource]="salaryData"></e-range>
    </e-ranges>
    <e-columns>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    <e-column [width]=100></e-column>
    </e-columns>
    </e-sheet>
    </e-sheets>
    </ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    dialogObj: Dialog | undefined;
    budgetData: object[] = dataSource1;
    salaryData: object[] = dataSource2;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Salary!A1:D1');
      // Creating dialog component,
      this.dialogObj = new Dialog({
        header: 'Spreadsheet',
        target: document.getElementById('spreadsheet') as any,
        content: '"A1:F3" range of cells has been unlocked.',
        showCloseIcon: true,

        visible: false,
        width: '500px',
        buttons: [{
          click: this.lockCells.bind(this), buttonModel: { content:
'Ok', isPrimary: true }

```

```

    }],
    });
    this.dialogObj.appendTo('#dialog');
  }
  btnClick(): void {
    this.dialogObj!.show();
  }
  lockCells(): void {
    this.spreadsheetObj!.lockCells('A1:F3', false);
    this.dialogObj!.hide();
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Protect Workbook

Protect workbook feature helps you to protect the workbook so that users cannot insert, delete, rename, hide the sheets in the spreadsheet.

You can use the [password](#) property to protect workbook with password.

You can use the [isProtected](#) property to protect or unprotect the workbook without the password.

The default value for `isProtected` property is `false`.

User Interface:

In the active Spreadsheet, you can protect the worksheet by selecting the Data tab in the Ribbon toolbar and choosing the **Protect Workbook** item. Then, enter the password and confirm it and click on OK.

The following example shows **Protect Workbook** by using the [isProtected](#) property in the Spreadsheet control.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet [isProtected]='true'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-
"

```

```
column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-
column [width]=130></e-column><e-column [width]=120></e-column></e-
columns></e-sheet></e-sheets></ejs-spreadsheet>"
}))
export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    ngOnInit(): void {
        this.data = data;
    }
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

The following example shows **Protect Workbook** by using the [password](#) property in the Spreadsheet control. To unprotect the workbook, click the unprotect workbook button in the data tab and provide the password as syncfusion in the dialog box.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
    imports: [

        SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template: "<ejs-spreadsheet #spreadsheet password='syncfusion'> <e-
sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-
ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-
column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-
column [width]=130></e-column><e-column [width]=120></e-column></e-
columns></e-sheet></e-sheets></ejs-spreadsheet>"
}))
export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    ngOnInit(): void {
        this.data = data;
    }
};
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Unprotect Workbook

Unprotect Workbook is used to enable the insert, delete, rename, move, copy, hide or unhide sheets feature in the spreadsheet.

User Interface:

In the active Spreadsheet, the workbook Unprotection can be done in any of the following ways:

- Select the **Unprotect Workbook** item in the Ribbon toolbar under the Data Tab and provide the valid password in the dialog box.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Hyperlink](#)

Searching in Angular Spreadsheet component

Find and Replace helps you to search for the target text and replace the found text with alternative text within the sheet or workbook. You can use the [allowFindAndReplace](#) property to enable or disable the Find and Replace functionality.

* The default value for **allowFindAndReplace** property is **true**.

Find

Find feature is used to select the matched contents of a cell within the sheet or workbook. It is extremely useful when working with large set of data source.

User Interface:

Find can be done by any of the following ways:

- Select the Search icon in the Ribbon toolbar or use **Ctrl + F** key to open the Find dialog.
- Use find Next and find Previous buttons to search the given value in the workbook.
- Select the option button in Find dialog to open the Find and Replace dialog. Then, select the below properties for enhanced searching.

* **Search within**: To search the target in a sheet (default) or in an entire workbook.

* **Search by**: It enhance the search, either By Rows (default), or By Columns.

* **Match case**: To find the matched value with case sensitive.

* **Match exact cell contents**: To find the exact matched cell value with entire match cases.

- Using [find\(\)](#) method to perform find operation.

Replace

Replace feature is used to change the find contents of a cell within sheet or workbook. Replace All is used to change all the matched contents of a cell within sheet or workbook.

User Interface:

Replace can be done by any of the following ways:

- Use **Ctrl + H** key to open the Find and Replace dialog.
- Use Replace button to change the found value in sheet or workbook.
- Using Replace All button, all the matched criteria can be replaced with find value based on sheet or workbook.
- Using [replace\(\)](#) method to perform replace operation by passing the argument `args.replaceby` as `replace`.
- Using [replace\(\)](#) method to perform replace all operation by passing the argument `args.replaceby` as `replaceall`.

Go to

Go to feature is used to navigate to a specific cell address in the sheet or workbook.

User Interface:

- Use **Ctrl + G** key to open the Go To dialog.
- Use [goTo\(\)](#) method to perform Go To operation.

In the following sample, searching can be done by following ways:

- Select the Home tab in the Ribbon toolbar, and then select the Search icon.
- Enter any value in the search textbox.
- Select the next (or) previous button to find the entered value in the spreadsheet.
- You can have more options to find values by selecting the more options in the search toolbar.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { enableRipple } from '@syncfusion/ej2-base';
import { dataSource1, dataSource2 } from './datasource';
enableRipple(true);
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
```

```

    template: `<ejs-spreadsheet #spreadsheet (created)="created()" "[
[showFormulaBar]="false">
    <e-sheets>
      <e-sheet name="Price Details">
        <e-ranges>
          <e-range [dataSource]="priceData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=150></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=110></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
          <e-column [width]=85></e-column>
        </e-columns>
      </e-sheet>
      <e-sheet name="Budget Details">
        <e-ranges>
          <e-range [dataSource]="budgetData"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
          <e-column [width]=100></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent | undefined;
    priceData: object[] = dataSource1;
    budgetData: object[] = dataSource2;
    created() {
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:H1');
      this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Budget Details!A1:D1');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limitations

- Undo/redo for Replace All is not supported in this feature.

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

Keyboard shortcuts in Angular Spreadsheet component

The keyboard shortcuts supported in the spreadsheet are,

| Shortcut | Description |
|-----------------------|---|
| ----- ----- | |
| Ctrl + O | Displays dialog to open a file. |
| Ctrl + S / Alt + F2 | Saves the workbook. |
| F2 | Enables edit mode. |
| ESC | Cancel edit mode and discard the changes. |
| Backspace and SPACE | Clears content of the active cell and enables edit mode. |
| Ctrl + C | Copies the selected cells. |
| Ctrl + X | Cuts the selected cells. |
| Ctrl + V | Paste the clipboard(cut or copied) content in the new selected range. |
| Ctrl + B | Applies or removes bold formatting. |
| Ctrl + I | Applies or removes <i>italic</i> formatting. |
| Ctrl + U | Applies or removes <u>underline</u> . |
| Ctrl + 5 | Applies or removes strikethrough . |
| Ctrl + Z | Reverses (Undo) the last action. |
| Ctrl + Y | Repeats (Redo) the last reversed action. |
| Ctrl + K | It opens the Insert Hyperlink dialog for adding new hyperlink to a cell. If the selected cell already contains hyperlink, it opens the Edit Hyperlink dialog. |
| Ctrl + F / Shift + F5 | Opens Find dialog. |
| Ctrl + H | Opens Find and Replace dialog. |
| Ctrl + G | Opens GoTo dialog, which helps to navigate to cell. |
| Ctrl + Shift + L | Applies filter to the first row of the selected range or used range. |
| Alt + F | Opens the File menu. |
| Alt + H | Go to Home tab. |
| Alt + N | Go to Insert tab. |
| Alt + M | Go to Formulas tab. |
| Alt + A | Go to Data tab. |
| Alt + W | Go to View tab. |

- | Tab | Navigate the active cell to the next cell in the same row. |
- | Shift + Tab | Navigate the active cell to the previous cell in the same row. |
- | Right or Left arrow | Move the focus to next or previous item in the tab if the focus is on ribbon tab. |
- | Up arrow | When a menu is open, move focus to the next item. |
- | Down arrow | When a menu is open, move focus to the previous item. |
- | Spacebar or Enter | Activate a selected button. |
- | Ctrl + F8 | Expand or collapse the ribbon content. |
- | Ctrl + Shift + U | Expand or collapse the formula bar. |
- | Ctrl + 9 | Hide the selected row. |
- | Ctrl + 0 | Hide the selected column. |
- | Home | Moves the selection to starting column in worksheet. |
- | Ctrl + Home | Move the selection to the first visible cell on a worksheet. |
- | Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |
- | Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |
- | Ctrl + & | Apply an outline border to the selected cells. |
- | Ctrl + Shift + & | Apply an outline border to the cells that you've chosen. |
- | Ctrl + Shift + ~ | Apply the **General** number format. |
- | Ctrl + Shift + \$ | Apply the **Currency** format with two decimal places (negative numbers in parentheses). |
- | Shift + F10 | Open the context menu. |
- | Ctrl + % | Apply the **Percentage** format with no decimal places. |
- | Ctrl + ^ | Apply the **Scientific** number format with two decimal places. |
- | Ctrl + Shift + # | Apply the **Date** format with the day, month, and year. |
- | Ctrl + Shift + @ | Apply the **Time** format with the hour and minute, and AM or PM. |
- | Ctrl + Shift + ! | Apply the **Number** format with two decimal places, thousands separator, and minus sign (-) for negative values. |
- | Ctrl + Spacebar | Select an entire column in a worksheet. |
- | Shift + Spacebar | Select an entire row in a worksheet. |
- | Shift + F3 | Opens **Insert Function** dialog. |
- | Ctrl + Alt + N | Opens new workbook. |
- | Shift + Page Down | Perform page down by selecting all cells between. |
- | Shift + Page Up | Perform page up by selecting all cells between. |
- | Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |

- | Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |
- | Ctrl + Up | Navigate to the first non-blank cell in the same row as the active cell. |
- | Ctrl + Down | Navigate to the last cell that is not blank in the same column as the active cell. |
- | Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Up | Extend the selection of cells to the first non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Down | Extend the selection of cells to the last non-blank cell in the same row as the active cell. |
- | Shift + Alt + K | Open the **List All Sheets** dropdown option. |
- | Up arrow | Navigate from the active cell to the previous cell in the same column. |
- | Down arrow | Navigate from the active cell to the next cell in the same column. |
- | Left arrow | Navigate from the active cell to the previous cell in the same row. |
- | Right arrow | Navigate from the active cell to the next cell in the same row. |
- | Page Up | Move page up. |
- | Page Down | Move page down. |
- | Shift + Up | Extend the selection of cells to the previous row. |
- | Shift + Down | Extend the selection of cells to the next row. |
- | Shift + Left | Extend the selection of cells to the previous column. |
- | Shift + Right | Extend the selection of cells to the next column. |
- | Ctrl + Top | Navigate to the non-blank cell before the active cell in the same column. |
- | Ctrl + Shift + Top | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |
- | Ctrl + Shift + Bottom | Extend the cell selection to the last non-blank cell in the same column as the active cell. |
- | Enter | Navigate the active cell to the next cell in the same column. |
- | Shift + Enter | Navigate to the previous cell in the same column from the active cell. |
- | Alt + Down | Open the list data validation menu and filter menu. |
- | Alt + Up | Close the list data validation menu and filter menu. |
- | Delete | Remove the contents of the cell. |
- | Shift + Home | Extend the cell selection to the first column of a worksheet. |
- | Shift + F11 | Add a new sheet. |
- | Ctrl + Shift + 9 | Unhide the selected rows. |

| Ctrl + Shift + 0 | Unhide the selected columns. |

| Ctrl + D | Fill the cell down. |

| Ctrl + R | Fill the cell right. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Formatting](#)

Ribbon in Angular Spreadsheet component

It helps to organize a spreadsheet's features into a series of tabs. By clicking the expand or collapse icon, you can expand or collapse the ribbon toolbar dynamically.

Ribbon Customization

You can customize the ribbon using the following methods,

| Method | Action |
|-------------------------------------|---|
| hideRibbonTabs | Used to show or hide the existing ribbon tabs. |
| enableRibbonTabs | Used to enable or disable the existing ribbon tabs. |
| addRibbonTabs | Used to add custom ribbon tabs. |
| hideToolbarItems | Used to show or hide the existing ribbon toolbar items. |
| enableToolbarItems | Used to enable or disable the specified toolbar items. |
| addToolbarItems | Used to add the custom items in ribbon toolbar. |
| hideFileMenuItems | Used to show or hide the ribbon file menu items. |
| enableFileMenuItems | Used to enable or disable file menu items. |

| [addFileMenuItems](#) | Used to add custom file menu items. |

The following code example shows the usage of ribbon customization.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent, MenuSelectEventArgs } from '@syncfusion/ej2-angular-spreadsheet';
import { DropDownButton, ItemModel, MenuEventArgs } from "@syncfusion/ej2-splitbuttons";
import { select, createElement } from '@syncfusion/ej2-base';
import { ClickEventArgs } from '@syncfusion/ej2-navigations';
import { dataSource } from '../datasource';
@Component({
  imports: [
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-spreadsheet #spreadsheet (created)="created()"
(fileMenuBeforeOpen)="fileMenuBeforeOpen()"
(fileMenuItemSelect)="fileMenuItemSelect($event)"
[openUrl]="openUrl" [saveUrl]="saveUrl" [showFormulaBar]="false"
[showSheetTabs]="false" [allowInsert]="false" [allowDelete]="false"
[allowMerge]="false">
    <e-sheets>
      <e-sheet>
        <e-ranges>
          <e-range [dataSource]="data"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=180></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=180></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=120></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj: SpreadsheetComponent | undefined;
  data: object[] = dataSource;
  openUrl =
'https://services.syncfusion.com/angular/production/api/spreadsheet/open';
  saveUrl =
'https://services.syncfusion.com/angular/production/api/spreadsheet/save';
  created() {
```

```

        this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:F1');
        // Hiding the `Insert` from ribbon.
        this.spreadsheetObj!.hideRibbonTabs(['Insert']);
        // Set disabled state to `View` ribbon tab.
        this.spreadsheetObj!.enableRibbonTabs(['View'], false);
        /// Adding the `Help` ribbon tab at the last index.
        // Specify the ribbon tab header text in last optional
argument(`insertBefore`) for inserting new tab before any existing tab.
        this.spreadsheetObj!.addRibbonTabs([{ header: { text: 'Help' },
content: [{ text: 'Feedback', tooltipText: 'Feedback',
        click: (args: ClickEventArgs): void => { /* Any click action for
this toolbar item will come here. */ } } ] }]);
        // Hiding the unwanted toolbar items from `Home` by specifying its
index.
        this.spreadsheetObj!.hideToolbarItems('Home', [0, 1, 2, 4, 14, 15,
21, 24]);
        // Set diable state to `Underline`, `Wrap text` toolbar items from
`Home` by specifying the item id.
        this.spreadsheetObj!.enableToolbarItems(
            'Home', [`${this.spreadsheetObj!.element.id}_underline`,
`${this.spreadsheetObj!.element.id}_wrap`], false);
        // Set disable state to `Protect Sheet` toolbar item from `Data` by
mentioning its index.
        this.spreadsheetObj!.enableToolbarItems('Data', [0], false);
        // Adding the new `Custom Formulas` toolbar item under `Formulas`
tab for adding custom formulas.
        this.spreadsheetObj!.addToolbarItems(
            'Formulas', [{ type: 'Separator' }, {
                text: 'Custom Formulas', tooltipText: 'Custom Formulas',
                // You can set click handler for each new custom toolbar
item
                click: (args: ClickEventArgs): void => {
                    // You can add custom formulas here.
                }
            }
        ]);
        // Adding new custom item `Import` after the existing `Open` item.
By default, new item will add after the specified item.
        this.spreadsheetObj!.addFileMenuItems([{ text: 'Import', iconCss:
'e-open e-icons' }], 'Open');
        // Adding new custom item `Export As` after the existing `Save As`
item.
        // Set `insertAfter` optional argument as `false` for adding new
item before the specified item.
        this.spreadsheetObj!.addFileMenuItems(
            [{
                text: 'Export As', iconCss: 'e-save e-icons', items: [{
text: 'XLSX', iconCss: 'e-xlsx e-icons' },
                { text: 'XLS', iconCss: 'e-xls e-icons' }, { text:
'CSV', iconCss: 'e-csv e-icons' } ]
            }],
            'Save As', false);
        // Adding the new `custom dropdown button` in the ribbon toolbar
item under the `Data` tab for adding a custom dropdown button using the
addToolbarItems method in the spreadsheet ribbon.
        this.spreadsheetObj!.addToolbarItems(
            'Data',

```

```

        [
            { type: 'Separator' },
            {
                id: 'custombtn',
                tooltipText: 'Custom Btn',
                template: this.appendDropDownBtn('custombtn'),
            },
        ],
    ],
    7
);
}
fileMenuBeforeOpen (): void {
    // Because the file menu items are created dynamically, you need to
    // perform the hide or show and enable/disable operations
    // under filemenu before open event.
    // Hiding the `Save As` and `Open` item.
    this.spreadsheetObj!.hideFileMenuItems(['Save As', 'Open']);
    // Set disable state to `New` item. You can perform any file menu
    // items customization by specifying the item id,
    // if it has more than one same item text. Set the last argument
    // `isUniqueId` as true for using the item id.
    this.spreadsheetObj!.enableFileMenuItems(['_${this.spreadsheetObj!.element.id}
    _New`], false, true);
}
fileMenuItemSelect (args: MenuSelectEventArgs): void {
    // Custom file menu items select handler
    switch (args.item.text) {
        case 'Import':
            select(`_${this.spreadsheetObj!.element.id}_fileUpload`,
            this.spreadsheetObj!.element).click();
            break;
        case 'XLSX': this.spreadsheetObj!.save({ saveType: 'Xlsx' });
            break;
        case 'XLS': this.spreadsheetObj!.save({ saveType: 'Xls' });
            break;
        case 'CSV': this.spreadsheetObj!.save({ saveType: 'Csv' });
            break;
    }
}
appendDropDownBtn(id: string): HTMLElement {
    let ddlItems: ItemModel[] = [
        {
            text: 'Download Excel',
        },
        {
            text: 'Download CSV',
        },
    ];
    let btnObj: DropDownButton = new DropDownButton({
        items: ddlItems,
        content: 'Download',
        iconCss: 'e-icons e-download',
        select: (args: MenuEventArgs): void => {
            alert(args.item.text + ' clicked');
        },
    });
}

```

```
        btnObj.appendTo(createElement('button', { id: id }));  
        return btnObj.element;  
    }  
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';  
import { AppComponent } from './app.component';  
import 'zone.js';  
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Worksheet](#)
- [Rows and columns](#)

Global local in Angular Spreadsheet component

Localization

The [Localization](#) library allows you to localize the default text content of the Spreadsheet. The Spreadsheet has static text on some features (cell formatting, Merge, Data validation, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and their values are used in the Spreadsheet.

Locale keywords | Text

Cut | Cut

Copy | Copy

Paste | Paste

PasteSpecial | Paste Special

All | All

Values | Values

Formats | Formats

Font | Font

FontSize | Font Size

Bold | Bold

Italic | Italic

Underline | Underline

Strikethrough | Strikethrough

TextColor | Text Color

FillColor | Fill Color

HorizontalAlignment | Horizontal Alignment

AlignLeft | Align Left

AlignCenter | Center

AlignRight | Align Right

VerticalAlignment | Vertical Alignment

AlignTop | Align Top

AlignMiddle | Align Middle

AlignBottom | Align Bottom

InsertFunction | Insert Function

Insert | Insert

Delete | Delete

Rename | Rename

Hide | Hide

Unhide | Unhide

NameBox | Name Box

ShowHeaders | Show Headers

HideHeaders | Hide Headers

ShowGridLines | Show Gridlines

HideGridLines | Hide Gridlines

AddSheet | Add Sheet

ListAllSheets | List All Sheets

FullScreen | Full Screen

CollapseToolbar | Collapse Toolbar

ExpandToolbar | Expand Toolbar

CollapseFormulaBar | Collapse Formula Bar

ExpandFormulaBar | Expand Formula Bar

File | File

Home | Home

Formulas | Formulas

View | View

New | New

Open | Open

SaveAs | Save As

ExcelXlsx | Microsoft Excel

ExcelXls | Microsoft Excel 97-2003

CSV | Comma-separated values

FormulaBar | Formula Bar

Ok | Ok

Close | Close

Cancel | Cancel

Apply | Apply

MoreColors | More Colors

StandardColors | Standard Colors

General | General

Number | Number

Currency | Currency

Accounting | Accounting

ShortDate | Short Date

LongDate | Long Date

Time | Time

Percentage | Percentage

Fraction | Fraction

Scientific | Scientific

Text | Text

NumberFormat | Number Format

MobileFormulaBarPlaceHolder | Enter value or Formula

PasteAlert | You can't paste it here because the copy area and paste area aren't in the same size. Please try pasting in a different range.

DestroyAlert | Are you sure you want to destroy the current workbook without saving and create a new workbook?

SheetRenameInvalidAlert | Sheet name contains invalid character.

SheetRenameEmptyAlert | Sheet name cannot be empty.

SheetRenameAlreadyExistsAlert | Sheet name already exists. Please enter another name.

DeleteSheetAlert | Are you sure you want to delete this sheet?

DeleteSingleLastSheetAlert | A Workbook must contain at least one visible worksheet.

PickACategory | Pick a category

Description | Description

UnsupportedFile | Unsupported File

InvalidUrl | Invalid Url

SUM | Adds a series of numbers and/or cells.

SUMIF | Adds the cells based on the specified condition.

SUMIFS | Adds the cells based on the specified conditions.

ABS | Returns the value of a number without its sign.

RAND | Returns a random number between 0 and 1.

RANDBETWEEN | Returns a random integer based on the specified values.

FLOOR | Rounds a number down to the nearest multiple of a given factor.

CEILING | Rounds a number up to the nearest multiple of a given factor.

PRODUCT | Multiplies a series of numbers and/or cells.

AVERAGE | Calculates average for the series of numbers and/or cells excluding text.

AVERAGEIF | Calculates average for the cells based on the specified criterion.

AVERAGEIFS | Calculates average for the cells based on the specified conditions.

AVERAGEA | Calculates the average for the cells evaluating TRUE as 1 text and FALSE as 0.

COUNT | Counts the cells that contain numeric values in a range.

COUNTIF | Counts the cells based on the specified condition.

COUNTIFS | Counts the cells based on specified conditions.

COUNTA | Counts the cells that contain values in a range.

MIN | Returns the smallest number of the given arguments.

MAX | Returns the largest number of the given arguments.

DATE | Returns the date based on the given year, month, and day.

DAY | Returns the day from the given date.

DAYS | Returns the number of days between two dates.

IF | Returns value based on the given expression.

IFS | Returns value based on the given multiple expressions.

AND | Returns TRUE if all the arguments are TRUE otherwise returns FALSE.

OR | Returns TRUE if any of the arguments are TRUE otherwise returns FALSE.

IFERROR | Returns value if no error found else it will return specified value.

CHOOSE | Returns a value from the list of values based on the index number.

INDEX | Returns the value of the cell in a given range based on row and column number.

FIND | Returns the position of a string within another string which is case sensitive.

CONCATENATE | Combines two or more strings together.

CONCAT | Concatenates a list or a range of text strings.

SUBTOTAL | Returns subtotal for a range using the given function number.

RADIANS | Converts degrees into radians.

MATCH | Returns the relative position of a specified value in the given range.

DefineNameExists | This name already exists try a different name.

CircularReference | When a formula refers to one or more circular references this may result in an incorrect calculation.

ShowRowsWhere | Show rows where |

CustomFilterDatePlaceholder | Choose a date

CustomFilterPlaceholder | Enter the value

CustomFilter | Custom Filter

Between | Between

MatchCase | Match Case

DateTimeFilter | DateTime Filters

Undo | Undo

Redo | Redo

DateFilter | Date Filters

TextFilter | Text Filters

NumberFilter | Number Filters

ClearFilter | Clear Filter

NoResult | No Matches Found

FilterFalse | False

FilterTrue | True

Blanks | Blanks

SelectAll | Select All

GreaterThanOrEqual | Greater Than Or Equal

GreaterThan | Greater Than

LessThanOrEqual | Less Than Or Equal

LessThan | Less Than

NotEqual | Not Equal

Equal | Equal
Contains | Contains
EndsWith | Ends With
StartsWith | Starts With
ClearButton | Clear
FilterButton | Filter
CancelButton | Cancel
OKButton | OK
Search | Search
Link | Link
Hyperlink | Hyperlink
EditHyperlink | Edit Hyperlink
OpenHyperlink | Open Hyperlink
RemoveHyperlink | Remove Hyperlink
InsertLink | Insert Link
EditLink | Edit Link
WebPage | WEB PAGE
ThisDocument | THIS DOCUMENT
DisplayText | Display Text
Url | URL
CellReference | Cell Reference
Sheet | Sheet
DefinedNames | Defined Names
EnterTheTextToDisplay | Enter the text to display
EnterTheUrl | Enter the URL
ProtectSheet | Protect Sheet
UnprotectSheet | Unprotect Sheet
SelectCells | Select cells
FormatCells | Format cells
FormatRows | Format rows
Format Columns | Format columns
InsertLinks | Insert links
ProtectContent | Protect the contents of locked cells

ProtectAllowUser | Allow all users of this worksheet to |
EditAlert | The cell you're trying to change is protected. To make a change, unprotect the sheet.
FindReplaceTooltip | Find & Replace
InsertingEmptyValue | Reference value is not valid.
ByRow | By Row
ByColumn | By Column
MatchExactCellElements | Match Exact Cell Contents
EnterCellAddress | Enter Cell Address
FindAndReplace | Find and Replace
ReplaceAllEnd | matches replaced with.
FindNextBtn | Find Next
FindPreviousBtn | Find Previous
ReplaceBtn | Replace
ReplaceAllBtn | Replace All
GotoHeader | Go To
GotoSpecialHeader | Go To Special
SearchWithin | Search within
SearchBy | Search by
Reference | Reference
Workbook | Workbook
NoElements | We couldn't find what you were looking for.
FindWhat | Find what
ReplaceWith | Replace with
EnterValue | Enter Value
DefineNameInvalid | The name that you entered is not valid.
FindValue | Find Value
ReplaceValue | Replace Value
DataValidation | Data Validation
CLEARALL | CLEAR ALL
APPLY | APPLY
CellRange | Cell Range
Allow | Allow
Data | Data

Minimum | Minimum

Maximum | Maximum

IgnoreBlank | Ignore blank

WholeNumber | Whole Number

Decimal | Decimal

Date | Date

TextLength | Text Length

List | List

NotBetween | Not between

EqualTo | Equal to

NotEqualTo | Not equal to

Greaterthan | Greater than

Lessthan | Less than

GreaterThanOrEqualTo | Greater than or equal to

LessThanOrEqualTo | Less than or equal to

InCellDropDown | In-cell-dropdown

Sources | Sources

Value | Value

Retry | Retry

DialogError | The list source must be a reference to a single row or column.

ValidationError | This value doesn't match the data validation restrictions defined for the cell.

HideRow | Hide Row

HideRows | Hide Rows

UnHideRows | UnHide Rows

HideColumn | Hide Column

HideColumns | Hide Columns

UnHideColumns | UnHide Columns

InsertRow | Insert Row

InsertRows | Insert Rows

InsertColumn | Insert Column

InsertColumns | Insert Columns

DeleteRow | Delete Row

DeleteRows | Delete Rows

DeleteColumn | Delete Column

DeleteColumns | Delete Columns

Borders | Borders

TopBorders | Top Borders

LeftBorders | Left Borders

RightBorders | Right Borders

BottomBorders | Bottom Borders

AllBorders | All Borders

HorizontalBorders | Horizontal Borders

VerticalBorders | Vertical Borders

OutsideBorders | Outside Borders

InsideBorders | Inside Borders

NoBorders | No Borders

BorderColor | Border Color

BorderStyle | Border Style

INTERCEPT | Calculates the point of the Y-intercept line via linear regression.

SLOPE | Returns the slope of the line from linear regression of the data points.

FreezePanels | Freeze Panels

FreezeRows | Freeze Rows

FreezeColumns | Freeze Columns

UnfreezePanels | Unfreeze Panels

UnfreezeRows | Unfreeze Rows

UnfreezeColumns | Unfreeze Columns

MergeCells | Merge Cells

MergeAll | Merge All

MergeHorizontally | Merge Horizontally

MergeVertically | Merge Vertically

Unmerge | Unmerge

UnmergeCells | Unmerge Cells

SortAscending | Ascending

SortDescending | Descending

CustomSort | Custom Sort

ClearAllFilter | Clear

ReapplyFilter | Reapply

Clear | Clear

ClearContents | Clear Contents

ClearAll | Clear All

ClearFormats | Clear Formats

ClearHyperlinks | Clear Hyperlinks

Image | Image

AddColumn | Add Column

SortBy | Sort by

ThenBy | Then by

Chart | Chart

Column | Column

Bar | Bar

Area | Area

Pie | Pie

Doughnut | Doughnut

PieAndDoughnut | Pie/Doughnut

Line | Line

Radar | Radar

Scatter | Scatter

ChartDesign | Chart Design

ClusteredColumn | Clustered Column

StackedColumn | Stacked Column

StackedColumn100 | 100% Stacked Column

ClusteredBar | Clustered Bar

StackedBar | Stacked Bar

StackedBar100 | 100% Stacked Bar

StackedArea | Stacked Area

StackedArea100 | 100% Stacked Area

StackedLine | Stacked Line

StackedLine100 | 100% Stacked Line

AddChartElement | Add Chart Element

Axes | Axes

AxisTitle | Axis Title

ChartTitle | Chart Title

DataLabels | Data Labels

Gridlines | Gridlines

Legends | Legends

PrimaryHorizontal | Primary Horizontal

PrimaryVertical | Primary Vertical

None | None

AboveChart | Above Chart

Center | Center

InsideEnd | Inside End

InsideBase | Inside Base

OutsideEnd | OutSide End

PrimaryMajorHorizontal | Primary Major Horizontal

PrimaryMajorVertical | Primary Major Vertical

PrimaryMinorHorizontal | Primary Minor Horizontal

PrimaryMinorVertical | Primary Minor Vertical

Right | Right

Left | Left

Bottom | Bottom

Top | Top

SwitchRowColumn | Switch Row/Column

ChartTheme | Chart Theme

ChartType | Chart Type

Material | Material

Fabric | Fabric

Bootstrap | Bootstrap

HighContrastLight | HighContrastLight

MaterialDark | MaterialDark

FabricDark | FabricDark

HighContrast | HighContrast

BootstrapDark | BootstrapDark

Bootstrap4 | Bootstrap4

VerticalAxisTitle | Vertical Axis Title

HorizontalAxisTitle | Horizontal Axis Title

EnterTitle | Enter Title

ProtectWorkbook | Protect Workbook

Password | Password (optional) |

unProtectPassword | Password

EnterThePassword | Enter the password

ConfirmPassword | Confirm Password

EnterTheConfirmPassword | Re-enter your password

PasswordAlert | Confirmation password is not identical

UnProtectWorkbook | Unprotect Workbook

UnProtectPasswordAlert | The password you supplied is not correct.

InCorrectPassword | Unable to open the file or worksheet with the given password.

PasswordAlertMsg | Please enter the password

ConfirmPasswordAlertMsg | Please enter the confirm password

IsProtected | is protected

Loading translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Spreadsheet in **French** culture. In the below sample we have translated the ribbon tab names and Home tab content (clipboard, cell style).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { L10n } from '@syncfusion/ej2-base';
import { data } from './datasource';
L10n.load({
  'fr-CH': {
    'spreadsheet': {
      'File': 'Fichier',
      'Home': 'Accueil',
      'Insert': 'Insérer',
      'Formulas': 'Formules',
      'Data': 'Les données',
      'View': 'Vue',
      'Cut': 'Coupe',
      'Copy': 'Copie',
      'Paste': 'Pâte',
      'PasteSpecial': 'Pâte spéciale',
      'All': 'Tous les',
      'Values': 'Valeurs',
```

```

        'Formats': 'Les formats',
        'Font': 'fonte',
        'FontSize': 'Taille de police',
        'Bold': 'Audacieux',
        'Italic': 'Italique',
        'Underline': 'Souligner',
        'Strikethrough': 'Barré',
        'TextColor': 'Couleur du texte',
        'FillColor': 'La couleur de remplissage',
        'HorizontalAlignment': 'Alignement horizontal',
        'AlignLeft': 'Alignez à gauche',
        'AlignCenter': 'centre',
        'AlignRight': 'Aligner à droite',
        'VerticalAlignment': 'Alignement vertical',
        'AlignTop': 'Aligner en haut',
        'AlignMiddle': 'Aligner le milieu',
        'AlignBottom': 'Aligner le bas',
        'InsertFunction': 'Insérer une fonction',
        'Delete': 'Effacer',
        'Rename': 'Rebaptiser',
        'Hide': 'Cacher',
        'Unhide': 'Démasquer',
        'NumberFormat': 'Nombre Format',
    }
}
});
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet locale='fr-CH'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='data'></e-range></e-ranges><e-columns><e-column [width]=90></e-column><e-column [width]=100></e-column><e-column [width]=96></e-column><e-column [width]=120></e-column><e-column [width]=130></e-column><e-column [width]=120></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public data?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.data = data;
  }
};

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Internationalization

The Internationalization library is used to globalize number, date, and time values in the spreadsheet component.

For importing json files in your application, you need to include the json-typings.d.ts file.

```
`typescript
declare module "*.json" {
  const value: any;
  export default value;
}
```

You need to load culture format files in **ngOnInit** function.

The following example demonstrates the Spreadsheet in French [fr-CH] culture. In the below sample we have globalized the Date(Date column), Time(Time column), and Currency(Amount column) formats.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { L10n, loadCldr, setCulture, setCurrencyCode } from
 '@syncfusion/ej2-base';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { defaultData } from './datasource';
L10n.load({
  'fr-CH': {
    'spreadsheet': {
      'File': 'Fichier',
      'Home': 'Accueil',
      'Insert': 'Insérer',
      'Formulas': 'Formules',
      'Data': 'Les données',
      'View': 'Vue',
      'Cut': 'Coupe',
      'Copy': 'Copie',
      'Paste': 'Pâte',
      'PasteSpecial': 'Pâte spéciale',
      'All': 'Tous les',
      'Values': 'Valeurs',
      'Formats': 'Les formats',
      'Font': 'fonte',
      'FontSize': 'Taille de police',
      'Bold': 'Audacieux',
      'Italic': 'Italique',
      'Underline': 'Souligner',
      'Strikethrough': 'Barré',
      'TextColor': 'Couleur du texte',
      'FillColor': 'La couleur de remplissage',
      'HorizontalAlignment': 'Alignement horizontal',
      'AlignLeft': 'Alignez à gauche',
```

```

        'AlignCenter': 'centre',
        'AlignRight': 'Aligner à droite',
        'VerticalAlignment': 'Alignement vertical',
        'AlignTop': 'Aligner en haut',
        'AlignMiddle': 'Aligner le milieu',
        'AlignBottom': 'Aligner le bas',
        'InsertFunction': 'Insérer une fonction',
        'Delete': 'Effacer',
        'Rename': 'Rebaptiser',
        'Hide': 'Cacher',
        'Unhide': 'Démasquer',
        'NumberFormat': 'Nombre Format',
    }
}
});
@Component({
imports: [

    SpreadsheetAllModule
],
standalone: true,
selector: 'app-container',
template: "<ejs-spreadsheet #spreadsheet locale='fr-CH'
(created)='created()'> <e-sheets> <e-sheet> <e-ranges> <e-range
[dataSource]='defaultData'></e-range></e-ranges><e-columns><e-column
[width]=90></e-column><e-column [width]=100></e-column><e-column
[width]=96></e-column><e-column [width]=120></e-column><e-column
[width]=130></e-column><e-column [width]=120></e-column></e-columns></e-
sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    defaultData?: Object[];
    ngOnInit(): void {
        this.defaultData = defaultData;
        setCulture('fr-CH');
        setCurrencyCode('EUR');
        loadCldr('./currencies.json',
            './numbers.json',
            './ca-gregorian.json',
            './timeZoneNames.json',
            './numberingSystems.json');
    }
    created() {
        //Applies cell and number formatting to specified range of the
        active sheet
        this.spreadsheetObj?.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:F1');
        this.spreadsheetObj?.numberFormat('$#,##0.00', 'F2:F11');
    };
};
};

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Spreadsheet component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Spreadsheet, set the [enableRtl](#) to true.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { L10n } from '@syncfusion/ej2-base';
import { data } from './datasource';
L10n.load({
  'ar-AE': {
    'spreadsheet': {
      "File": "ملف",
      "Home": "هم",
      "Insert": "إدراج",
      "Formulas": "الصيغ",
      "View": "معاينة",
      "Data": "البيانات",
      "Cut": "قطع",
      "Copy": "نسخ",
      "Paste": "معجون",
      "PasteSpecial": "لصق خاص",
      "All": "جميع",
      "Values": "القيم",
      "Formats": "شكل",
      "Font": "الخط",
      "FontSize": "حجم الخط",
      "Bold": "جريء",
      "Italic": "مائل",
      "Underline": "أكد",
      "Strikethrough": "يتوسطه",
      "TextColor": "لون الخط",
      "FillColor": "لون التعبئة",
      "HorizontalAlignment": "المحاذاة الأفقية",
      "AlignLeft": "محاذاة إلى اليسار",
      "AlignCenter": "مركز",
      "AlignRight": "محاذاة إلى اليمين",
      "VerticalAlignment": "محاذاة عمودية",
      "AlignTop": "محاذاة أعلى",
      "AlignMiddle": "محاذاة الأوسط",
      "AlignBottom": "أسفل محاذاة",
      "InsertFunction": "إدراج وظيفة",
      "Delete": "حذف",
      "Rename": "إعادة تسمية",
      "Hide": "إخفاء",
      "Unhide": "ظهار"
    }
  }
});
```

```

    }
  });
  @Component({
    imports: [

      SpreadsheetAllModule
    ],
    standalone: true,
    selector: 'app-container',
    template: "<ejs-spreadsheet #spreadsheet locale='ar-AE'
[enableRtl]='true'> <e-sheets> <e-sheet> <e-ranges> <e-range
[dataSource]='data'></e-range></e-ranges><e-columns><e-column
[width]=90></e-column><e-column [width]=100></e-column><e-column
[width]=96></e-column><e-column [width]=120></e-column><e-column
[width]=130></e-column><e-column [width]=120></e-column></e-columns></e-
sheet></e-sheets></ejs-spreadsheet>"
  })
  export class AppComponent implements OnInit {
    public data?: object[];
    @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
    ngOnInit(): void {
      this.data = data;
    }
  };

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Localization](#)

Undo redo in Angular Spreadsheet component

Undo option helps you to undo the last action performed and **Redo** option helps you to do the same action which is reverted in the Spreadsheet. You can use the [allowUndoRedo](#) property to enable or disable undo redo functionality in spreadsheet.

* The default value for **allowUndoRedo** property is **true**.

By default, the **UndoRedo** module is injected internally into Spreadsheet to perform undo redo.

Undo

It reverses the last action you performed with Spreadsheet. Undo can be done by any of the following ways:

- Select the undo item from HOME tab in Ribbon toolbar.
- Use **Ctrl + Z** keyboard shortcut to perform the undo.
- Use the [undo](#) method programmatically.

Redo

It reverses the last undo action you performed with Spreadsheet. Redo can be done by any of the following ways:

- Select the redo item from HOME tab in Ribbon toolbar.
- Use **Ctrl + Y** keyboard shortcut to perform the redo.
- Use the [redo](#) method programmatically.

Update custom actions in UndoRedo collection

You can update your own custom actions in UndoRedo collection, by using the [updateUndoRedoCollection](#) method. And also customize the undo redo operations of your custom action by using [actionComplete](#) event.

The following code example shows **How to update and customize your own actions for undo redo functionality in the Spreadsheet control.**

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { defaultData } from './datasource';
import { addClass, removeClass } from '@syncfusion/ej2-base';
import { SpreadsheetComponent, getRangeIndexes } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<button id='customBtn' class='e-btn' (click)='updateCollection()'> add/remove Class</button> <ejs-spreadsheet #spreadsheet (actionComplete)='actionComplete($event)'> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='defaultData'></e-range></e-ranges><e-columns><e-column [width]=130></e-column><e-column [width]=92></e-column><e-column [width]=96></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public defaultData?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.defaultData = defaultData;
  }
  actionComplete (args: any) {
    let actionEvents: any = args;
    if (actionEvents.eventArgs.action == "customCSS") {
```

```

        let Element:HTMLElement =
this.spreadsheetObj!.getCell(actionEvents.eventArgs.rowIdx,actionEvents.eventArgs.colIdx);
        if (actionEvents.eventArgs.requestType == "undo") {
            removeClass([Element],'customClass'); // To remove the
custom class in undo action
        }
        else {
            addClass([Element],'customClass');// To add the custom class
in redo action
        }
    }
    updateCollection() {
        var cell = this.spreadsheetObj!.getActiveSheet().activeCell;
        var cellIdx = getRangeIndexes(cell as any);
        var Element= this.spreadsheetObj!.getCell(cellIdx[0], cellIdx[1]);
        if (!Element.classList.contains("customClass")) {
            Element.classList.add('customClass'); // To add the custom class
in active cell element
            this.spreadsheetObj!.updateUndoRedoCollection({ eventArgs: {
class: 'customClass', rowIdx: cellIdx[0], colIdx: cellIdx[1], action:
'customCSS' } }); // To update the undo redo collection
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

See Also

- [Sorting](#)
- [Filtering](#)
- [Hyperlink](#)

Accessibility in Angular Spreadsheet component

The Spreadsheet 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 Spreadsheet 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 Spreadsheet component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Spreadsheet component:

| Attributes | Purpose |

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

| **grid** (role) | This role is added to the spreadsheet content table and describes the collection of rows and columns. |

| **gridcell** (role) | This role is added to the cell element and describes the rows **<td>** element. |

- | **rowheader** (role) | This role is added to the row header and describes the header of the rows. |
- | **colheader** (role) | This role is added to the column header and describes the header of the columns. |
- | **aria-rowindex** (attribute) | This attribute describes the table's row index in the spreadsheet. |
- | **aria-colindex** (attribute) | This attribute describes the table's column index in the spreadsheet. |
- | **aria-selected** (attribute) | This attribute describes an item's (cell, menu, checkbox, etc.) current selected state in the spreadsheet. |
- | **aria-rowcount** (attribute) | This attribute describes the total number of rows in the table. |
- | **aria-colcount** (attribute) | This attribute describes the total number of columns in the table. |
- | **aria-busy** (attribute) | This attribute describes a currently updated or modified element. |
- | **aria-label** (attribute) | This attribute describes the accessible name for the interactive elements. |
- | **textbox** (role) | This role is assigned to the textbox that accepts text input. |
- | **menu** (role) | This role has been added to the menu and describes the menu items. |
- | **aria-expanded** (attribute) | This attribute describes the control (for example, dropdown) is expanded or collapsed. |
- | **aria-multiline** (attribute) | This attribute defines what the Alt + Enter key does in the spreadsheet editor. |

Keyboard interaction

The Spreadsheet 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 Spreadsheet component.

- | Press | To do this |
- |-----|-----|
- | Up arrow | Navigate from the active cell to the previous cell in the same column. |
- | Down arrow | Navigate from the active cell to the next cell in the same column. |
- | Left arrow | Navigate from the active cell to the previous cell in the same row. |
- | Right arrow | Navigate from the active cell to the next cell in the same row. |
- | Tab | Navigate the active cell to the next cell in the same row. |
- | Shift + Tab | Navigate the active cell to the previous cell in the same row. |
- | Home | Moves the selection to starting column in worksheet. |
- | Ctrl + Home | Move the selection to the first visible cell on a worksheet. |
- | Shift + Home | Extend the cell selection to the first column of a worksheet. |
- | Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |
- | Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |
- | Page Up | Move page up. |

- | Page Down | Move page down. |
- | Shift + Page Up | Perform page up by selecting all cells between. |
- | Shift + Page Down | Perform page down by selecting all cells between. |
- | Ctrl + Up | Navigate to the non-blank cell before the active cell in the same column. |
- | Ctrl + Down | Navigate to the last non-blank cell in the same column as the active cell. |
- | Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |
- | Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |
- | Shift + Up | Extend the selection of cells to the previous row. |
- | Shift + Down | Extend the selection of cells to the next row. |
- | Shift + Left | Extend the selection of cells to the previous column. |
- | Shift + Right | Extend the selection of cells to the next column. |
- | Ctrl + Shift + Up | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |
- | Ctrl + Shift + Down | Extend the cell selection to the last non-blank cell in the same column as the active cell. |
- | Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |
- | Enter | Navigate the active cell to the next cell in the same column. |
- | Shift + Enter | Navigate to the previous cell in the same column from the active cell. |
- | Alt + Enter | While editing, add a new line. |
- | Enter | Complete the cell editing and select the cell below in the same column. |
- | Shift + Enter | Complete the cell editing and select the cell above in the same column. |
- | Tab | Complete the cell editing and select the next cell in the same row. |
- | Shift + Tab | Complete the cell editing and select the previous cell in the same row. |
- | Alt | Focus on the active ribbon tab. |
- | Left | Move the focus to the previous items in the ribbon content. |
- | Right | Move the focus to the next items in the ribbon content. |
- | Alt + Down | Open the ribbon dropdown menu. |
- | Esc / Alt + Up | Close the ribbon dropdown menu. |

Ensuring accessibility

The Spreadsheet component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Spreadsheet component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Spreadsheet component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Styles in Angular Spreadsheet component

To modify the Spreadsheet appearance, you need to override the default CSS of the spreadsheet. Please find the CSS structure that can be used to modify the Spreadsheet appearance. Also, you have an option to create your own custom theme for all the JavaScript controls using our [Theme Studio](#).

Customizing the Spreadsheet

Use the below CSS to customize the Spreadsheet root element.

```
`css
.e-spreadsheet {
font-family: cursive;
}
```

Header

Customizing the Spreadsheet Ribbon

Use the below CSS to customize the Spreadsheet Ribbon.

```
`css
.e-spreadsheet .e-ribbon {
background-color: #808080;
}
```

Customizing the Spreadsheet formula bar panel

You can customize the Spreadsheet formula bar panel by using this CSS.

```
`css
.e-spreadsheet .e-formula-bar-panel {
border: none;
}
```

Customizing the Spreadsheet formula bar text

You can customize the Spreadsheet formula bar text by using this CSS.

```
`css
.e-spreadsheet .e-formula-bar-panel .e-formula-bar {
```

```
font-weight: bold;
```

```
}
```

```
,
```

Sheet

Customizing the Spreadsheet sheet element

Using this CSS, you can customize the Spreadsheet sheet element.

```
`css
```

```
.e-spreadsheet .e-sheet-panel .e-sheet {
```

```
border-color: #000000;
```

```
}
```

```
,
```

Customizing the Spreadsheet sheet header

Use the below CSS to customize the Spreadsheet sheet header.

```
`css
```

```
.e-spreadsheet .e-sheet-panel .e-sheet .e-header-panel {
```

```
font-style: oblique;
```

```
}
```

```
,
```

Customizing the Spreadsheet row header

Use the below CSS to customize the Spreadsheet row header.

```
`css
```

```
.e-spreadsheet .e-row-header .e-header-cell {
```

```
color: #0000FF !important;
```

```
}
```

```
,
```

Customizing the Spreadsheet column header

Use the below CSS to customize the Spreadsheet column header.

```
`css
```

```
.e-spreadsheet .e-column-header .e-header-cell {
```

```
color: #0000FF !important;
```

```
}
```

```
,
```

Customizing the Spreadsheet selection element

Customize the Spreadsheet selection element.

```
`css
```

```
.e-spreadsheet .e-selection {  
border-color: #0000FF;  
}  
`css
```

Customizing the Spreadsheet active cell element

Customize the Spreadsheet active cell element.

```
`css  
.e-spreadsheet .e-active-cell {  
border-color: #0000FF;  
}  
`css
```

Customizing the Spreadsheet cell element

Using this CSS, you can customize the Spreadsheet cell element.

```
`css  
.e-spreadsheet .e-cell {  
background-color: #0078d7 !important;  
}  
`css
```

Ribbon Items

Customizing the Spreadsheet sorting icon

Use the below CSS to customize the Spreadsheet sorting icon in the Spreadsheet ribbon. You can use the available Syncfusion [icons](#) based on your theme.

```
`css  
.e-spreadsheet .e-sort-filter-icon:before {  
background-color: #deecf9;  
content: '\e306';  
}  
`css
```

Customizing the filter dialog content

Use the below CSS to customize the Spreadsheet filter dialog content element.

```
`css  
.e-spreadsheet .e-filter-popup .e-dlg-content {  
background-color: #deecf9;  
}  
`css
```


Customizing the filter dialog footer

Spreadsheet filter dialog footer element can be customized by using the below CSS.

```
`css
.e-spreadsheet .e-filter-popup .e-footer-content {
background-color: #ccffff;
}
`
```

Customizing the filter dialog input element

Use the below CSS to customize the Spreadsheet filter dialog input element.

```
`css
.e-spreadsheet .e-filter-popup .e-input-group input.e-input{
font-family: cursive;
}
`
```

Customizing the filter dialog button element

Use the below CSS to customize the Spreadsheet filter dialog button element.

```
`css
.e-spreadsheet .e-filter-popup .e-btn{
font-family: cursive;
}
`
```

Customizing the Excel filter dialog number filters element

Spreadsheet Excel filter dialog number filters element can be customized by using the below CSS.

```
`css
.e-spreadsheet .e-filter-popup .e-contextmenu-wrapper ul{
background-color: #deecf9;
}
`
```

*Footer**Customizing the Spreadsheet sheet tab panel*

Spreadsheet sheet tab panel can be customized by using the below CSS.

```
`css
.e-spreadsheet .e-sheet-tab-panel {
background: #08fbfb;
}
`
```

Customizing the Spreadsheet sheet tab

Spreadsheet sheet tab element can be customized by using the below CSS.

```
`css
.e-spreadsheet .e-sheet-tab-panel .e-tab-header .e-toolbar-item.e-active .e-tab-wrap .e-tab-text {
font-weight: bold;
}
`
```

Ej1 api migration in Angular Spreadsheet component

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

Editing

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

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

| Enables or disables the editing feature | **Property:** *allowEditing*

<ej-spreadsheet
[allowEditing]="true"></ej-spreadsheet> | **Property:** *allowEditing*

<ejs-spreadsheet
[allowEditing]="true"></ejs-spreadsheet> |

| Edit a particular cell | **Method:** *XLEdit.editCell*

<ej-spreadsheet
id="spreadsheet"></ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); xObj.XLEdit.editCell(1, 1); } | **Method:** *startEdit*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
<TS>
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.selectRange('A1');
this.spreadsheetObj.startEdit(); |

| Save the edited cell value | **Method:** *XLEdit.saveCell*

<ej-spreadsheet
id="spreadsheet"></ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); xObj.XLEdit.saveCell(); } | **Method:** *endEdit*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
<TS>
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.endEdit(); |

| Update a particular cell value | **Method:** *XLEdit.updateCell*

<ej-spreadsheet
id="spreadsheet"></ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); xObj.XLEdit.updateCell({rowIndex: 1, colIndex: 1,
"product"}); } | **Method:** *updateCell*

<ejs-spreadsheet #default>
</ejs-
spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj:
SpreadsheetComponent;
this.spreadsheetObj.updateCell({ value: 'product' }, 'A1'); |

| Triggers when the cell is edited | **Event:** *cellEdit*

<ej-
spreadsheet id="spreadsheet" (cellEdit)="onCellEdit(\$event)">
</ej-
spreadsheet>
<TS>
onCellEdit(args){ } | **Event:** *cellEdit*

<ejs-
spreadsheet (cellEdit)="onCellEdit(\$event)">
</ejs-
spreadsheet>
<TS>
onCellEdit(args){ } |

| Triggers when the edited cell is saved | **Event:** *cellSave*

<ej-spreadsheet id="spreadsheet" (cellSave)="onCellSave(\$event)">
</ej-spreadsheet>
TS
onCellSave(args){ } | **Event:** *cellSave*

<ejs-spreadsheet (cellSave)="onCellSave(\$event)">
</ejs-spreadsheet>
TS
onCellSave(args){ }

Selection

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

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

| Enables or disables the selection feature | **Property:** *allowSelection*

<ej-spreadsheet [allowSelection]="true">
</ej-spreadsheet> | **Property:** *selectionSettings.mode*

<ejs-spreadsheet [selectionSettings]="selectionSettings">
</ejs-spreadsheet>
TS
this.selectionSettings = { mode: 'Multiple' }; |

| Defines active cell in the sheet | **Property:** *selectionSettings.activeCell*

<ej-spreadsheet [selectionSettings]="selectionSettings">
</ej-spreadsheet>
Ts
this.selectionSettings = { activeCell: "A1" }; | **Property:** *activeCell*

<ejs-spreadsheet>
<e-sheets><e-sheet activeCell="A1"></e-sheet></e-sheets>
</ejs-spreadsheet> |

| Set selection unit | **Property:** *selectionSettings.selectionUnit*

<ej-spreadsheet [selectionSettings]="selectionSettings">
</ej-spreadsheet>
Ts
this.selectionSettings = { selectionUnit: ej.Spreadsheet.SelectionUnit.Single }; | **Property:** *selectionSettings.mode*

<ejs-spreadsheet [selectionSettings]="selectionSettings">
</ejs-spreadsheet>
Ts
this.selectionSettings = { mode: 'Single' }; |

| Select the specified range of cells | **Method:** *XLSelection.selectRange*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLSelection.selectRange("A1:B2"); } | **Method:** *selectRange*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.selectRange("A1:B2"); |

| Select a cell or range | **Method:** *performSelection*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.performSelection("B1:C3"); } | **Method:** *selectRange*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.selectRange("B1:C3"); |

| Triggers before the cell selection | **Event:** *beforeCellSelect*

<ej-spreadsheet id="spreadsheet" (beforeCellSelect)="onBeforeCellSelect(\$event)">
</ej-spreadsheet>
TS
onBeforeCellSelect(args){ } | **Event:** *beforeSelect*

<ejs-spreadsheet (beforeSelect)="onBeforeSelect(\$event)">
</ejs-spreadsheet>
TS
onBeforeSelect(args){ }

| Triggers when the cell is selected | **Event:** *cellSelected*

<ej-spreadsheet id="spreadsheet" (cellSelected)="onCellSelected(\$event)">
</ej-

spreadsheet>
TS
onCellSelected(args){ }| **Event:** *select*

<ej-spreadsheet (select)="onSelect(\$event)">
</ej-spreadsheet>
TS
onSelect(args){ }

Clipboard

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

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

| Enables or disables the clipboard feature | **Property:** *allowClipboard*

<ej-spreadsheet [allowClipboard]="true">
</ej-spreadsheet>| **Property:** *enableClipboard*

<ej-spreadsheet [enableClipboard]="true">
</ej-spreadsheet>|

| Copy the selected cells | **Method:** *XLClipboard.copy*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLClipboard.copy(); }| **Method:** *copy*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.copy("A1");|

| Cut the selected cells | **Method:** *XLClipboard.cut*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLClipboard.cut(); }| **Method:** *cut*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.cut("A1");|

| Paste the cut or copied cells data | **Method:** *XLClipboard.paste*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLClipboard.paste(); }| **Method:** *paste*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.paste("B1");|

Formulas

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

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

| Enables or disables the formula bar | **Property:** *allowFormulaBar*

<ej-spreadsheet [allowFormulaBar]="true">
</ej-spreadsheet>| **Property:** *showFormulaBar*

<ej-spreadsheet [showFormulaBar]="true">
</ej-spreadsheet>|

| Set name manager | **Property:** *nameManager*

<ej-spreadsheet [nameManager]="nameManager">
</ej-spreadsheet>
TS
this.nameManager:[{ name: "inputRange", refersto: "=Sheet1!\$A\$1:\$A\$2" }];| **Property:** *definedNames*

<ej-spreadsheet [definedNames]="definedNames">
</ej-spreadsheet>
TS
this.definedNames= [{ name: 'namedRange1', refersTo: 'Sheet1!A1:B5' }];|

| Add the custom formulas | **Property:** *customFormulas*

<ej-spreadsheet [customFormulas]="customFormulas">
</ej-spreadsheet>
TS
this.customFormulas = [{ formulaName:"CUSTOMTOTAL", functionName:"customTotal" }];
customTotal(args) { }| **Method:** *addCustomFunction*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj:

```
SpreadsheetComponent; <br>this.spreadsheetObj.addCustomFunction("CustomFuntion",
"SQRT"); <br>window.CustomFuntion = num => Math.sqrt(num); |
```

| Method to Add custom formulas | **Method:** *addCustomFormula*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$(("#spreadsheet").data("ejSpreadsheet")); xObj.addCustomFormula("CUSTOMTOTAL", "customTotal"); }
customTotal(args){ } | **Method:** *addCustomFunction*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj:

```
SpreadsheetComponent; <br>this.spreadsheetObj.addCustomFunction("CustomFuntion",
"SQRT"); <br>window.CustomFuntion = num => Math.sqrt(num); |
```

| Add a name for a range in the name manager | **Method:** *XL Ribbon.addNamedRange*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$(("#spreadsheet").data("ejSpreadsheet")); xObj.XLRibbon.addNamedRange("PRICE_LIST", "=Sheet1!\$A\$2:\$A\$7"); } | **Method:** *addDefinedName*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.addDefinedName({name: 'value', refersTo: '=Sheet1!B2' }); |

| Delete the defined name for a range in the name manager | **Method:** *XL Ribbon.removeNamedRange*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$(("#spreadsheet").data("ejSpreadsheet")); xObj.XLRibbon.removeNamedRange("PRICE_LIST"); } | **Method:** *removeDefinedName*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.removeDefinedName('value'); |

Formatting

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

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

| Enables or disables the cell format feature | **Property:** *allowCellFormatting*

<ej-spreadsheet [allowCellFormatting]="true">
</ej-spreadsheet> | **Property:** *allowCellFormatting*

<ejs-spreadsheet [allowCellFormatting]="true">
</ejs-spreadsheet> |

| Enables or disables the conditional format feature | **Property:** *allowConditionalFormats*

<ej-spreadsheet [allowConditionalFormats]="true">
</ej-spreadsheet> | **Property:** *allowConditionalFormat*

<ejs-spreadsheet [allowConditionalFormat]="true">
</ejs-spreadsheet> |

| Enables or disables the cell border feature | **Property:** *formatSettings.allowCellBorder*

<ej-spreadsheet [formatSettings.allowCellBorder]=true>
</ej-spreadsheet> | By default, it is enabled by enabling the *allowCellFormatting* property |

| Enables or disables the decimal places | **Property:** *formatSettings.allowDecimalPlaces*

<ej-spreadsheet [formatSettings.allowDecimalPlaces]=true>
</ej-spreadsheet> | By default, it is enabled by enabling the *allowNumberFormatting* property |

| Specifies the conditional formatting for the range of cells | **Property:** *sheets.cFormatRule*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
Ts
this.sheets = [{ cFormatRule: [{ action: ej.Spreadsheet.CFormatRule.LessThan, inputs: ["30"], color: ej.Spreadsheet.CFormatHighlightColor.RedFillwithDarkRedText, range: "A1:E1" }] }]| **Property:** *sheets.conditionalFormats*

<ej-spreadsheet>
<e-sheets><e-sheet><e-conditionalformats><e-conditionalformat type="GreaterThan" cFColor="RedFT" value="700" range="B2:B9"></e-conditionalformat></e-conditionalformats></e-sheet></e-sheets>
</ej-spreadsheet>`|

| Clear the applied conditional formatting rules | **Method:** *XLFormat.clearCF*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLFormat.clearCF([1, 0, 7, 0]); }| **Method:** *clearConditionalFormat*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.clearConditionalFormat("A1:B3");|

| Set the conditional formatting rule | **Method:** *XLFormat.setCFRule*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLFormat.setCFRule({ action: "lessthan", inputs: ["30"], color: "yellowft", range: "H3:H7" }); }| **Method:** *conditionalFormat*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.conditionalFormat({type:"GreaterThan" cFColor:"RedFT" value:"700", range:"B2:B9"});|

| Set format style for the range of cells | **Method:** *XLFormat.format*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLFormat.format({style:{ "background-color": "#C0C0C0"}}, "A1:C10"); }| **Method:** *cellFormat*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.cellFormat({ fontWeight: 'bold', fontSize: '12pt', backgroundColor: '#279377', color: 'ffffff' }, 'A2:E2');|

| Triggers before formatting the cells | **Event:** *beforeCellFormat*

<ej-spreadsheet id="spreadsheet" (beforeCellFormat)="onBeforeCellFormat(\$event)">
</ej-spreadsheet>
TS
onBeforeCellFormat(args){ }| **Event:** *beforeCellFormat*

<ej-spreadsheet (beforeCellFormat)="onBeforeCellFormat(\$event)">
</ej-spreadsheet>
TS
onBeforeCellFormat(args){ }|

| Specifies the border for the cell | **Property:** *sheets.border*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
Ts
this.sheets = [{ border: [{ type: ej.Spreadsheet.BorderType.AllBorder, color: "#456534", range: "C6:D9" }] }];| **Property:** *sheets.rows.cells.border*

<ej-spreadsheet>
<e-sheets><e-sheet><e-rows><e-row><e-cells><e-cell border='1px solid #456534'></e-cell></e-cells></e-row></e-rows></e-sheet></e-sheets>
</ej-spreadsheet>|

| Set border for the specified range of cells | **Method:** *setBorder* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = $("#spreadsheet").data("ejSpreadsheet"); xlObj.setBorder({ style: "solid", type: "outside", color: "#000000"}, "B2:B6"); }|` **Method:** *setBorder* `

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.setBorder({ border: "1px solid #000000" }, "B2:B6", "Outer");|`

Filtering

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

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

| Enables or disables the filtering feature | **Property:** *allowFiltering* `

<ej-spreadsheet [allowFiltering]="true">
</ej-spreadsheet>|` **Property:** *allowFiltering* `

<ejs-spreadsheet [allowFiltering]="true">
</ejs-spreadsheet>|`

| Clear the filter in the filtered columns | **Method:** *XLFilter.clearFilter* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = $("#spreadsheet").data("ejSpreadsheet"); xlObj.XLFilter.clearFilter(); }|` **Method:** *clearFilter* `

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.clearFilter();|`

| Apply filter for the specified range of cells | **Method:** *XLFilter.filter* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = $("#spreadsheet").data("ejSpreadsheet"); xlObj.XLFilter.filter("A3:C8"); }|` **Method:** *applyFilter* `

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.applyFilter([{ field: 'E', operator: 'equal', value: '20' }], 'A1:H1');|`

Sorting

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

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

| Enables or disables the sorting feature | **Property:** *allowSorting* `

<ej-spreadsheet [allowSorting]="true">
</ej-spreadsheet>|` **Property:** *allowSorting* `

<ejs-spreadsheet [allowSorting]="true">
</ejs-spreadsheet>|`

| Sort a particular range of cells based on its values | **Method:** *XLSort.sortByRange* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = $("#spreadsheet").data("ejSpreadsheet"); xlObj.XLSort.sortByRange("A1:D3", "B", "ascending"); }|` **Method:** *sort* `

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.sort({ sortDescriptors: { order: 'Ascending' }, containsHeader: true}, 'A1:H11');|`

Hyperlink

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

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

| Enables or disables the hyperlink feature | **Property:** *allowHyperlink*

<ej-spreadsheet [allowHyperlink]="true"></ej-spreadsheet> | **Property:** *allowHyperlink*

<ejs-spreadsheet [allowHyperlink]="true"></ejs-spreadsheet> |

| Remove the hyperlink in the specified cells | **Method:** *removeHyperlink*

<ej-spreadsheet id="spreadsheet"></ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.removeHyperlink("A2:A3"); } | **Method:** *removeHyperlink*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.removeHyperlink("A2:A3"); |

| Set the hyperlink in the specified cells | **Method:** *setHyperlink*

<ej-spreadsheet id="spreadsheet"></ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.setHyperlink("A2:A3",{ "cellAddr": "A2:A8"}, 3); } | **Method:** *addHyperlink*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.addHyperlink({ address: 'B2' }, 'A1'); |

Protection

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

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

| Enables or disables the lock cell feature | **Property:** *allowLockCell*

<ej-spreadsheet [allowLockCell]="true"></ej-spreadsheet> | By default, it is enabled. |

| Protect or Unprotect the active sheet | **Method:** *protectSheet*

<ej-spreadsheet id="spreadsheet"></ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.protectSheet(); } | **Method:** *protectSheet*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.protectSheet(0, {}); |

| Lock or Unlock the range of cells | **Method:** *lockCells*

<ej-spreadsheet id="spreadsheet"></ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.lockCells("A3:B5", true); } | **Method:** *lockCells*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.lockCells("A3:B5", true); |

Find and Replace

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

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

| Enables or disables the find & replace feature | **Property:** *allowSearching*

<ej-spreadsheet [allowSearching]="true"></ej-spreadsheet> | **Property:** *allowFindAndReplace*

<ejs-spreadsheet [allowFindAndReplace]="true"></ejs-spreadsheet> |

| Find the next occurrence of the given value | **Method:** *XLSearch.findNext*

<ej-spreadsheet id="spreadsheet"></ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.XLSearch.findNext("g", {isCSen: false,

isEMatch: false, type: "value", mode: "sheet", searchBy: "rows"}, 1); }| **Method:** *find*

<ej-spreadsheet #default>
</ej-spreadsheet>
<TS>
@ViewChild('default')
 public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.find({ value: "Jenna
 Schoolfield", sheetIndex: 1, findOpt: "next", mode: "Sheet", isCSen: false, isEMatch: false,
 searchBy: "By Row" });|

| Find the previous occurrence of the given value | **Method:** *XLSearch.findPrevious*

<ej-
 spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
 \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLSearch.findPrevious("g", {isCSen: true,
 isEMatch: false, type: "value", mode: "sheet", searchBy: "columns"}, 1); }| **Method:** *find*

<ej-spreadsheet #default>
</ej-spreadsheet>
<TS>
@ViewChild('default')
 public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.find({ value: "Jenna
 Schoolfield", sheetIndex: 1, findOpt: "previous", mode: "Sheet", isCSen: false, isEMatch: false,
 searchBy: "By Row" });|

| Find and replace all the data by sheet | **Method:** *XLSearch.replaceAllBySheet*

<ej-
 spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
 \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLSearch.replaceAllBySheet("Sheet",
 "Spreadsheet", true, false); }| **Method:** *replace*

<ej-spreadsheet #default>
</ej-
 spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj:
 SpreadsheetComponent;
this.spreadsheetObj.replace({replaceValue: 'new value', mode:
 'Sheet', replaceBy: 'replaceAll', value: '10'});|

| Find and replace all the data by workbook | **Method:** *XLSearch.replaceAllByBook*

<ej-
 spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
 \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLSearch.replaceAllByBook("Sheet",
 "Spreadsheet", true, false); }| **Method:** *replace*

<ej-spreadsheet #default>
</ej-
 spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj:
 SpreadsheetComponent;
this.spreadsheetObj.replace({replaceValue: 'new value', mode:
 'Workbook', replaceBy: 'replaceAll', value: '10'});|

Ribbon

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

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

| Show or hide the ribbon | **Property:** *showRibbon*

<ej-spreadsheet
 [showRibbon]="true">
</ej-spreadsheet>| **Property:** *showRibbon*

<ej-spreadsheet
 [showRibbon]="true">
</ej-spreadsheet>|

| Add the menu items in the file menu | **Method:** *XL.Ribbon.addMenuItem*

<ej-spreadsheet
 id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj =
 \$("#spreadsheet").data("ejSpreadsheet"); xObj.XL.Ribbon.addMenuItem([{ id: "newitem", text:
 "New Item", parentId: "FILE" }], 2); }| **Method:** *addFileMenuItems*

<ej-spreadsheet
 #default>
</ej-spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj:
 SpreadsheetComponent;
this.spreadsheetObj.addFileMenuItems([{ text: 'New Item' }],
 "Save As");|

| Add the tab in the ribbon | **Method:** *XL.Ribbon.addTab*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xLObj = \$(("#spreadsheet").data("ejSpreadsheet")); let tabGroup = [{ alignType: ej.Ribbon.AlignType.Rows, content: [{ groups: [{ id: "new", text: "New", toolTip: "New", buttonSettings: { contentType: ej.ContentType.ImageOnly, imagePosition: ej.ImagePosition.ImageTop, prefixIcon: "e-icon e-ssr-cut", click: "executeAction" } }], defaults: { type: ej.Ribbon.Type.Button, width: 60, height: 70 } }]}; xLObj.XL.Ribbon.addTab("Tab2", tabGroup, 2); } | **Method:** *addRibbonTabs*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.addRibbonTabs([{ header: { text: 'Custom' }, content: [{ text: 'Custom', tooltipText: 'Custom Btn' }]}], 'Data'); |

| Disable ribbon items | **Method:** *XL.Ribbon.disableRibbonItems*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xLObj = \$(("#spreadsheet").data("ejSpreadsheet")); xLObj.XL.Ribbon.disableRibbonItems(["SpreadsheetRibbonInsertIllustrationsPictures"]); } | **Method:** *enableToolbarItems*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.enableToolbarItems('Home', ['spreadsheet_line-through'], false); |

| Enable ribbon items | **Method:** *XL.Ribbon.enableRibbonItems*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xLObj = \$(("#spreadsheet").data("ejSpreadsheet")); xLObj.XL.Ribbon.enableRibbonItems(["SpreadsheetRibbonInsertIllustrationsPictures"]); } | **Method:** *enableToolbarItems*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.enableToolbarItems('Home', ['spreadsheet_line-through']); |

| Hide the file menu in the ribbon tab | **Method:** *XL.Ribbon.hideMenu*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xLObj = \$(("#spreadsheet").data("ejSpreadsheet")); xLObj.XL.Ribbon.hideMenu(); } | **Method:** *hideFileMenuItems*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.hideFileMenuItems(['File']); |

| Triggers when the file menu item is selected | **Event:** *menuClick*

<ej-spreadsheet id="spreadsheet" (menuClick)="onMenuClick(\$event)">
</ej-spreadsheet>
TS
onMenuClick(args){ } | **Event:** *fileMenuItemSelect*

<ejs-spreadsheet (fileMenuItemSelect)="onFileMenuItemSelect(\$event)">
</ejs-spreadsheet>
TS
onFileMenuItemSelect(args){ }

[Undo and Redo](#)

| Behavior | [API in Essential JS 1](#) | [API in Essential JS 2](#) |

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

| Enables or disables the undo and redo feature | **Property:** *allowUndoRedo*

<ej-spreadsheet [allowUndoRedo]="true">
</ej-spreadsheet> | **Property:** *allowUndoRedo*

<ejs-spreadsheet [allowUndoRedo]="true">
</ejs-spreadsheet>|

| Update the details for custom undo and redo operations. | **Method:** *updateUndoRedoCollection*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.updateUndoRedoCollection({ action: "custom", cell: xObj.getActiveCell(), sheetIndex: 1 }); } | **Method:** *updateUndoRedoCollection*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.updateUndoRedoCollection({ eventArgs: { class: 'customClass', rowIdx: 0, colIdx: 0, action: 'customCSS' } });|

Worksheet

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

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

| Set active sheet index in the workbook | **Property:** *activeSheetIndex*

<ej-spreadsheet [activeSheetIndex]="true">
</ej-spreadsheet> | **Property:** *activeSheetIndex*

<ejs-spreadsheet [activeSheetIndex]="true">
</ejs-spreadsheet>|

| Specifies the rows for a sheet | **Property:** *sheets.rows*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ rows:[{ height:30, index: 1, cells:[{ value: "Item Name" }] }]}] | **Property:** *sheets.rows*

<ejs-spreadsheet>
<e-sheets><e-sheet><e-rows><e-row height=30 index=1><e-cells><e-cell value="Item Name"></e-cell></e-cells></e-row></e-rows></e-sheet></e-sheets>
</ejs-spreadsheet>`|

| Specifies the cells of a row | **Property:** *sheets.rows.cells*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ rows:[{ cells:[{ index: 1, value: "Item Name", style: { "font-weight": "bold", "color": "#FFFFFF", "background-color": "#428bca" }, format: { type: "general" }, hyperlink: { webAddr: "www.google.com" }, isLocked: true } }] }]}] | **Property:** *sheets.rows.cells*

<ejs-spreadsheet>
<e-sheets><e-sheet><e-rows><e-row><e-cells><e-cell index=1 value="Item Name" [style]="{ fontFamily: 'Axettac Demo', verticalAlign: 'middle', textAlign: 'center', fontSize: '18pt', fontWeight: 'bold', color: '#279377', backgroundColor: '#428bca', border: '1px solid #e0e0e0' }" format="General" hyperlink="https://www.google.com/" [isLocked]='true'></e-cell></e-cells></e-row></e-rows></e-sheet></e-sheets>
</ejs-spreadsheet>`|

| Show or hide the grid lines | **Property:** *sheets.showGridlines*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ showGridlines: true }];| **Property:** *sheets.showGridLines*

<ejs-spreadsheet ><e-sheets><e-sheet [showGridLines]="true"></e-sheet></e-sheets>
</ejs-spreadsheet>|

| Show or hide the headings | **Property:** *sheets.showHeadings*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ showHeadings: true }];| **Property:** *sheets.showHeaders*

<ejs-spreadsheet ><e-sheets><e-sheet [showHeaders]="true"></e-sheet></e-sheets>
</ejs-spreadsheet>|

| Specifies the name for the sheet | **Property:** *sheets.sheetName*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ sheetName: "Sheet Name"}]; | **Property:** *sheets.name*

<ejs-spreadsheet ><e-sheets><e-sheet name="Sheet Name"></e-sheet></e-sheets>
</ejs-spreadsheet>|

| Show or hide the pager | **Property:** *showPager*

<ej-spreadsheet [showPager]="true">
</ej-spreadsheet>| **Property:** *showSheetTabs*

<ejs-spreadsheet [showSheetTabs]="true">
</ejs-spreadsheet>|

| Defines the number of rows to be rendered in the sheet | **Property:** *sheets.rowCount*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ rowCount: 21}]; | **Property:** *sheets.rowCount*

<ejs-spreadsheet ><e-sheets><e-sheet [rowCount]="21"></e-sheet></e-sheets>
</ejs-spreadsheet>|

| Defines the number of columns to be rendered in the sheet | **Property:** *sheets.colCount*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ colCount: 25}]; | **Property:** *colCount*

<ejs-spreadsheet>
<e-sheets><e-sheet [colCount]="21"></e-sheet></e-sheets>
</ejs-spreadsheet>|

Open and Save

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

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

| Enables or disables the import feature | **Property:** *allowImport*

<ej-spreadsheet [allowImport]="true">
</ej-spreadsheet>| **Property:** *allowOpen*

<ejs-spreadsheet [allowOpen]="true">
</ejs-spreadsheet>|

| Enables or disables the exporting feature | **Property:** *exportSettings.allowExporting*

<ej-spreadsheet [exportSettings]="exportSettings">
</ej-spreadsheet>
Ts
this.exportSettings = { allowExporting: true }| **Property:** *allowSave*

<ejs-spreadsheet [allowSave]="true">
</ejs-spreadsheet>|

| Defines the excelUrl to export to the excel format | **Property:** *exportSettings.excelUrl*

<ej-spreadsheet [exportSettings]="exportSettings">
</ej-spreadsheet>
Ts
this.exportSettings = { excelUrl: "http://js.syncfusion.com/demos/ejservices/api/Spreadsheet/ExcelExport" }| **Property:** *saveUrl*

<ejs-spreadsheet saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/save'>
</ejs-spreadsheet>|

| Defines the csvUrl to export to the csv format | **Property:** *exportSettings.csvUrl*

<ej-spreadsheet [exportSettings]="exportSettings">
</ej-spreadsheet>
Ts
this.exportSettings = { csvUrl: "http://js.syncfusion.com/demos/ejservices/api/Spreadsheet/CsvExport" }| **Property:** *saveUrl*

You can use the same service url and specify saveType as Csv in the beforeSave event<ejs-spreadsheet saveUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/save'

```
(beforeSave)=‘beforeSave($event)’><br></ej-spreadsheet><br>Ts<br>beforeSave(args:
BeforeSaveEventArgs) { args.saveType = ‘Csv’ }|
```

```
| Import mapper to perform the import feature | Property: importSettings.importMapper <br><br><ej-
spreadsheet [importSettings]=‘importSettings’><br></ej-
spreadsheet><br>Ts<br>this.importSettings= { importMapper:
“http://js.syncfusion.com/demos/ejservices/api/Spreadsheet/Import” };| Property: openUrl
<br><br><ej-spreadsheet
openUrl=‘https://services.syncfusion.com/angular/production/api/spreadsheet/open’><br></ej-
s-spreadsheet>|
```

```
| Import excel file | Method: import <br><br><ej-spreadsheet id=“spreadsheet”><br></ej-
spreadsheet><br>Ts<br>ngAfterViewInit(){ let xlObj = $(“#spreadsheet”).data(“ejSpreadsheet”);
xlObj.import({ file: file }); }| Method: open <br><br><ej-spreadsheet #default><br></ej-s-
spreadsheet><br>Ts<br>@ViewChild(‘default’) public spreadsheetObj:
SpreadsheetComponent;<br>this.spreadsheetObj.open({ file: file });|
```

```
| Load JSON data of the Spreadsheet | Method: loadFromJSON <br><br><ej-spreadsheet
id=“spreadsheet”><br></ej-spreadsheet><br>Ts<br>ngAfterViewInit(){ let xlObj =
$(“#spreadsheet”).data(“ejSpreadsheet”); let response = xlObj.saveAsJSON();
xlObj.loadFromJSON(response); }| Method: openFromJson <br><br><ej-spreadsheet
#default><br></ej-spreadsheet><br>Ts<br>@ViewChild(‘default’) public spreadsheetObj:
SpreadsheetComponent;<br>this.spreadsheetObj.saveAsJson().then(response =>
(this.spreadsheetObj.openFromJson({ file: response.jsonObject })));|
```

```
| Triggers when a file is imported | Event: onImport <br><br><ej-
spreadsheet id=“spreadsheet” (onImport)=“onImport($event)”><br></ej-
spreadsheet><br>Ts<br>onImport(args){ }| Event: openComplete <br><br><ej-s-
spreadsheet (openComplete)=“onOpenComplete($event)”><br></ej-s-
spreadsheet><br>Ts<br>onOpenComplete(args){ }
```

```
| Triggers when the opened Excel file fails to load | Event: openFailure <br><br><ej-
spreadsheet id=“spreadsheet” (openFailure)=“onOpenFailure($event)”><br></ej-
spreadsheet><br>Ts<br>onOpenFailure(args){ }| Event: openFailure <br><br><ej-s-
spreadsheet (openFailure)=“onOpenFailure($event)”><br></ej-s-
spreadsheet><br>Ts<br>onOpenFailure(args){ }
```

```
| Save the sheet data as Excel or CSV document | Method: XLExport.export <br><br><ej-spreadsheet
id=“spreadsheet”><br></ej-spreadsheet><br>Ts<br>ngAfterViewInit(){ let xlObj =
$(“#spreadsheet”).data(“ejSpreadsheet”); xlObj.XLExport.export(“Excel”); }| Method: save
<br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>Ts<br>@ViewChild(‘default’)
public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.save({ saveType: ‘Xlsx’
});|
```

```
| Save the sheet data as Excel or CSV document | Method: XLExport.export <br><br><ej-spreadsheet
id=“spreadsheet”><br></ej-spreadsheet><br>Ts<br>ngAfterViewInit(){ let xlObj =
$(“#spreadsheet”).data(“ejSpreadsheet”); xlObj.XLExport.export(“Excel”); }| Method: save
<br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>Ts<br>@ViewChild(‘default’)
```

```
public spreadsheetObj: SpreadsheetComponent; <br> this.spreadsheetObj.save({ saveType: 'Xlsx'
});|
```

| Save JSON data of the Spreadsheet | **Method:** *saveAsJSON*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); let response = xObj.saveAsJSON();
xObj.loadFromJSON(response); }| **Method:** *saveAsJson*

<ejs-spreadsheet
#default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj:
SpreadsheetComponent;
 this.spreadsheetObj.saveAsJson().then(response =>
(this.spreadsheetObj.openFromJson({ file: response.jsonObject })));|

Data Binding

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

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

| Specifies the single range or multiple range settings for a sheet | **Property:** *sheets.rangeSettings*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{
rangeSettings: [{ dataSource: defaultData, showHeader: true, startCell: "A1", query:
ej.Query().take(50) }] }]| **Property:** *sheets.ranges*

<ejs-spreadsheet>
<e-sheets><e-
sheet><e-ranges><e-range [dataSource]="defaultData" startCell="A1"
[showFieldAsHeader]="true" [query]="query"></e-range></e-ranges></e-sheet></e-
sheets>
</ejs-spreadsheet>`|

Context Menu

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

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

| Enables or disables the context menu | **Property:** *enableContextMenu*

<ej-spreadsheet
[enableContextMenu]="true">
</ej-spreadsheet>| **Property:** *enableContextMenu*

<ejs-spreadsheet [enableContextMenu]="true">
</ejs-spreadsheet>|

| Dynamically add items in the context menu | **Method:** *XLCMenu.addItem*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet");
xObj.XLCMenu.addItem(ej.Spreadsheet.ContextMenu.Cell, [{"text": "Added item 1!!!",
"url": "#", "id": "Added item1", "spriteCssClass": "e-icon e-ss-cut" }], 'insertbefore'); }| **Method:**
addContextMenuItems

<ejs-spreadsheet
(contextMenuBeforeOpen)="contextMenuBeforeOpen(\$event)" #default>
</ejs-
spreadsheet>
TS
@ViewChild('default') public spreadsheetObj:
SpreadsheetComponent;
 contextMenuBeforeOpen(args) {
this.spreadsheetObj.addContextMenuItems([{ text: 'Custom Item' }], 'Paste Special', false); }|

| Disable the items in the context menu | **Method:** *XLCMenu.disableItem*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet");
xObj.XLCMenu.disableItem(ej.Spreadsheet.ContextMenu.Cell, [1,2,3]); }| **Method:**
enableContextMenuItems

<ejs-spreadsheet


```
(contextMenuBeforeOpen)="contextMenuBeforeOpen($event)" #default<br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>contextMenuBeforeOpen(args) { this.spreadsheetObj.enableContextMenuItems(['Copy'], false); }|
```

| Enable the items in the context menu | **Method:** *XLCMenu.enableItem*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLCMenu.enableItem(ej.Spreadsheet.ContextMenu.Cell, [1,2,3]); }| **Method:** *enableContextMenuItems*

<ejs-spreadsheet (contextMenuBeforeOpen)="contextMenuBeforeOpen(\$event)" #default
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
contextMenuBeforeOpen(args) { this.spreadsheetObj.enableContextMenuItems(['Copy'], true); }|

| Remove the items in the context menu | **Method:** *XLCMenu.removeItem*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLCMenu.removeItem(ej.Spreadsheet.ContextMenu.Cell, [1,2,3]); }| **Method:** *removeContextMenuItems*

<ejs-spreadsheet (contextMenuBeforeOpen)="contextMenuBeforeOpen(\$event)" #default
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
contextMenuBeforeOpen(args) { this.spreadsheetObj.removeContextMenuItems(['Copy']); }|

| Triggers before the context menu is opened | **Event:** *beforeOpen*

<ej-spreadsheet id="spreadsheet" (beforeOpen)="onBeforeOpen(\$event)">
</ej-spreadsheet>
TS
onBeforeOpen(args){ }| **Event:** *contextMenuBeforeOpen*

<ejs-spreadsheet (contextMenuBeforeOpen)="onContextMenuBeforeOpen(\$event)">
</ej-spreadsheet>
TS
onContextMenuBeforeOpen(args){ }|

| Triggers when the context menu item is selected | **Event:** *contextMenuClick*

<ej-spreadsheet id="spreadsheet" (contextMenuClick)="onContextMenuClick(\$event)">
</ej-spreadsheet>
TS
onContextMenuClick(args){ }| **Event:** *contextMenuItemSelect*

<ejs-spreadsheet (contextMenuItemSelect)="onContextMenuItemSelect(\$event)">
</ej-spreadsheet>
TS
onContextMenuItemSelect(args){ }|

Cell Template

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

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

| Enables or disables the cell type feature | **Property:** *allowCellType*

<ej-spreadsheet [allowCellType]="true">
</ej-spreadsheet>| By default, it is enabled. |

| Specifies the cell types for a cell or range | **Property:** *sheets.cellTypes*

<ej-spreadsheet [sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ cellTypes: [{ range: 'F5', settings: { type: ej.Spreadsheet.CustomCellType.Button, background-color: 'yellow', color: 'black', text: 'BUTTON' } } }];| **Property:** *sheets.ranges.template*

<ejs-spreadsheet>
<e-

```

sheets><e-sheet><e-ranges><e-range template="<button class='e-button-
template'>BUTTON</button>" address="F5"></e-range></e-ranges></e-sheet></e-
sheets><br></ejs-spreadsheet>|

```

Merge

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

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

| Enables or disables the merge feature | **Property:** *allowMerging*

<ej-spreadsheet
[allowMerging]="true">
</ej-spreadsheet> | **Property:** *allowMerge*

<ejs-spreadsheet
[allowMerge]="true">
</ejs-spreadsheet>|

| Merge cells across | **Method:** *mergeAcrossCells*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); xObj.mergeAcrossCells("A3:B5"); }| **Method:** *merge*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.merge("A3:B5",
"Horizontally");|

| Merge the specified ranges | **Property:** *sheets.mergeCells*

<ej-spreadsheet
[sheets]="sheets">
</ej-spreadsheet>
TS
this.sheets = [{ mergeCells:["A1:A2"]}]|
Property: *sheets.rows.cells.rowSpan* & *sheets.rows.cells.colSpan*

<ejs-spreadsheet>
<e-
sheets><e-sheet><e-rows><e-row><e-cells><e-cell [rowSpan]=2 [colSpan]=2></e-cell></e-
cells></e-row></e-rows></e-sheet></e-sheets>
</ejs-spreadsheet>|

| Method to merge the specified ranges | **Method:** *mergeCells*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =
\$("#spreadsheet").data("ejSpreadsheet"); xObj.mergeCells("A3:B5"); }| **Method:** *merge*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.merge("A3:B5",
"All");|

Insert and Delete

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

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

| Enables or disables the delete feature | **Property:** *allowDelete*

<ej-spreadsheet
[allowDelete]="true">
</ej-spreadsheet> | **Property:** *allowDelete*

<ejs-spreadsheet
[allowDelete]="true">
</ejs-spreadsheet>|

| Enables or disables the insert feature | **Property:** *allowInsert*

<ej-spreadsheet
[allowInsert]="true">
</ej-spreadsheet> | **Property:** *allowInsert*

<ejs-spreadsheet
[allowInsert]="true">
</ejs-spreadsheet>|

| Add a new sheet | **Method:** *addNewSheet*

<ej-spreadsheet id="spreadsheet">
</ej-
spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet");
xObj.addNewSheet(); }| **Method:** *insertSheet*

<ejs-spreadsheet #default>
</ejs-
spreadsheet>
TS
@ViewChild('default') public spreadsheetObj:
SpreadsheetComponent;
this.spreadsheetObj.insertSheet();|

| Insert a column | **Method:** *insertEntireColumn* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.insertEntireColumn(1, 2); }| Method: insertColumn

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.insertColumn(1, 2);|`

| Insert a row | **Method:** *insertEntireRow* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.insertEntireRow(1, 2); }| Method: insertRow

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.insertRow(1, 2);|`

| Insert a sheet | **Method:** *insertSheet* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.insertSheet(); }| Method: insertSheet

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.insertSheet();|`

| Delete the entire column | **Method:** *deleteEntireColumn* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.deleteEntireColumn(2, 3); }| Method: delete

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.delete(2, 3, 'Column');|`

| Delete the entire row | **Method:** *deleteEntireRow* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.deleteEntireRow(2, 3); }| Method: delete

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.delete(2, 3, 'Row');|`

| Delete a sheet | **Method:** *deleteSheet* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.deleteSheet(2); }| Method: delete

<ejs-spreadsheet #default>
</ejs-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.delete(2, null, 'Sheet');|`

Clear

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

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

| Enables or disables the clear feature | **Property:** *allowClear* `

<ej-spreadsheet [allowClear]="true">
</ej-spreadsheet>|` By default, it is enabled. |

| Clear all the data and format in the specified range of cells | **Method:** *clearAll* `

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.clearAll("A2:A6"); }| Method: clear`

```
<br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.clear({ type: "Clear
All", range: "A2:A6" });|
```

| Clear all the format in the specified range of cells | **Method:** *clearAllFormat*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.clearAllFormat("A2:A6"); }| **Method:** *clear*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.clear({ type: "Clear Formats", range: "A2:A6" });|

| Clear the contents in the specified range of cells | **Method:** *clearContents*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.clearContents("A2:A6"); }| **Method:** *clear*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.clear({ type: "Clear Contents", range: "A2:A6" });|

Data Validation

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

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

| Enables or disables the data validation feature | **Property:** *allowDataValidation*

<ej-spreadsheet [allowDataValidation]="true">
</ej-spreadsheet>| **Property:** *allowDataValidation*

<ej-spreadsheet [allowDataValidation]="true">
</ej-spreadsheet>|

| Apply data validation rules in a selected range of cells based on the defined condition | **Method:** *XLValidate.applyDVRules*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.XLValidate.applyDVRules("A1:D3", ["Between", "15", "20"], "number", true, true); }| **Method:** *addDataValidation*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.addDataValidation({ type: 'TextLength', operator: 'LessThanOrEqualTo', value1: '4' }, 'A2:A5');|

| Clear the applied validation rules in a specified range of cells | **Method:** *XLValidate.clearDV*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.XLValidate.clearDV("A2:A7"); }| **Method:** *removeDataValidation*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.removeDataValidation("A2:A5");|

| Clear invalid data highlights in the given range | **Method:** *XLValidate.clearHighlightedValData*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj = \$(("#spreadsheet").data("ejSpreadsheet")); xlObj.XLValidate.clearHighlightedValData("A2:A7"); }| **Method:** *removeInvalidHighlight*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default')

public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.removeInvalidHighlight("A1:H5"); |

| Highlight invalid data in a specified range of cells | **Method:** *XLValidate.highlightInvalidData*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.XLValidate.highlightInvalidData("A2:A7"); } | **Method:** *addInvalidHighlight*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.addInvalidHighlight('A1:H5');

Wrap

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

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

| Enables or disables the wrap text feature | **Property:** *allowWrap*

<ej-spreadsheet [allowWrap]="true">
</ej-spreadsheet> | **Property:** *allowWrap*

<ejs-spreadsheet [allowWrap]="true">
</ejs-spreadsheet> |

| Unwrap the specified range of cells | **Method:** *unWrapText*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.unWrapText("A1:B3"); } | **Method:** *wrap*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.wrap("A1:B3", false); |

| Wrap the specified range of cells | **Method:** *wrapText*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
<TS>
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.wrapText("A1:B3"); } | **Method:** *wrap*

<ejs-spreadsheet #default>
</ejs-spreadsheet>
<TS>
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.wrap("A1:B3"); |

Scrolling

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

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

| Enables or disables the scrolling feature | **Property:** *scrollSettings.allowScrolling*

<ej-spreadsheet [scrollSettings]="scrollSettings">
</ej-spreadsheet>
<Ts>
this.scrollSettings = { allowScrolling: true }; | **Property:** *allowScrolling*

<ejs-spreadsheet [allowScrolling]="true">
</ejs-spreadsheet> |

| Enables or disables the sheet on demand | **Property:** *scrollSettings.allowSheetOnDemand*

<ej-spreadsheet [scrollSettings]="scrollSettings">
</ej-spreadsheet>
<Ts>
this.scrollSettings = { allowSheetOnDemand: true }; | By default, each sheet will be rendered on demand |

| Enables or disables the virtual scrolling feature | **Property:** *scrollSettings.allowVirtualScrolling*

<ej-spreadsheet [scrollSettings]="scrollSettings">
</ej-spreadsheet>
<Ts>
this.scrollSettings = { allowVirtualScrolling: true }; | **Property:** *scrollSettings.enableVirtualization*

<ejs-spreadsheet

```
[scrollSettings]="scrollSettings"><br></ej-spreadsheet><br>Ts<br>this.scrollSettings = {
enableVirtualization: true };|
```

| Set the scroll mode to finite | **Property:** *scrollSettings.scrollMode*

<ej-spreadsheet
[scrollSettings]="scrollSettings">
</ej-spreadsheet>
Ts
this.scrollSettings = {
scrollMode: ej.Spreadsheet.scrollMode.Normal };| **Property:** *scrollSettings.isFinite*

<ej-spreadsheet
[scrollSettings]="scrollSettings">
</ej-spreadsheet>
Ts
this.scrollSettings = { isFinite: true };|

| Perform goto operation | **Method:** *XLScroll.goTo*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj =
\$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLScroll.goTo("A30"); }| **Method:** *goTo*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.goTo('A30');|

| Scroll the sheet content to the specified cell address | **Method:** *XLScroll.scrollToCell*

<ej-spreadsheet
id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xlObj =
\$("#spreadsheet").data("ejSpreadsheet"); xlObj.XLScroll.scrollToCell("A30"); }| **Method:** *goTo*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default')
public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.goTo('A30');|

Comparision between EJ1 and EJ2 Spreadsheet features

The following table compares Excel functionality with the availability of EJ1 and EJ2 Spreadsheet features.

| Features | Available in EJ1 Spreadsheet | Available in EJ2 Spreadsheet | Comments |
|----------------------------------|------------------------------|------------------------------|---|
| --- | --- | --- | --- |
| Ribbon | Yes | Yes | - |
| Formula bar | Yes | Yes | - |
| Sheet tab | Yes | Yes | - |
| Show / Hide gridlines and header | Yes | Yes | - |
| Scrolling | Partially | Yes | - |
| Selection | Yes | Yes | - |
| Editing | Yes | Yes | - |
| Formulae | Yes | Partially | EJ2 supports limited number of most used formulas |
| Named range | Yes | Partially | EJ2 Spreadsheet Named range supports only in workbook scope |
| Data Binding | Yes | Yes | - |
| Formatting | Yes | Yes | - |
| Context menu | Yes | Yes | - |
| Keyboard navigation | Yes | Yes | - |
| Keyboard shortcuts | Yes | Yes | - |
| Sorting | Yes | Yes | - |

| Filtering | Yes | Yes | - |

| Hyperlink | Yes | Yes | - |

| Undo & redo | Yes | Yes | - |

| Open and Save | Yes | Yes | - |

| Resize / Autofit | Yes | Yes | - |

| Clipboard | Yes | Yes | - |

| Collaborative editing | No | Yes | - |

| Wrap text | Yes | Yes | - |

| Template | No | Yes | - |

| Merge cells | Yes | Yes | - |

| Show / Hide rows and columns | Yes | Yes | - |

| Sheet customizations | Yes | Partially | Move or copy sheet is not supported in EJ2 spreadsheet. |

| Data Validation | Yes | Yes | - |

| Table | Yes | No | - |

| Chart | Yes | Yes | - |

| Image | Yes | Yes | - |

| Conditional formatting | Yes | Yes | - |

| Freeze Pane | Yes | Yes | - |

| Scaling | No | No | - |

| Print | Yes | No | - |

| Grouping | No | No | - |

| Autofill | Yes | No | - |

| Auto Sum | Yes | Yes | - |

| Format painter | Yes | No | - |

| Cell Style | Yes | Partially | We can only customize the cell style in EJ2 Spreadsheet through API. |

| Protection | Yes | Partially | Custom encryption is not supported in EJ2 Spreadsheet's protect workbook. |

| Find and replace | Yes | Yes | - |

| Drag and Drop | Yes | No | - |

| Notes | Yes | No | - |

| Comments | No | No | - |

| Pivot table | Yes | No | - |

| Sparklines | Yes | No | - |

| Form controls | Yes | No | - |

| Shapes | No | No | - |

| Clear | Yes | Yes | - |

Common Properties

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

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

| Enables or disables the keyboard navigation feature | **Property:** *allowKeyboardNavigation*
`

<ej-spreadsheet [allowKeyboardNavigation]="true">
</ej-spreadsheet>` | **Property:**
enableKeyboardNavigation `

<ejs-spreadsheet`
`[enableKeyboardNavigation]="true">
</ejs-spreadsheet>` |

| Enables or disables the resizing feature | **Property:** *allowResizing* `

<ej-spreadsheet`
`[allowResizing]="true">
</ej-spreadsheet>` | **Property:** *allowResizing* `

<ejs-spreadsheet`
`[allowResizing]="true">
</ejs-spreadsheet>` |

| Add the CSS class to the root element to customize the appearance | **Property:** *cssClass* `

<ej-`
`spreadsheet cssClass="custom-class">
</ej-spreadsheet>` | **Property:** *cssClass* `

<ejs-`
`spreadsheet cssClass="custom-class">
</ejs-spreadsheet>` |

| Enables or disables the touch support | **Property:** *enableTouch* `

<ej-spreadsheet`
`[enableTouch]="true">
</ej-spreadsheet>` | By default, it is enabled. |

| Overrides the global culture and localization | **Property:** *locale* `

<ej-spreadsheet locale="en-`
`ES">
</ej-spreadsheet>` | **Property:** *locale* `

<ejs-spreadsheet locale="en-`
`US">
</ejs-spreadsheet>` |

| Enables or disables the picture feature | **Property:** *pictureSettings.allowPictures* `

<ej-`
`spreadsheet [pictureSettings]="pictureSettings">
</ej-`
`spreadsheet>
``Ts``
``this.pictureSettings = { allowPictures: true }` | **Property:** *allowImage*
`

<ejs-spreadsheet [allowImage]="true">
</ejs-spreadsheet>` |

| Set the height of the Spreadsheet | **Property:** *scrollSettings.height* `

<ej-spreadsheet`
`[scrollSettings]="scrollSettings">
</ej-spreadsheet>
``Ts``
``this.scrollSettings = { height:`
`600 };` | **Property:** *height* `

<ejs-spreadsheet height=600>
</ejs-spreadsheet>` |

| Set the width of the Spreadsheet | **Property:** *scrollSettings.width* `

<ej-spreadsheet`
`[scrollSettings]="scrollSettings">
</ej-spreadsheet>
``Ts``
``this.scrollSettings = { width:`
`1300 };` | **Property:** *width* `

<ejs-spreadsheet width=1300>
</ejs-spreadsheet>` |

| Hide the specified columns | **Property:** *sheets.hideColumns* `

<ej-spreadsheet`
`[sheets]="sheets">
</ej-spreadsheet>
``Ts``
``this.sheets = [{ hideColumns: [3] }]` |
Property: *sheets.columns.hidden* `

<ejs-spreadsheet>
<e-sheets><e-sheet><e-`
`columns><e-column index=3 [hidden]="true"></e-column></e-columns></e-sheet></e-`
`sheets>
</ejs-spreadsheet>` |

| Hide the specified rows | **Property:** *sheets.hideRows* `

<ej-spreadsheet`
`[sheets]="sheets">
</ej-spreadsheet>
``Ts``
``this.sheets = [{ hideRows: [3] }]` | **Property:**


```
sheets.rows.hidden <br><br><ej-spreadsheet><br><e-sheets><e-sheet><e-rows><e-row index=3
[hidden]="true"></e-row></e-rows></e-sheet></e-sheets><br></ej-spreadsheet>`|
```

Common Methods

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

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

| Get the data in the specified range | **Method:** *getRangeData*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.getRangeData({range: [2, 6, 2, 6], property: ["value", "value2", "format"], sheetIdx: 1}); }| **Method:** *getData*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.getData(getRangeAddress('A1:D5')).then((cell s)=>{ cells.forEach((cell, key)=>{ }) });|

| Get the range indices array based on the specified alpha range | **Method:** *getRangeIndices*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.getRangeIndices("A1:A9"); }| **Method:** *getRangeIndexes*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
import { getRangeIndexes } from '@syncfusion/ej2-spreadsheet';
getRangeIndexes("A1:A9")|

| Send a paging request to the specified sheet Index | **Method:** *gotoPage*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.gotoPage(1); }| **Method:** *goTo*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.goTo('Sheet2!A1');|

| Hide the specified columns | **Method:** *hideColumn*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.hideColumn(1, 4); }| **Method:** *hideColumn*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.hideColumn(1, 4);|

| Hide the specified rows | **Method:** *hideRow*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.hideRow(1, 4); }| **Method:** *hideRow*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.hideRow(1, 4);|

| Refresh the Spreadsheet | **Method:** *refreshSpreadsheet*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = \$("#spreadsheet").data("ejSpreadsheet"); xObj.refreshSpreadsheet(); }| **Method:** *refresh*

<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.refresh(0, {});|

| Set the height for the rows | **Method:** *setHeightToRows*

<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj =

```
$(("#spreadsheet").data("ejSpreadsheet")); xObj.setHeightToRows([{rowIndex: 2, height: 40}]);
}| Method: setRowHeight <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.setRowHeight(40, 2);|
```

```
| Set the width for the columns | Method: setWidthToColumns <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.setWidthToColumns([{colIndex: 2, width: 40}]); }| Method: setColWidth <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.setColWidth(40, 2);|
```

```
| Show the hidden columns within the specified range | Method: showColumn <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.showColumn(3, 6); }| Method: hideColumn <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.hideColumn(3, 6, false);|
```

```
| Show the hidden rows in the specified range | Method: showRow <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.showRow(3, 6); }| Method: hideRow <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.hideRow(3, 6, false);|
```

```
| Show waiting pop-up in the Spreadsheet | Method: showWaitingPopUp <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.showWaitingPopUp(); }| Method: showSpinner <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.showSpinner();|
```

```
| Hide displayed waiting pop-up in Spreadsheet | Method: hideWaitingPopUp <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.hideWaitingPopUp(); }| Method: hideSpinner <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.hideSpinner();|
```

```
| Fit the height of rows | Method: XLResize.fitHeight <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.XLResize.fitHeight([2,3,4,5]); }| Method: autoFit <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.autoFit('1:4');|
```

```
| Fit the width of columns | Method: XLResize.fitWidth <br><br><ej-spreadsheet id="spreadsheet"><br></ej-spreadsheet><br>TS<br>ngAfterViewInit(){ let xObj = $(("#spreadsheet").data("ejSpreadsheet")); xObj.XLResize.fitWidth([2,3,4,5]); }| Method: autoFit <br><br><ej-spreadsheet #default><br></ej-spreadsheet><br>TS<br>@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;<br>this.spreadsheetObj.autoFit('A:D');|
```


| Set the column width of the specified column index | **Method:** *XLResize.setColWidth*
`<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.XLResize.setColWidth(2, 100); }| Method: setColWidth
<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.setColWidth(100, 2);|`

| Set the row height of the specified row index | **Method:** *XLResize.setRowHeight*
`<ej-spreadsheet id="spreadsheet">
</ej-spreadsheet>
TS
ngAfterViewInit(){ let xObj = $("#spreadsheet").data("ejSpreadsheet"); xObj.XLResize.setRowHeight(2, 100); }| Method: setRowHeight
<ej-spreadsheet #default>
</ej-spreadsheet>
TS
@ViewChild('default') public spreadsheetObj: SpreadsheetComponent;
this.spreadsheetObj.setRowHeight(100, 2);|`

Common Events

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

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

| Triggers for every action before it starts | **Event:** *actionBegin*
`<ej-spreadsheet id="spreadsheet" (actionBegin)="onActionBegin($event)">
</ej-spreadsheet>
TS
onActionBegin(args){ }| Event: actionBegin
<ej-spreadsheet (actionBegin)="onActionBegin($event)">
</ej-spreadsheet>
TS
onActionBegin(args){ }|`

| Triggers for every completed action | **Event:** *actionComplete*
`<ej-spreadsheet id="spreadsheet" (actionComplete)="onActionComplete($event)">
</ej-spreadsheet>
TS
onActionComplete(args){ }| Event: actionComplete
<ej-spreadsheet (actionComplete)="onActionComplete($event)">
</ej-spreadsheet>
TS
onActionComplete(args){ }|`

| Triggers after the sheet is loaded | **Event:** *loadComplete*
`<ej-spreadsheet id="spreadsheet" (loadComplete)="onLoadComplete($event)">
</ej-spreadsheet>
TS
onLoadComplete(args){ }| Event: created
<ej-spreadsheet (created)="onCreated()">
</ej-spreadsheet>
TS
onCreated(){ }|`

How To

Sort a range by custom list in Angular Spreadsheet component

You can also define the sorting of cell values based on your own customized personal list. In this article, custom list is achieved using **custom sort comparer**.

In the following demo, the **Trustworthiness** column is sorted based on the custom lists **Perfect**, **Sufficient**, and **Insufficient**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { tradeData } from './datasource';
```

```

import { DataManager, DataUtil } from '@syncfusion/ej2-data';
import { SpreadsheetComponent, CellModel } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet (dataBound)='dataBound()' (sortComplete)='sortComplete($event)'"> <e-sheets> <e-sheet> <e-ranges> <e-range [dataSource]='tradeData'></e-range></e-ranges><e-columns><e-column [width]=100></e-column><e-column [width]=120></e-column><e-column [width]=96></e-column></e-columns></e-sheet></e-sheets></ejs-spreadsheet>"
})
export class AppComponent implements OnInit {
  public tradeData?: object[];
  @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
  ngOnInit(): void {
    this.tradeData = tradeData;
  }
  dataBound(){
    if (this.spreadsheetObj!.activeSheetIndex === 0) {
      this.spreadsheetObj!.sort({sortDescriptors: { field: 'F',
sortComparer: this.mySortComparer }, containsHeader: true}, 'A1:H20');
    }
  };
  sortComplete (args : any) {
    this.spreadsheetObj!.selectRange(args.range);
    // code here.
  }
  mySortComparer(x: CellModel, y: CellModel): number {
    // custom sort comparer to sort based on the custom list.
    let customList: string[] = ['Perfect', 'Sufficient', 'Insufficient'];
    let comparer: Function = DataUtil.fnSort('Ascending');
    return comparer(x ? customList.indexOf((x as any).value) : x, y ?
customList.indexOf((y as any).value) : y);
  };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

Print in Angular Spreadsheet component

You can use the `print` method by importing from `ej2-base` package. Here, the `Select` event in the dropdown and the `dataBound` event are used to print the single/multiple sheets of data. To print the single/multiple sheets, use the dropdown button and select the `Print` (or) `Print All` option. In the following sample, you can be able to print the single/multiple sheets.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { dataSource1, dataSource2, printElement, isPrint } from
'./datasource';
import { SpreadsheetComponent, CellModel, UsedRangeModel, SheetModel } from
'@syncfusion/ej2-angular-spreadsheet';
import { ItemModel, MenuEventArgs } from '@syncfusion/ej2-angular-
splitbuttons';
import { getComponent, print } from '@syncfusion/ej2-base';
@Component({
  imports: [

    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<button ej2-dropdownbutton [items]='items' content='Print'
(select)='itemSelect($event)'></button>
<ej2-spreadsheet #spreadsheet id="spreadsheet" (created)="created()"
(dataBound)="dataBound()">
  <e-sheets>
    <e-sheet name="Budget">
      <e-ranges>
        <e-range [dataSource]="budgetData"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
    <e-sheet name="Salary">
      <e-ranges>
        <e-range [dataSource]="salaryData"></e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
`
})
```

```

        </ejs-spreadsheet>`
    })
    export class AppComponent implements OnInit {
        ngOnInit(): void {
            throw new Error('Method not implemented.');
        }
        @ViewChild('spreadsheet') public spreadsheetObj?: SpreadsheetComponent;
        budgetData: object[] = dataSource1;
        salaryData: object[] = dataSource2;
        public items: ItemModel[] = [
            {
                text: "Print"
            },
            {
                text: "Print All"
            }
        ];
        public itemSelect(args: MenuEventArgs) {
            let spreadsheet = getComponent((document as
any).getElementById("spreadsheet"), "spreadsheet");
            if (args.item.text === 'Print') {
                printElement.querySelector(".e-sheet-content")!.innerHTML =
document.querySelector(
                ".e-sheet-content"
            )!.outerHTML; // To add the spreadsheet table
                let usedRange: UsedRangeModel = (spreadsheet as
any).getActiveSheet().usedRange;
                let tbody: Element = printElement.querySelector('tbody') as Element;
                for (let i: number = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
                    if (tbody.getElementsByClassName('e-row')[i] && parseInt((tbody as
any).getElementsByClassName('e-row')[i].getAttribute('aria-rowindex')) >
usedRange.rowIndex! + 1) {
                        tbody.getElementsByClassName('e-row')[i].remove();
                    }
                }
                (printElement.querySelector('.e-sheet-
content')!.children[0].getElementsByClassName('e-virtualtrack')[0] as
HTMLElement).style.height = 'auto';
                print(printElement);
                printElement.querySelector(".e-sheet-content")!.innerHTML = '';
            }
            if (args.item.text === 'Print All') {
                let sheets: SheetModel[] = (spreadsheet as any).sheets;
                if ((spreadsheet as any).activeSheetIndex === 0) {
                    printElement.querySelector(".e-sheet-content")!.innerHTML =
document.querySelector(
                        ".e-sheet-content"
                    )!.outerHTML; // To add the spreadsheet table
                    let usedRange: UsedRangeModel = (spreadsheet as
any).getActiveSheet().usedRange;
                    let tbody: Element = printElement.querySelector('tbody') as Element;
                    for (let i: number = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
                        if (tbody.getElementsByClassName('e-row')[i] &&
parseInt((tbody.getElementsByClassName('e-row')[i] as
any).getAttribute('aria-rowindex')) > usedRange.rowIndex! + 1) {
                            tbody.getElementsByClassName('e-row')[i].remove();
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
  }
  if (sheets[(spreadsheet as any).activeSheetIndex + 1]) {
    (spreadsheet as any).goTo(sheets[(spreadsheet as
any).activeSheetIndex + 1].name + "!A1");
  } else {
    print(printElement);
    printElement.querySelector(".e-sheet-content")!.innerHTML = '';
  }
  } else {
    if (sheets[0]) {
      (spreadsheet as any).goTo(sheets[0].name + "!A1");
    }
  }
}
}
created() {
  this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'A1:D1');
  this.spreadsheetObj!.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
  this.spreadsheetObj!.cellFormat({ fontWeight: 'bold', textAlign:
'center' }, 'Salary!A1:D1');
  this.spreadsheetObj!.cellFormat({ fontWeight: 'bold'},
'Salary!A11:D11');
}
dataBound() {
  if (isPrint) {
    let spreadsheet = getComponent(document.getElementById("spreadsheet")
as any, "spreadsheet");
    printElement.querySelector(
      '.e-sheet-content'
    )!.innerHTML += document.querySelector('.e-sheet-content')!.outerHTML;
    let usedRange: UsedRangeModel = (spreadsheet as any).getActiveSheet()
      .usedRange;
    let tbody: Element = printElement
      .querySelector('.e-sheet-content')!
      .children[(spreadsheet as
any).activeSheetIndex].querySelector('tbody') as Element;
    for (
      let i: number = tbody.getElementsByClassName('e-row').length;
      i >= 0;
      i--
    ) {
      if (
        tbody.getElementsByClassName('e-row')[i] &&
        parseInt(
          tbody
            .getElementsByClassName('e-row')
              [i].getAttribute('aria-rowindex') as any
        ) >
          usedRange.rowIndex! + 1
      ) {
        tbody.getElementsByClassName('e-row')[i].remove();
      }
    }
    let sheets: SheetModel[] = (spreadsheet as any).sheets;
    if (sheets.length - 1 === (spreadsheet as any).activeSheetIndex) {

```

```

        let count: number = printElement.querySelector('.e-sheet-content')!
            .childElementCount;
        if (count > 1) {
            for (let i: number = 0; i < count; i++) {
                (printElement
                    .querySelector('.e-sheet-content')!
                    .children[i].getElementsByClassName(
                        'e-virtualtrack'
                    )[0] as HTMLElement).style.height = 'auto';
                printElement
                    .querySelector('.e-sheet-content')!
                    .children[i].setAttribute('style', 'page-break-after:
always;');
            }
        }
        print(printElement);
        printElement.querySelector('.e-sheet-content')!.innerHTML = '';
    } else {
        if (sheets[(spreadsheet as any).activeSheetIndex + 1]) {
            (spreadsheet as any).goTo(
                sheets[(spreadsheet as any).activeSheetIndex + 1].name + '!A1'
            );
        }
    }
}
}
}
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Changing the active sheet while importing a file in Angular Spreadsheet component

You can change the active sheet of imported file by updating [activeSheetIndex](#) property on the [openComplete](#) event.

The following code example shows how to set the active sheet when importing an Excel file.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

      SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<ejs-spreadsheet #spreadsheet
(openComplete)='openCompleteHandler($event)'
openUrl='https://services.syncfusion.com/angular/production/api/spreadsheet/
open'></ejs-spreadsheet>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj: SpreadsheetComponent;
    openCompleteHandler(args: Object) {
      if (this.spreadsheetObj) {
        this.spreadsheetObj.activeSheetIndex = 2;
      }
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Import an excel document using file uploader in ##Platform_Name## Spreadsheet component

If you explore your machine to select and upload an excel document using the file uploader, you will receive the uploaded document as a raw file in the [success](#) event of the file uploader. In this [success](#) event, you should pass the received raw file as an argument to the Spreadsheet's [open](#) method to see the appropriate output.

The following code example shows how to import an excel document using file uploader in spreadsheet.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
import { UploaderModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [
    DropDownButtonModule,
    UploaderModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template:
    "<div class='control-section'><ejs-spreadsheet #default
[openUrl]='openUrl' [saveUrl]='saveUrl'></ejs-spreadsheet><ejs-uploader
#defaultupload id='defaultfileupload' [asyncSettings]='path'
(success)='onUploadSuccess($event)'
[allowedExtensions]='allowExtensions'></ejs-uploader> </div>",
  })

```

```
export class AppComponent {
  @ViewChild('default')
  public spreadsheetObj: SpreadsheetComponent;
  public openUrl: string =
    'https://services.syncfusion.com/angular/production/api/spreadsheet/open';
  public saveUrl: string =
    'https://services.syncfusion.com/angular/production/api/spreadsheet/save';
  public path: Object = {
    saveUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Save',
    removeUrl:
      'https://services.syncfusion.com/angular/production/api/FileUploader/Remove'
  };
  public allowedExtensions: string = '.xlsx, .xls, .csv';
  onUploadSuccess(args) {
    if (args.operation == 'upload')
      this.spreadsheetObj.open({ file: args.file.rawFile });
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Identify the context menu opened in Angular Spreadsheet component

The Spreadsheet includes several context menus that will open and display depending on the action. When you right-click on a cell, for example, a context menu with options related to the cell element appears.

The class name returned by the [contextMenuBeforeOpen](#) event can be used to identify the context menu that is opened. The context menus and their class names are tabulated below.

| Class name | Context menu name |
|------------------|----------------------------|
| ----- ----- | |
| .e-sheet-content | Cell context menu |
| .e-toolbar-item | Footer context menu |
| .e-rowhdr-table | Row header context menu |
| .e-colhdr-table | Column header context menu |

The following code example shows how to identify the context menu opened.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { DropDownButtonModule } from '@syncfusion/ej2-angular-splitbuttons'
import { SpreadsheetAllModule } from '@syncfusion/ej2-angular-spreadsheet'
```



```
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { closest } from '@syncfusion/ej2-base';
import { BeforeOpenCloseMenuEventArgs } from '@syncfusion/ej2-navigations';
@Component({
  imports: [
    DropDownButtonModule,
    SpreadsheetAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: "<ejs-spreadsheet #spreadsheet
(contextMenuBeforeOpen)='contextMenuBeforeOpen($event)'></ejs-spreadsheet>"
})
export class AppComponent {
  @ViewChild('spreadsheet')
  public spreadsheetObj!: SpreadsheetComponent;
  contextMenuBeforeOpen(args: BeforeOpenCloseMenuEventArgs) {
    if (closest(args.event.target as Element, '.e-sheet-content')) {
      console.log('Cell Context Menu');
    } else if (closest(args.event.target as Element, '.e-colhdr-table')) {
      console.log('Column Header Context Menu');
    } else if (closest(args.event.target as Element, '.e-rowhdr-table')) {
      console.log('Row Header Context Menu');
    } else if (closest(args.event.target as Element, '.e-toolbar-item')) {
      console.log('Footer Context Menu');
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Save and open Spreadsheet data as a Base64 string in Angular Spreadsheet component

In the Spreadsheet component, there is currently no direct option to save and open data as a Base64 string. You can achieve this by saving the Spreadsheet data as blob data and then converting that saved blob data to a Base64 string using FileReader.

You can get the Spreadsheet data as blob in the [saveComplete](#) event when you set the `needBlobData` as `true` and `isFullPost` as `false` in the [beforeSave](#) event.

The following code example shows how to save and open the spreadsheet data as base64 string.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
```

```

import { SpreadsheetComponent, BeforeSaveEventArgs, SaveCompleteEventArgs }
from '@syncfusion/ej2-angular-spreadsheet';
import { data } from './datasource';
@Component({
  imports: [

    SpreadsheetModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div class="control-section">
    <button class="e-btn custom-btn" (click)="import()">Import
Base64</button>
    <button class="e-btn custom-btn" (click)="export()">Export as
Base64</button>
    <ejs-spreadsheet #spreadsheet
openUrl="https://services.syncfusion.com/angular/production/api/spreadsheet/
open" (beforeSave)="beforeSave($event)"
(saveComplete)="saveComplete($event)">
      <e-sheets>
        <e-sheet name="Car Sales Report">
          <e-ranges>
            <e-range [dataSource]="data"></e-range>
          </e-ranges>
          <e-columns>
            <e-column [width]=180></e-column>
            <e-column [width]=130></e-column>
            <e-column [width]=130></e-column>
            <e-column [width]=180></e-column>
            <e-column [width]=130></e-column>
            <e-column [width]=120></e-column>
          </e-columns>
        </e-sheet>
      </e-sheets>
    </ejs-spreadsheet>
  </div>`
})
export class AppComponent {
  @ViewChild('spreadsheet')
  spreadsheetObj!: SpreadsheetComponent;
  data: Object[] = data;
  base64String!: string | ArrayBuffer;
  beforeSave(args: BeforeSaveEventArgs): void {
    args.needBlobData = true; // To trigger the saveComplete event.
    args.isFullPost = false; // Get the spreadsheet data as blob data in
the saveComplete event.
  };
  saveComplete(args: SaveCompleteEventArgs): void {
    // Convert blob data to base64 string.
    let reader: FileReader = new FileReader();
    reader.readAsDataURL(args.blobData);
    reader.onloadend = () => {
      this.base64String = reader.result ? reader.result : '';
    };
  };
  import(): void {
    fetch(this.base64String as string)

```

```

        .then((response) => response.blob())
        .then((fileBlob) => {
            let file: File = new File([fileBlob], 'Sample.xlsx');
            this.spreadsheetObj.open({ file: file });
        });
    };
    export(): void {
        this.spreadsheetObj.save({
            url:
                'https://services.syncfusion.com/angular/production/api/spreadsheet/save',
            fileName: 'Worksheet',
            saveType: 'Xlsx',
        }); // Specifies the save URL, file name, file type need to be
        saved.
        // Logs base64 string into the console.
        console.log('Base64 String - ', this.base64String);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Insert a sheet programmatically and make it the active sheet in Angular Spreadsheet component

A sheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data. Using [insertSheet](#) method, you can insert one or more sheets at the desired index. Then, you can make the inserted sheet as active sheet by focusing the start cell of that sheet using the [goTo](#) method.

The following code example shows how to insert a sheet programmatically and make it the active sheet.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, ViewChild } from '@angular/core';
import { SpreadsheetComponent } from '@syncfusion/ej2-angular-spreadsheet';
import { data, employeeData } from './datasource';
@Component({
  imports: [

      SpreadsheetModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<div >
    <button class="e-btn custom-btn" (click)="onClick()">Insert
    Sheet</button>
    <ejs-spreadsheet #spreadsheet >
      <e-sheets>
        <e-sheet name="Car Sales Report">

```

```

        <e-ranges>
          <e-range [dataSource]="dataSource"></e-range>
        </e-ranges>
        <e-columns>
          <e-column [width]=180></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=180></e-column>
          <e-column [width]=130></e-column>
          <e-column [width]=120></e-column>
        </e-columns>
      </e-sheet>
    </e-sheets>
  </ejs-spreadsheet>
</div>`
  })
  export class AppComponent {
    @ViewChild('spreadsheet')
    spreadsheetObj!: SpreadsheetComponent;
    dataSource: Object[] = data;
    onClick(): void {
      this.spreadsheetObj.insertSheet(
        [
          {
            index: 1,
            name: 'new_sheet',
            ranges: [
              {
                dataSource: employeeData,
                startCell: 'A1'
              },
            ],
          },
        ],
      );
      // Use the timeout function to wait until the sheet is inserted.
      setTimeout(() => {
        // Method for switching to a new sheet.
        this.spreadsheetObj.goTo('new_sheet!A1');
      })
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Get the filtered rows in Angular Spreadsheet component

Filtering allows you to view specific rows in a spreadsheet while hiding the others. The [allowFiltering](#) property allows you to enable or disable filtering functionality through the UI. You can also use the [allowFiltering](#) property and [applyFilter](#) method combination to filter data via code behind. The filtered

rows can be identified by iterating through the row collection on the sheet and using the `isFiltered` property available in each row object.

The following code example shows how to get the filtered rows.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SpreadsheetModule } from '@syncfusion/ej2-angular-spreadsheet'
import { Component, OnInit, ViewChild } from '@angular/core';
import { defaultData } from './datasource';
import { PredicateModel } from '@syncfusion/ej2-grids';
import { ExtendedRowModel, SpreadsheetComponent, UsedRangeModel, SheetModel } from '@syncfusion/ej2-angular-spreadsheet';
@Component({
  imports: [

    SpreadsheetModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<button class="e-btn custom-btn" (click)="onClick()">Get
Filtered Rows</button>
<ejs-spreadsheet #spreadsheet (created)='created()'>
  <e-sheets>
    <e-sheet>
      <e-ranges>
        <e-range [dataSource]='filterData'>
        </e-range>
      </e-ranges>
      <e-columns>
        <e-column [width]=150></e-column>
        <e-column [width]=120></e-column>
        <e-column [width]=100></e-column>
      </e-columns>
    </e-sheet>
  </e-sheets>
</ejs-spreadsheet>`
})
export class AppComponent implements OnInit {
  public filterData?: object[];
  @ViewChild('spreadsheet')
  public spreadsheetObj!: SpreadsheetComponent;
  ngOnInit(): void {
    this.filterData = defaultData;
  }
  created() {
    // Applies cell formatting to specified range of the active sheet.
    this.spreadsheetObj.cellFormat({ fontWeight: 'bold', textAlign:
'center', verticalAlign: 'middle' }, 'A1:F1');
    // Construct the predicate model to be updated to the data.
    let predicates: PredicateModel[] = [{
      field: 'C',
      operator: 'equal',
      value: 'Pink',
      matchCase: false
    }]
```

```

    }];
    // Apply filter to the specified range.
    this.spreadsheetObj.applyFilter(predicates, 'A1:C7');
  };
  onClick(): void {
    let activeSheet: SheetModel = this.spreadsheetObj.getActiveSheet();
    let usedRange: UsedRangeModel = activeSheet.usedRange!;
    for (let i: number = 0; i <= usedRange.rowIndex!; i++) {
      // Get the filtered row using isFiltered property.
      let filteredRow: Object = (activeSheet.rows![i] as
ExtendedRowModel).isFiltered!;
      if (!filteredRow) {
        let rowData: Object = this.spreadsheetObj.getRowData(i);
        console.log("Row:", i + 1, "Cells", rowData);
      }
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Mobile responsiveness in Angular Spreadsheet component

The Spreadsheet control rendered in desktop mode will be adaptive in all mobile devices where the layout gets adjusted based on their parent element's dimensions to accommodate any resolution.

You can see the overflowed items of ribbon header, ribbon content, and sheet tab using touch and swipe action. The right navigation arrow is added at the end of the ribbon content through which the user can navigate towards overflowed items. Once you reached the rightmost end of the ribbon content, the right navigation arrow will change to left navigation arrow through which you can navigate to the left of the ribbon content.



| | A | B |
|----|----------------------|----------------|
| 1 | Customer Name | Model |
| 2 | Romona Heaslip | Taurus |
| 3 | Clare Batterton | Sparrow |
| 4 | Eamon Traise | Grand Cherokee |
| 5 | Julius Gorner | GTO |
| 6 | Jenna Schoolfield | LX |
| 7 | Marylynne Harring | Catera |
| 8 | Vilhelmina Leipelt | 7 Series |
| 9 | Barby Heisler | Corvette |
| 10 | Karyn Boik | Regal |
| 11 | Jeanette Pamplin | S4 |
| 12 | Cristi Espinos | TL |
| 13 | Issy Humm | Club Wagon |
| 14 | Tuesday Fautly | V8 Vantage |
| 15 | Rosemaria Thomann | Caravan |
| 16 | Lyell Fuentez | Bravada |
| 17 | Raynell Layne | Colorado |
| 18 | Raye Whines | 4Runner |
| 19 | Virgina Aharoni | TSX |

Note

You can refer to our [Angular Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Spreadsheet example](#) to know how to present and manipulate data.

Stepper

Getting started with Angular Stepper component

This section explains how to create a simple Stepper, and demonstrate the basic usage of the Stepper module in an Angular environment.

Dependencies

The list of dependencies required to use the Stepper module in your application is given below:

```
`javascript
|-- @syncfusion/ej2-angular-navigations
|-- @syncfusion/ej2-base
```

```
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-angular-base
,
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
npm install -g @angular/cli
,
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
ng new my-app
cd my-app
,
```

Installing Syncfusion Stepper Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-navigations](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-navigations --save
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-navigations@ngcc](#) package to the application.

```
`bash
```



```
npm install @syncfusion/ej2-angular-navigations@ngcc --save
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-navigations:"21.1.35-ngcc"
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Stepper module

Import Stepper module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-navigations`.

```
`javascript
```

```
import { NgModule } from "@angular/core";
```

```
import { BrowserModule } from "@angular/platform-browser";
```

```
// Import Syncfusion Stepper module from stepper package.
```

```
import { StepperModule, StepperAllModule } from '@syncfusion/ej2-angular-navigations';
```

```
import { AppComponent } from "../app.component";
```

```
@NgModule({
```

```
  imports: [BrowserModule, StepperAllModule, StepperModule], // Registering EJ2 Stepper Module.
```

```
  declarations: [AppComponent],
```

```
  bootstrap: [AppComponent],
```

```
})
```

```
export class AppModule {}
```

Adding CSS reference

Add Stepper component's styles as given below in `style.css`.

```
`css
```

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

```
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
```

```
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
```

Adding Syncfusion Stepper component

Modify the template in `app.component.ts` file with `ejs-stepper` to render the Stepper component.

```
`javascript
import { Component } from "@angular/core";
import { StepModel, Stepper } from '@syncfusion/ej2-angular-navigations';
@Component({
  selector: "app-root",
  template: `<!-- To Render Stepper. -->
<div>
<ejs-stepper id="default"></ejs-stepper>
</div>`,
})
export class AppComponent {
}
```

Adding Steps

You can define steps by using `<e-step>` tag directive.

```
`javascript
import { Component } from "@angular/core";
@Component({
  selector: "app-root",
  template: `<!-- To Render Stepper. -->
<div>
<ejs-stepper id="default">
<e-steps>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
<e-step></e-step>
</e-steps>
</ejs-stepper>
</div>`,
})
export class AppComponent {
```

```
}
,
```

Running the application

Run the application in the browser using the following command:

```
,
```

```
ng serve
```

```
,
```

The following example shows a default Stepper component.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="defaultStepper">
  <ejs-stepper>
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

Configure icon and label

You can define the step icon and label by setting the `iconCss` and `label` properties using `<e-step>` tag directive.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
```

```
imports: [ StepperAllModule, StepperModule ],
standalone: true,
selector: 'app-root',
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="iconWithLabel">
  <ejs-stepper>
    <e-steps>
      <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
      <e-step label="Address" iconCss="sf-icon-user"></e-step>
      <e-step label="Delivery" iconCss="sf-icon-transport"></e-step>
      <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
      <e-step label="Ordered" iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAArAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQAIwC
3AQkBGGAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwcHBQUEAw
MBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwcGBgUEAwMBAQEBAwM
EBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1foBAGQFBgcH/iwNDAwLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABABgAAAAAD8wOWAAYAQgBaAGwArQDuAAABZ
```

```

c:fAaUhlwIHtY8PNS8CKwIPHQEHLWwEjDwElLwMjNz0BJzcfBTcfAg8BLWY3OwEfAqCvHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPcBAGQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQEA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4IERESFBQUFQQDagECagMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGAwMBAQMDBg
YICQoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
YBQBQAQQFCAoLDQ8QehIUFRYXAAAAAQAaaaaa/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEWcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDagEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDagH+aaICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAQCAQECawQEBAYFBwYHCAGICAgICAcIBgcFBgQEBAICAsH6jAFKjPpu/Z3NwgJZ/dzDaf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBgICakIBwGHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICACHwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0daAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgghBQEBaIkCagIEAwQFDA0SBwcGAgIBAQICBgCHBxYK
CQkJCACHBgUFBAMCAQEBAQIDAwQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDagEBAQECAwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWfXyXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYQGgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBAQQCAQEBAwKRNRIHBqADChI1DQoFagEBAGMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBaIIDAuUGBwgJCgICAQENAQEFaWIDAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICAgJBwGHBgcGBgUFBAQEBOQIagEBaf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICAKJCwsMKsckIiAeGxoYfHqTERAPDQwLCgkIDxsJBQSQnEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPPfz0REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGagIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISIiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAEEAACAFgABAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwAlwABA
AAAAALABIawwADAEEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lubiBNZXRybyBtDhVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAIYQB0AGU
AZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAA
AAAAAAAAAAAAAAAAAAAAAABAgEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlciltb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXXkFY2h1Y2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
.iconWithLabel {
margin-top: 30px;
padding: 30px;
}
.iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;

```

```

-moz-osx-font-smoothing: grayscale;
}
.iconWithLabel .sf-icon-cart:before { content: "\e710"; }
.iconWithLabel .sf-icon-user:before { content: "\e708"; }
.iconWithLabel .sf-icon-transport:before { content: "\e702"; }
.iconWithLabel .sf-icon-payment:before { content: "\e706"; }
.iconWithLabel .sf-icon-success:before { content: "\e715"; }

```

Steps in Angular Stepper component

The Angular Stepper allows you to add steps using the `<e-step>` tag directive. Each step can be configured with options such as `iconCss`, `text`, `label`, `cssClass` and more.

Adding steps

You can define the icon and text content for each step using the `iconCss`, `text` and `label` properties.

Defining icon CSS

You can define the CSS class to show the icon for each step using the `iconCss` property.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="stepperIcon">
  <ejs-stepper>
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>

```

APP.COMPONENT.CSS

```

@font-face {

```

```
font-family: 'Default';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQAGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAAQAArxtT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAgAAAAAMAAUAMAAQAAABQABABKAAAADAIAIAIABOCc5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAAwADAAMAawADAAAAEABAAACAAMABQAAAAAAAEQAIwC
3AQkBgGgAAAAFAAAAAAP0A18APwB/AIkAxAqDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBggICAKJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkKJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkKJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkKJCQoLCgkKJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkKJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHIY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BjZcfBTcfAg8BLwY30wEfAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEAF5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFYQYGBAgFBgIWA
gQHCPcBAGQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXfHQUEXEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCASMDQ4IERESFBQUFQDDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXfYVFBwI8EA8NCwo
IBQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBGcFBgQEBaICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgHBwgICAKIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY
8HDWYdAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBaIkCAGIEAwQFDA0SBwCAGIBAQICBgCHBxYK
CQkKJCAcHBgUFBAQCAQEBAQIDAQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQDDAgEBAQECAwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFh
YWFxYXfHcXfXcWFxYXfHYWFHYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYQGgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBAgQCAQEBAwKRNRIHBqADChI1DQoFAGEBAGMEBAOMew8eTw4IVxkXCwkJ
BwYCOAIBAIIDAwUGBwgJCgICAQENAQEFawIDAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAQYDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW
QEAQEBAGICBAMFBQUGBwCICAgJBwgHBGcGBgUFBAQEBQqIAGEBaf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFbGYICAKJCwsMKSckIiAeGxoYfHqTTERAPDQwLCgkIDxsJBQUBQqNEAkKcwwNDxAREXQWGBob
HiAiJccCoAMDawQECA8XPRcKCgUPFz0REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGagIwXmMXFBIIcCsQKAEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiiIiIjIkJAPRwB8AQmCQMBARQuNgsMDGcIJCYnmyZOTycmJiYlJSQJiiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAABAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAABAAAFgABAAAAAABAAAsAHQABAAAAAAGAAcAKAABAAAAAABAAKAwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZwZhdWx0UmVndWxhcRlZmF1bHREZwZhdWx0VmVyc2lvbiAxLjBEZwZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
```

```

GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.stepperIcon {
    margin-top: 30px;
    padding: 30px;
}
.stepperIcon [class^="sf-icon-"], .stepperIcon [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperIcon .sf-icon-cart:before { content: "\e710"; }
.stepperIcon .sf-icon-transport:before { content: "\e702"; }
.stepperIcon .sf-icon-payment:before { content: "\e706"; }
.stepperIcon .sf-icon-success:before { content: "\e715"; }

```

Defining text content

You can define text instead of an icon by setting the `text` property and display label content for a step using the `label` property.

When both label and text are defined, the label takes priority for display based on the `stepType`.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


APP.COMPONENT.HTML

```

<div class="stepper">
  <div class="stepText">
    <ejs-stepper stepType="Indicator">
      <e-steps>
        <e-step text="A"></e-step>
        <e-step text="B"></e-step>
        <e-step text="C"></e-step>
        <e-step text="D"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
  <div class="labelStepper">
    <ejs-stepper>
      <e-steps>
        <e-step label="Cart"></e-step>
        <e-step label="Delivery Address"></e-step>
        <e-step label="Payment"></e-step>
        <e-step label="Ordered"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</div>

```

Optional steps

You can show whether the step is optional or not by using `optional` property. By default, the `optional` property is `false`.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="stepperOptional">
  <ejs-stepper>
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
    </e-steps>
  </ejs-stepper>
</div>

```

```

    <e-step iconCss="sf-icon-transport"></e-step>
    <e-step iconCss="sf-icon-payment" optional="true"></e-step>
    <e-step iconCss="sf-icon-success"></e-step>
  </e-steps>
</ejs-stepper>
</div>

```

APP.COMPONENT.CSS

```

@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYWI1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAABAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAAAEAAAABA
AAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcC5wbnCOCQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAawADAAAMAAwADAAAAAEABAACAAMABQAAAAAABAEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKCAgIBGcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkKCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkKJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkKJCQoLCgkKJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkKJCQoKCgkKCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCagJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcFbTcfAg8BLwY30wEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEAF5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFYGBAgFBgIWA
gQHCPCBAgQGBgKcgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMCMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFB1SEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBGcFBgQEBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDWydAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQEBAikCagIEAwQFDA0SBwCGAgIBAQICBgCHBxYK
CQkKJCAcHBgUFBAQCAQEBAQIDAQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgEBAQECAwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBaIIDAUGBwgJCgICAQENAQEFawIDAQECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAQYDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW

```

```

QEAgEBAQICBAMFBQUGBwcICAgJBwgHBgcGBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICakJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUFBQqNEAkKCwwNDxARExQWGBob
HiAiJccCoAMDawQECA8XPRcKCgUPFz0REAkIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAgIwXmMXFBIICCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNGsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAEEAaAFgABAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAyQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAA
AAAAAAAAAAAAAAAAAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4ldXNlciltb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('true');
    font-weight: normal;
    font-style: normal;
}
.stepperOptional {
    margin-top: 30px;
    padding: 30px;
}
.stepperOptional [class^="sf-icon-"], .stepperOptional [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperOptional .sf-icon-cart:before { content: "\e710"; }
.stepperOptional .sf-icon-transport:before { content: "\e702"; }
.stepperOptional .sf-icon-payment:before { content: "\e706"; }
.stepperOptional .sf-icon-success:before { content: "\e715"; }

```

Disabling steps

You can use the `disabled` property to disable a step, preventing user interaction when set to `true`. By default, the value is `false`.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithIcon">
  <ejs-stepper>
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment" disabled="true"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABGABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAAADAAIAAIABOcC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAAAAAAEQaiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxAqDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz8OOWEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHbWcICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHbGgICAKJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkKJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkKJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkKJCQoLCgkKJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkKJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKJCgGuAQIGehYJBKYp/10ICAcG
BQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDAwLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABGAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BjZcfBTcfAg8BLwY3OwEfAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEAF5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFYQYGBAgFBgIWA
gQHCPcBAGQGBgkKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECAyGCAkLCwwNDg8PEBAQEAE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECAyGCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
```

```

IBQQBAQQFCAoLDQ8QEHtUFRYXAAAAAQAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAgEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB AQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwgICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDWydAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWefBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCGgHBQEBAikCAGIEAwQFDA0SBwcGAGIBAQICBgcHBxYK
CQkJCACHBgUFBAQCAQEBAQIDAQFBQYGBwcPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHtHB
wYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQDAGEB AQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCAgJCgsLDA0MDQ0NDBk2EQYGGqYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRIHBqADChI1DQoFAgEBAGMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAiIDAUGBwgJCgICAQENAEFAwIDAQECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwcGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwcICAgJBwgHBgcGBgUFBAQEBAQIAGEBaf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICAKJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBQqNEAKKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAGIwXmMXFBIICCsKqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAEEAaAFgABAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGU
AZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAoAAAAA
AAAAAABAAAAAAYBAgEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2R2Zn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}
 stepperWithIcon {
margin-top: 30px;
padding: 30px;
}
 stepperWithIcon [class^="sf-icon-"], .stepperWithIcon [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
 stepperWithIcon .sf-icon-cart:before { content: "\e710"; }
 stepperWithIcon .sf-icon-transport:before { content: "\e702"; }
 stepperWithIcon .sf-icon-payment:before { content: "\e706"; }
 stepperWithIcon .sf-icon-success:before { content: "\e715"; }

```

Setting active step

You can set the active step by specifying its index using the [activeStep](#) property. The default value is 0.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperIcon">
  <ejs-stepper activeStep="1">
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAAAAYAAAAyb
G9jYQhUBlAAAAH4AADm1heHABFgErAAABCAAAACBuYW11uF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABGABAAAAAQAArxT6wV8PPPU
ACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAAADAAIAAIABOcC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAAEQaiwC
3AQkBGgAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQCwBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
```

```

CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCChCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBgCh/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABZ
cfAwUhLwIHIY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcFbTcfAg8BLwY30wEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPCBAgQGBgKcGsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEASPDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQEXEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAAsMDQ4IERESFBQUDFQDAGECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDFMAMHwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEASPDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAGICAgICAcIBgcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCChCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCChCAcICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBaikaAgIEAwQFDA0SBwCAGIBAQICBgCHBxYK
CQkJCAcHBgUFBAQCAQEBAQIDAQwQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAGEBQECawMEBAYFBgCHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhCxfXcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFagEBAGMEBAOMew8eTw4IVxkXCwkJ
BwYCOAIBaIIDAUGBwgJCgICAQENAEFAwIDAQECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQCEBAYDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEA
QEAQEBAGICBAMFBQUGBwCICAgJBwgHBgCGBgUFBAQEBCQIaAgEBAf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICakJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAgIwXmMXFBIIcCsKqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAA
AAADAACADwABAAAAAEEAACAFgABAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcmcGU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAx4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBIAHIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAA
AAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXxkFY2hly2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
 stepperIcon {
margin-top: 30px;
padding: 30px;
}
 stepperIcon [class^="sf-icon-"], .stepperIcon [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;

```

```

font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.stepperIcon .sf-icon-cart:before { content: "\e710"; }
.stepperIcon .sf-icon-transport:before { content: "\e702"; }
.stepperIcon .sf-icon-payment:before { content: "\e706"; }
.stepperIcon .sf-icon-success:before { content: "\e715"; }

```

Step status

Each step's progress state can be specified using the `status` property. The possible values are `NotStarted`, `InProgress` and `Completed`. By default, the value is `NotStarted`.

APP.COMPONENT.TS

```

import { Component, ViewChild } from "@angular/core";
import { StepperComponent, StepperChangedEventArgs } from "@syncfusion/ej2-angular-navigations";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  @ViewChild('ejStepper') stepper: StepperComponent | any;
  handleStepChanged = (args: StepperChangedEventArgs) => {
    let status = this.stepper.steps[1].status
    this.updateStatus(status)
  }
  updateStatus = (stepStatus: string) => {
    let statusMap = {
      'NotStarted' : { text: 'Your payment has not started yet', color: '#e74d4d' },
      'InProgress' : { text: 'Processing your payment', color: 'orange' },
      'Completed' : { text: 'Payment successful', color: '#4CAF50' }
    }
    let currentStatus = document.getElementById("paymentStatus");
    if (currentStatus) {
      let {text, color} = (statusMap as any)[stepStatus];
      currentStatus.innerText = text;
      currentStatus.style.backgroundColor = color;
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```



```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepper-status-section">
  <ejs-stepper #ejStepper id="stepper"
    (stepChanged)="handleStepChanged($event)">
    <e-steps>
      <e-step iconCss="sf-icon-cart" label="Cart"></e-step>
      <e-step iconCss="sf-icon-payment" label="Payment"></e-step>
      <e-step iconCss="sf-icon-success" label="Confirmation"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
<div id="paymentStatus">Your payment has not started yet</div>
```

APP.COMPONENT.CSS

```
#paymentStatus {
  padding: 10px;
  background-color: #e74d4d;
  color: white;
  border-radius: 5px;
  margin: 10px auto;
  text-align: center;
  font-weight: bold;
  max-width: 350px;
  margin-top: 50px;
}
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQAGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABAAAAAQAARxT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAAGAR8ABgAAAAAAGAA
AAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAABAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAAEAAAAABA
AAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAABQABABKAAAADAAIAAIBOOC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKCAgIBgcFBQOEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQAQwEBAQEBAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkKJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkKJCQoLCgkKJCkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkKJCQoKCgkKCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKJCgGuAQIGehYJBKYp/10ICAcG
BQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDAwLC
woJCQgHBgUEAwEBAQEBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY3OwEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
```

```

gQHCPcBAGQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUEExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4IERESFBQUFQDQAgECAGMEBAUG
BgYIBWgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBggoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBWgICAgICAgIBWYHBQYEB
AQDAGEBAGMEBAQGBQcGBWgICAgICAgIBWYHBQYEBAQDAGH+aAICAgQEBAYFBWYIBWgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBWYHCAGICAgICAcIBgcFBgQEBaICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwGICaKIBWgHBWYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBWYGBQUEBAICAQEBAQICBAQFBQYGBWCHCAgICQgHCAcHBgYFBQDQAwIBAQIDA
wQFBQYGBWcIBWgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQEBAikCAGIEAwQFDA0SBwcGAGIBAQICBgcHBxYK
CQkJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBWcPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwcWCgkKCAgHBWYFBQDQAgEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFAGEBAGMEBAOMew8eTw4IVxkXCwkJ
BwYCOAIBAIIDAuUGBwgJCgICAQENAEFAWIDAGECAGMFAwMEBAUDBAMFAWIBAQECAwMEBAUGBgYHC
AgICQgHBwcGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwcICAgJBWgHBgcGBgUFBAQEBOQIAgEBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICaKJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUFBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAGIwXmMXFBIIcCsqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISIiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAAAAAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAAAEAAcAFgABAAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Zlc2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxA4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBIAHIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAA
AAAAAAAAAAAAAAAAAAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}

.stepper-status-section [class^="sf-icon-"], .stepper-status-section
[class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

.stepper-status-section .sf-icon-cart:before { content: "\e710"; }
.stepper-status-section .sf-icon-payment:before { content: "\e706"; }

```

```
.stepper-status-section .sf-icon-success:before { content: "\e715"; }
.stepper-status-section {
  width: 75%;
  margin: 0px auto;
  min-width: 85px;
  padding: 25px 0;
}
```

Step styling

You can use the `cssClass` property to customize the appearance of each step.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithIcon">
  <ejs-stepper id="stepper">
    <e-steps>
      <e-step iconCss="sf-icon-cart" label="Cart"></e-step>
      <e-step iconCss="sf-icon-transport" label="Delivery Address"></e-step>
      <e-step iconCss="sf-icon-payment" label="Payment" cssClass="custom-
step" optional=true></e-step>
      <e-step iconCss="sf-icon-success" label="Confirmation"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAADm1heHABFgErAAABCAAAACBuYW11luF5THQAActgAAAIlcG9zdJ8LuoMAAA
```

516

```

    font-weight: normal;
    font-style: normal;
}

.stepperWithIcon {
    margin-top: 30px;
    padding: 30px;
}

.stepperWithIcon [class^="sf-icon-"], .stepperWithIcon [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

.stepperWithIcon .sf-icon-cart:before { content: "\e710"; }
.stepperWithIcon .sf-icon-transport:before { content: "\e702"; }
.stepperWithIcon .sf-icon-payment:before { content: "\e706"; }
.stepperWithIcon .sf-icon-success:before { content: "\e715"; }
.stepperWithIcon #stepper .custom-step .e-step-label-optional {
    font-style: italic;
    font-weight: 900;
    color: #299100;
}

```

Step validation

You can set the validation state for each step to displaying a success or error icon by using `isValid` property.

To know more about Stepper validation, please refer to the [Validation](#) section.

Step types in Angular Stepper component

The Stepper component provides support for displaying steps with the following step types.

Default type

In default type, the Stepper displays steps with a combination of both indicators and labels by setting the `stepType` property as `Default`. By default, the Stepper displays steps in the `Default` type.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="iconWithLabel">
  <ejs-stepper stepType="Default">
    <e-steps>
      <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
      <e-step label="Delivery Address" iconCss="sf-icon-transport"></e-step>
      <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
      <e-step label="Confirmation" iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAAUAMAAQAABQABABKAAAADAIAAIAABOC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQAiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKCAgIBGcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBWgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAd3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkKCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDAwLC
woJCQgHBgUEAwEBAQEEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABbz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY30wEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPcBAGQGBgkKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUEExEQDw0LCgGfBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAA8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcxYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRmMMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBWgICAgICAgIBwYHBQYEB
```

```

AQDAgEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAGICAgICAcIBgcFBgQEBaICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwgICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDWydAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAgENDAwKCgGHBQEBAikCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYK
CQkJCAcHBgUFBAQCAQEBAQIDAQFBQYGBwcPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwcWCgkKCAGHBwYFBQQDAgEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBAgQCAQEBAwKRNRIHBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAiIDAUGBwgJCgICAQENAQEFawIDAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwcGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAw
QEAgEBAQICBAMFBQUGBwcICAgJBwgHBgcGBgUFBAQEBCQiAgEBaf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICAKJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REAkIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAGIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAA
AAADAACADWABAAAAAEEAACAFgABAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALWABA
AAAAAALABIAWwADAAEEECQAAAAIAbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4Aiw
ADAAEEECQAEAA4AmQADAAEEECQAFABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQBLAGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxA4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBIAHIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAA
AAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1b2R2Rn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZkFY2hly2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}

.iconWithLabel {
    margin-top: 30px;
    padding: 30px;
}

.iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}

.iconWithLabel .sf-icon-cart:before { content: "\e710"; }
.iconWithLabel .sf-icon-transport:before { content: "\e702"; }
.iconWithLabel .sf-icon-payment:before { content: "\e706"; }
.iconWithLabel .sf-icon-success:before { content: "\e715"; }

```

Label type

In label type, the Stepper displays the steps with only the step labels by setting the [stepType](#) property as **Label**.

When both label and text are defined, the label takes priority in displaying the steps.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="iconWithLabel">
  <ejs-stepper stepType="Label">
    <e-steps>
      <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
      <e-step label="Delivery Address" iconCss="sf-icon-transport"></e-step>
      <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
      <e-step label="Confirmation" iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAANmhoZWEIUQQHAAArAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYWI1luF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAQAAZAABQAAAOkCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAAAUAAAAQAABQABABKAAAADAAIAAIABOcC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAAwADAAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQaiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxdDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
```



```

xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwcHBQUEAw
MBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkJCQoLCgkJCQkIBwcGBgUEAwMBAQEBAwM
EBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABz
cfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY3OwEfAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCpCBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQEGBAYGCAkLCwwNDg8PEBAQEA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQEGBAYGCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXfXyVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAQAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAgEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBGcHBwgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BagoCAGENDAwKCgGHBQEBAikCagIEAwQFDA0SBwcGAGIBAQICBgcHBxYK
CQkJCAcHBgUFBAQCAQEBAQIDAQFBQYGBwcPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQCAQEBAgYHBwcWCgkKCAgHBwYFBQDAgEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFAQCAQICBAUGCAgJCgsLDA0MDQ0NDBk2EQYgGqYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwKRNRIHBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAiIDAwUGBwgJCgICAQENAQEFawIDAgECAGMFawMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwcGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwcICAgJBwgHBgcGBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICakJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAGIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISIiIiIjIkJAPRwB8AQmQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAA
AAADAACAdwABAAAAAEEAaAFgABAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGWAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAIBnAGUAbgBIAHIAAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAIBNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAA
AAAAAAAAAAAAAAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}

.iconWithLabel {
    margin-top: 30px;
    padding: 30px;
}

```

```

}

.iconWithLabel [class^="sf-icon-"], .iconWithLabel [class*=" sf-icon-"] {
  font-family: 'Default' !important;
  speak: none;
  font-style: normal;
  font-weight: normal;
  font-variant: normal;
  text-transform: none;
  line-height: inherit;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.iconWithLabel .sf-icon-cart:before { content: "\e710"; }
.iconWithLabel .sf-icon-transport:before { content: "\e702"; }
.iconWithLabel .sf-icon-payment:before { content: "\e706"; }
.iconWithLabel .sf-icon-success:before { content: "\e715"; }

```

Label positions

You can display the label on the top, bottom, left, or right side of the steps using the [labelPosition](#) property.

The following label positions are supported in Stepper:

Value	Description
----- -----	
Top	Positions the label at the top of each step.
Bottom	Positions the label at the bottom of each step.
Start	Positions the label to the left side of each step.
End	Positions the label to the right side of each step.

APP.COMPONENT.TS

```

import { Component, ViewChild } from "@angular/core";
import { StepperComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  @ViewChild('ejStepper') stepper: StepperComponent | any;
  public updateLabelPosition(args: any): void {
    this.stepper.labelPosition = args.currentTarget.value;
  };
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithLabel">
  <div class="e-btn-group labelPosition">
    <input type="radio" id="start" name="position" value="start"
(click)="updateLabelPosition($event);" />
    <label class="e-btn" for="start">Start</label>
    <input type="radio" id="end" name="position" value="end"
(click)="updateLabelPosition($event);" />
    <label class="e-btn" for="end">End</label>
    <input type="radio" id="top" name="position" value="top"
(click)="updateLabelPosition($event);" />
    <label class="e-btn" for="top">Top</label>
    <input type="radio" id="bottom" name="position" value="bottom"
(click)="updateLabelPosition($event);" checked />
    <label class="e-btn" for="bottom">Bottom</label>
  </div>
  <div class="stepper-section">
    <ejs-stepper #ejStepper>
      <e-steps>
        <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
        <e-step label="Shipped" iconCss="sf-icon-transport"></e-step>
        <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
        <e-step label="Delivered" iconCss="sf-icon-success"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXcDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQQAaZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAGAAAAAAMAAUAAMAAQAABQABABKAAAADAIAAIABOC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAAEQaiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAd3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCakJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
```

```

EBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCagJCgoLDA0NDQECBAUGBwQI1foBAGQFBgcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABbz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BjZcfBTcfAg8BLwY3OwEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFYQYGBAgFBgIWA
gQHCPcBAGQGBgGKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBaQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDagECagMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAQAaaaaa/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDagEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDagH+aaICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECawQEBAYFBwYHCAgICAgICAcIBgcFBgQEBaICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECagQEBQUGBgCHBgICakIBwgHBwYGBQUEAwMCAQECawMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQDDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0daQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBaIkCAGIEAwQFDA0SBwCAGIBAQICBgCHBxYK
CQkJCAcHBgUFBAMCAQEBQIDAQWFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQDDagEBAQECawMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhCxfXcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYgGgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBawKRNRIHBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBaIIDAuUGBwgJCgICAQENAQEFawIDAgECagMFawMEBAUDBAMFAwIBAQECawMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAYDIgICAQECaiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAw
QEAgEBAQICBAMFBQUGBwCICAgJBwgHBgcGBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICakJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECa8XPRcKCgUPFz0REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAAAAAAAAEAAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAA
AAADAAcADwABAAAAAAAEAAcAFgABAAAAAAAFaAsAHQABAAAAAAAGAAcAKAABAAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZwZhdWx0UmVndWxhcRlZmF1bHREZwZhdWx0VmVyc2lvbiAxLjBEZwZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcuU3luY2Zlc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAIAYQB0AGU
AZAAgAHUAacwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQADQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAA
AAAAAAAAAAAAAAAAAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4ldXNlcil1tb2RpbmZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.stepperWithLabel {
    margin-top: 30px;
    padding: 30px;
}
.stepperWithLabel .labelPosition {
    margin-left: 38%
}
.stepper-section {
    margin: 50px 100px;
}

```

```
.stepperWithLabel [class^="sf-icon-"], .stepperWithLabel [class*=" sf-icon-"]
{
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperWithLabel .sf-icon-cart:before { content: "\e710"; }
.stepperWithLabel .sf-icon-transport:before { content: "\e702"; }
.stepperWithLabel .sf-icon-payment:before { content: "\e706"; }
.stepperWithLabel .sf-icon-success:before { content: "\e715"; }
```

Indicator type

In indicator type, the Stepper displays steps with only the step indicators by setting the [stepType](#) property as **Indicator**.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepIndicator">
  <div class="indicatorOnly">
    <ejs-stepper stepType="Indicator">
      <e-steps>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
  <div class="textOnly">
```

```

    <ejs-stepper stepType="Indicator">
      <e-steps>
        <e-step text="1"></e-step>
        <e-step text="2"></e-step>
        <e-step text="3"></e-step>
        <e-step text="4"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</div>

```

Orientations in Angular Stepper component

The Stepper component supports the display of steps in both horizontal and vertical orientations by using the [orientation](#) property.

Horizontal

In horizontal orientation, the steps are displayed in a side-by-side manner by setting the [orientation](#) property to **Horizontal**. By default, the steps are displayed in the horizontal orientation.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="horizontalStepper">
  <ejs-stepper orientation="horizontal">
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>

```

APP.COMPONENT.CSS

```

@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgT1MvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABGABAAAAAQAArxtT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAABAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIBOCc5wbncOCq5xxX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAADAAMAawADAAAAAEABAACAAMABQAAAAAABAEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQCEBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHIY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcFbTcfAg8BLwY30wEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEAF5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFYQGBAgFBgIWA
gQHCPCBAgQGBGgKCgsMDQ4PDxQEBApDw4NDAsLCQgGBgQCAQCECBAYGCAkLCwwNDg8PEBAQEAP8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUEXEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCASMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEzawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQCECBAYGCAkLCwwNDg8PEBAQEAP8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcxXfYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAGH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBGcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBAikCAGIEAwQFDA0SBwcGAGIBAQICBgCHBxYK
CQkJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgEBAQECAwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFh
YWFxYXfHcXfXcWFxYXfHYWFhYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYgGgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFAGEBAGMEBAOMew8eTw4IVxkXCwkJ
BwYCOAIBAIIDAwUGBwgJCgICAQENAQEFawIDAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICAgJBwgHBgCGBgUFBAQEBAQICAgEBAf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICakJCwsMKSckIiAeGxoYfHqTERRPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxAREXQWGBob
HiAiJccCoAMDawQECA8XPRcKCgUPFz0REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAgIwXmMXFBIIcCsKqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAABAAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAA
AAADAACADwABAAAAAABAAcAFgABAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAABAAcAwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW51

```

```

cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
.horizontalStepper {
    margin-top: 30px;
    padding: 30px;
}
.horizontalStepper [class^="sf-icon-"], .horizontalStepper [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.horizontalStepper .sf-icon-cart:before { content: "\e710"; }
.horizontalStepper .sf-icon-transport:before { content: "\e702"; }
.horizontalStepper .sf-icon-payment:before { content: "\e706"; }
.horizontalStepper .sf-icon-success:before { content: "\e715"; }

```

Vertical

You can display the steps one below the other vertically by setting the [orientation](#) property to **Vertical**.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML


```
<div class="verticalStepper">
  <ejs-stepper orientation="vertical">
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUblAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYWlluf5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFWEAAAAAAD9AABAAAAAABAAAAAABAAAAAQAARxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAABAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAABAA
AAAAAABAAAAAABAAAAAUGZFZABA5wLnFQAAAAAXAQAAAAAABAAAAAABAAAAAQAABAAAAEAAAAABA
AAAAQAAAAEAAAAAABgAAAAAABAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAABAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAAAABEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgCfBQOEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwCfBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
CHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGCHCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGCH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwWMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABZz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY30wEFAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPCBAgQGBgGKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAyGCAkLCwWNDg8PEBAQEAB8PDg
0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBAwUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAyGCAkLCwWNDg8PEBAQEAB8PDg0MCwoKCAyGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXfYyVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEHlUFrYXAAAAAQAaaaaa/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgCfBQOEBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BagoCagENDAwKCgGHBQEBAikCagIEAwQFDA0SBwCGAgIBAQICBgcHBxYK
CQkJCAcHBgUFBAMCAQEBAQIDAQwQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQDQAgEBAQECAwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFh
```

```

YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQQAQEBawkRNRiHBqADChI1DQoFagEBagMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAiIDAwUGBwgJCgICAQENAQEFawIDAgECagMFawMEBAUDBAMFawIBaQECAwMEBAUGBgYHC
AgICQgHBwcGBgYFBQQEBAYDIgICAQECAiICBAUGBwgJCQMBaGEMAQUDAwIDAQICBAQDBAQEBaQEaw
QEAgEBAQICBAMFBQUGBwcICAgJBwgHBgcGBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUdAwIBaQI
DAwUFBgYICakJCwsMKScKiiAeGxoYFhQTERAPDQwLCgkIDxsJBQUFBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGagIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAAAEAAcAFgABAAAAAAFAAsAHQABAAAAAAGAacAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEEECQAAAAIAbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiW
ADAAEEECQAEAA4AmQADAAEEECQAFABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAA
AAAAAAAAAAAAAAAAAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
    font-weight: normal;
    font-style: normal;
}
.verticalStepper {
    margin-top: 30px;
    padding: 30px;
    min-height: 450px;
    display: flex;
    justify-content: center;
}
.verticalStepper [class^="sf-icon-"], .verticalStepper [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.verticalStepper .sf-icon-cart:before { content: "\e710"; }
.verticalStepper .sf-icon-transport:before { content: "\e702"; }
.verticalStepper .sf-icon-payment:before { content: "\e706"; }
.verticalStepper .sf-icon-success:before { content: "\e715"; }

```

Linear Flow in Angular Stepper component

The Stepper component enables users to progress sequentially through each step, ensuring navigation from one step to the next in a linear way by setting the [linear](#) property to `true`. The default value is `false` allowing navigation between any steps and vice versa.

The example demonstrates the functionality of both linear and non-linear flow in the Stepper.

APP.COMPONENT.TS

```
import { Component, ViewChild } from "@angular/core";
import { StepperComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  @ViewChild('ejStepper') stepper: StepperComponent | any;
  public updateLinear(args: any): void {
    this.stepper.linear = (/true/).test(args.currentTarget.value) ? true :
false;
    this.stepper.reset();
  };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithLabel">
  <div class="e-btn-group stepperFlow">
    <input type="radio" id="linear" name="linear" value="true"
(click)="updateLinear($event);" checked/>
    <label class="e-btn" for="linear">Linear</label>
    <input type="radio" id="nonLinear" name="linear" value="false"
(click)="updateLinear($event);" />
    <label class="e-btn" for="nonLinear">Non-Linear</label>
  </div>
  <div class="stepper-section">
    <ejs-stepper #ejStepper linear="true">
      <e-steps>
        <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
        <e-step label="Shipped" iconCss="sf-icon-transport"></e-step>
        <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
        <e-step label="Delivered" iconCss="sf-icon-success"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
```

```

pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxtT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQQAaZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAAADAAIAAIBAOc5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAWADAAMAaWADAAAAAEABAACAAMABQAAAAAAAEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBgcFBQQEAgH+CwEBAGQEBQUHBggICakJCgoKCQoICQg
HBwCfBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAd3n0BAwMGBgYI
CAMEBQYHBwKJCGSLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgSLDA0MDQ0NDQwLcwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFbAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDAwLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABZ
cfAwUhLwIH1y8PNS8CKwIPHQELwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY3OwEfAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEA5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPcBAgQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4IERESFBQUFQQDagECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEzAwIBAQIGBQMCAQQDBgZqBgoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVfHcXFxYVFBISEA8NCwo
IBQQAQOQFCAoLDQ8QEH1UFYRXAAAAAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwCTIRmMMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDagEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQADagH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAGQEBAYFBwYHCAGICAgICAgIBgcFBgQEBACAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwgICakIBwgHBwYGBQUEAwMCAQECAGMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFbQQDAwIBAQIDA
wQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBaIkCAGIEAwQFDA0SBwCAGIBAQICBgcHBxYK
CQkJCACHBgUFBAQCAQEBQIDAQwQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgEBAQECAGMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXfHcXFxcWFxYXfHYWFHYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAIIDAwUGBwgJCgICAQENAQEFawIDAQECAGMFawMEBAUDBAwMFawIBAQECAGMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICAgJBwGHBgcGBgUFBAQEBQqIAGeBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICakJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAGIwXmMXFBIICCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISIiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYNmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAAAEAAABAAAAAAABAAcAAQABAAAAAAACAACAABAAAA
AAADAACAdwABAAAAAAAEAAcAFgABAAAAAAAFaAsAHQABAAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALBIAwAwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAIAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAOAAAAA

```

```

AAAAAAAAAAAAAAAAAAAAAAAAAAYBAgEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZn
kRc2hvcHBpbnctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('true');
    font-weight: normal;
    font-style: normal;
}
.stepperWithLabel {
    margin-top: 30px;
    padding: 30px;
}
.stepperWithLabel .stepperFlow {
    margin-left: 38%
}
.stepper-section {
    margin: 50px 100px;
}
.stepperWithLabel [class^="sf-icon-"], .stepperWithLabel [class*=" sf-icon-"]
{
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperWithLabel .sf-icon-cart:before { content: "\e710"; }
.stepperWithLabel .sf-icon-transport:before { content: "\e702"; }
.stepperWithLabel .sf-icon-payment:before { content: "\e706"; }
.stepperWithLabel .sf-icon-success:before { content: "\e715"; }

```

Steps validation in Angular Stepper component

The Stepper component allows you to set the validation state for each step, displaying either a success or error icon. You can define the success state of a step by setting the `isValid` property to `true`. If set to `false`, the step will display an error state. By default, the `isValid` property is `null`.

Based on the `stepType`, the validation state icon will be displayed either as an indicator or as part of the step label/text.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithIconLabel">
  <div class="stepper">
    <ejs-stepper>
      <e-steps>
        <e-step iconCss="sf-icon-cart" [isValid]=true></e-step>
        <e-step iconCss="sf-icon-transport"></e-step>
        <e-step iconCss="sf-icon-payment" [isValid]=false></e-step>
        <e-step iconCss="sf-icon-success"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
  <div class="labelStepper">
    <ejs-stepper>
      <e-steps>
        <e-step label="Cart" [isValid]=true></e-step>
        <e-step label="Address"></e-step>
        <e-step label="Payment" [isValid]=false></e-step>
        <e-step label="Confirmation"></e-step>
      </e-steps>
    </ejs-stepper>
  </div>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAADm1heHABFgErAAABCAAAACBuYW11uF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABoC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQAiwC
3AQkBGGAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwcHBQUEAw
MBAQEBAwMEBQUHBwcICQgKCQoKCgkKCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwcFBQDQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBWYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwcGBgUEAwMBAQEBAwM
EBQYGBwcICQkJCQoKCgkJCagIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCagJCgoLDA0NDQECBAUGBwQI1foBAGQFBgcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAYAQgBaAGwArQDuAAABBz
cfAwUhLwIHly8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcFBTcfAg8BLwY3OwEfaQcVHw8/Dy8PDw4
```

```

BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPcBAgQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQEAE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WfxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCasMDQ4IERESFBQUFQQDagECagMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBggoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEHlUFRYXAAAAAQAaaaaa/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICagICagIBwYHBQYEB
AQDagEBAGMEBAQGBQcGBwgICagICagIBwYHBQYEBAQDagH+aAICagQEBAYFBwYIBwgICagICagHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCagICagICacIBgcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICagHBwCGBgUFBAQCAgEBAQEAgQEBQUGBgCHBgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCACICQgICacHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCagICQgHCacHBgYFBQQDawIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDWydAR8WDw0daQ8BKwIvAT0BLwc9AT8COWefBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCagENDAwKCggHBQEBAikCagIEAwQFDA0SBwCAGaIBAQICBgCHBxYK
CQkJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICagQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCagHBwYFBQQDagEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAIdAwUGBwgJCgICAQENAQEFaWIDAQECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICagJBwgHBgcGBgUFBAQEBOQIagEBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICakJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBFQQnEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDAwMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYSrEJwwGagIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAACAAQABAAAAAACAACACAABAAAA
AAADAACADwABAAAAAEEAACAFgABAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEEECQAAAAIAbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiW
ADAAEEECQAEAA4AmQADAAEEECQAFABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBIAHIAAYQB0AGU
AZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAA
AAAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCADXRYYW5zcG9ydC12YW4LdXNlcil1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}

.stepperWithIconLabel {
  margin-top: 30px;
  padding: 30px;
}

.stepperWithIconLabel .labelStepper{
  margin-top: 50px;
}

.stepperWithIconLabel [class^="sf-icon-"], .stepperWithIconLabel [class*="
sf-icon-"] {
  font-family: 'Default' !important;
  speak: none;

```

```

font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

.stepperWithIconLabel .sf-icon-cart:before { content: "\e710"; }
.stepperWithIconLabel .sf-icon-transport:before { content: "\e702"; }
.stepperWithIconLabel .sf-icon-payment:before { content: "\e706"; }
.stepperWithIconLabel .sf-icon-success:before { content: "\e715"; }

```

Template in Angular Stepper component

The Stepper component allows you to customize the default appearance and content of each step, creating a personalized experience for the user. You can use the [template](#) property to set the template content for the steps.

The step model and current step index are passed as `step` and `currentStep` properties in the template context for customization.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="default-stepper-section">
  <ejs-stepper id="stepperTemplate" activeStep="1" [template]="template">
    <e-steps>
      <e-step label="PowerPoint" iconCss="sf-icon-powerpoint"></e-step>
      <e-step label="Presentation" iconCss="sf-icon-projector"></e-step>
      <e-step label="Backup" iconCss="sf-icon-onedrive"></e-step>
    </e-steps>
    <ng-template #template let-data="">
      <div class="template-content">
        <span class="{{data.step.iconCss}}"></span><br>
        <span class="e-label">{{data.step.label}}</span>
      </div>
    </ng-template>
  </ejs-stepper>
</div>

```



```

    </div>
  </ng-template>
</ejs-stepper>
</div>

```

APP.COMPONENT.CSS

```

.template-content {
  background: #fff;
  width: 65px;
}

/* Stepper progressbar customization */
#stepperTemplate.e-stepper .e-stepper-progressbar {
  height: 3px;
  top: 25px;
}
#stepperTemplate.e-stepper .e-stepper-progressbar .e-progressbar-value {
  background-color: #27d96d;
}

/* Stepper status customization */
#stepperTemplate .e-step-completed * {
  color: #19cd60;
}

#stepperTemplate .e-step-inprogress * {
  color: #3479f3;
}

#stepperTemplate .e-step-notstarted * {
  color: #bdbdbd;
}

@font-face {
  font-family: 'template';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjltSfUAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbHlm39zz
MQAAAdwAAAacaGVhZCaImHMAAADQAAAAANmhoZWEIuQQGAAAArAAAACRobXR4FAAAAAAAAAAAAAUUb
G9jYQR8BVAIAAHQAAAADG1heHABFwEbAAABCAAAACBuYW1ldkXdggAAChgAAAKRcG9zdD2fuhIAAA
sMAAAAXwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABQABAAAAQAAPdGf18PPPU
ACwQAAAAAAG7KcEAAAAA4bspwQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAFAQ8ACAAAAAAAAGAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAQA AAAEAAAABA
AAAAQAAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wAAAAEABAA
AAAEAAgADAAQAAAAAAVQCAGMoA04ACAAAAAAD0wP0AB8APABcAHwA+gD+AQoBDgAAAQ8HLwY9AT8G
HwYlDwcVbJ0BPwI7AR8FNw8HLwY9AT8GHwYlDwYrAS8GPwcZHWUHF8HMzcXDwIfBz8HJzcFAz8CF
w8BHwc/By8HDwMnPWl1LwcPBxUfAQcvBCMPAic/Ay8HDwYlESERAYEHFzcZfZcnIREhJyE1IQMiAQ
ECAwQEBQUFBAUDAwICAgIDAwUEBQUFBAQDAgH+mwEBAGMEBAUFBBQFQFawMCAgILBgUFBQQEAWIBnAE
BAGMEBAUFBBQEBAMCAgICAwQEBAUFBBQEEAwIB/rkBAQMDAwUEBQUFBAQDAgEBAQECAwQEBQUFBAUD
AwMBUAEDBgYJCQsLCAxoAgQBAQQFBWgJCwsLCwkJBgYDAQFECAgICakIBIEBAQEDBQcICgoLDAoJC
QcFAwEBAwUHCQkKDAoJCQl4BQMCAQMFBWgKCGwLCgoIBwUDAQICPwEJCQoMBwCGCGQFAwIBAQQFBw
gJCwsLCwkJBgUEAsb89j8BZ8sr+gX7K8sBZ/x4DwOm/FoB7QUFBAQDAgEBAQECAwQEBQUFBAUDAwM
BAQEBAwMDBQqzBQUEBAMCAQEBAQIDBAQFBQUGCwICAgMEBARgBQUEAwMDAQEBAQMDAwQFBQUFBAMD
AwEBAQEDAwMEBUsFBAQEAWICAgIDBAQEBQUFBAQDAgEBAQIDBAQFBQUGCgoIBwUDAQNNBAsLCwsJC

```

[illegible]

```
width: 75%;
margin: 0px auto;
min-width: 85px;
padding: 25px 0;
}
```

Tooltip in Angular Stepper component

The Stepper component supports tooltip to show additional information in the steps by setting the `showTooltip` property to `true`.

The tooltip appears when the user hovers over the step, providing the information such as the label or text. By default, the `showTooltip` property is `false`.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperTooltip">
  <ejs-stepper showTooltip="true">
    <e-steps>
      <e-step label="Cart" iconCss="sf-icon-cart"></e-step>
      <e-step label="Delivery Address" iconCss="sf-icon-transport"></e-step>
      <e-step label="Payment" iconCss="sf-icon-payment"></e-step>
      <e-step label="Confirmation" iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMjlvSgcAAAEoAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
```

pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxtT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAAADAAIAAIABoC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAEQaiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBgcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwCfBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAd3n0BAwMGBgYI
CAMEBQYHBwKJCGsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLcwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFbAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFbGcHCAkJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABZ
cfAwUhLwIH1y8PNS8CKwIPHQELwEjDwE1LwMjNz0BJZcfBTcfAg8BLwY3OwEfAQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgyEBQMEA5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPcBAgQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFxcXFhQUExEQDw0LCgGFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4IERESFBQUFQQDAGECAGMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEzAwIBAQIGBQMCAQQDBgZqBgoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEAE8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQAQQFCAoLDQ8QEH1UFYRXAAAAAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwCTIRmMMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAGEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBADAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAGQEBAYFBwYHCAGICAgICAgIBgcFBgQEBAlCAsH6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwgICakIBwgHBwYGBQUEAwMCAQECAGMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFbQQDAwIBAQIDA
wQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0daQ8BKwIvAT0BLwC9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQEBAikCAGIEAwQFDA0SBwCAGIBAQICBgcHBxYK
CQkJCACHBgUFBAQCAQEBAQIDAQWQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXfHcXFxcWFxYXfHYWFHYVFRUVFAQCAQICBAUGCAgJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAIIDAwUGBwgJCgICAQENAQEFawIDAQECAGMFAwMEBAUDBAwMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICAgJBwGHBgcGBgUFBAQEBOQiAgEBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICakJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUFBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGAgIwXmMXFBIICCsQKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAAAEAAABAAAAAAABAAcAAQABAAAAAAACAACAABAAAA
AAADAACAdwABAAAAAAAEAAcAFgABAAAAAAAFaAsAHQABAAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAwAwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhckRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW51
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAoAAAAA

```

AAAAAAAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.stepperTooltip {
    margin-top: 30px;
    padding: 30px;
}
.stepperTooltip [class^="sf-icon-"], .stepperTooltip [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperTooltip .sf-icon-cart:before { content: "\e710"; }
.stepperTooltip .sf-icon-transport:before { content: "\e702"; }
.stepperTooltip .sf-icon-payment:before { content: "\e706"; }
.stepperTooltip .sf-icon-success:before { content: "\e715"; }

```

Tooltip template

You can use the [tooltipTemplate](#) property to specify a custom template for the tooltips, providing detailed information about the steps.

When hovering over the step, the current step model is passed in the template context, allowing you to include dynamic information about the step.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="stepperTooltip">

```

```

<ejs-stepper showTooltip=true tooltipTemplate='<span class="template-
content"><span class="stepper-icon ${value.iconCss}"></span><span
class="step-label">${value.text}</span></span>'>
  <e-steps>
    <e-step text="Review your cart items" label="Cart" iconCss="sf-icon-
cart"></e-step>
    <e-step text="Enter your delivery address" label="Delivery Address"
iconCss="sf-icon-transport"></e-step>
    <e-step text="Complete your purchase securely" label="Payment"
iconCss="sf-icon-payment"></e-step>
    <e-step text="Verify your order details" label="Confirmation"
iconCss="sf-icon-success"></e-step>
  </e-steps>
</ejs-stepper>
</div>

```

APP.COMPONENT.CSS

```

.template-content {
  display: flex;
  align-items: center;
}

.e-stepper-tooltip {
  border-radius: 5px;
}

.e-tip-content {
  padding: 7px;
}

.step-label {
  margin-left: 7px;
  font-weight: bold;
}

.stepperTooltip {
  margin-top: 30px;
  padding: 30px;
}

@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAAABA
AAAAQAAAAEAAAAAAAgAAAAAMAAAUAMAAQAAABQABABKAAAADAIAIAIABOcC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAAAAEQAiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwcHBQUEAw

```

```
font-weight: normal;
font-style: normal;
}
```

```

font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}

.sf-icon-cart:before { content: "\e710"; }
.sf-icon-transport:before { content: "\e702"; }
.sf-icon-payment:before { content: "\e706"; }
.sf-icon-success:before { content: "\e715"; }

```

Animation in Angular Stepper component

The Stepper progress state can be animated, smoothly transitioning from one step to another. You can customize the animation's **duration** and **delay** by using the [animation](#) property.

You can disable the animation by setting the [enable](#) property to **false**. By default, the value is **true**.

| Fields | Type | Description |

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

| [duration](#) | **number** | Specifies the duration of the animated transition for each step. The default value is **2000** milliseconds. |

| [delay](#) | **number** | Specifies the delay to initiate the animated transition for each step in milliseconds. The default value is **0**. |

The example demonstrates the animation **duration** and **delay** settings for the Stepper.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
import { StepperAnimationSettingsModel } from "@syncfusion/ej2-angular-navigations";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public animateStepper: StepperAnimationSettingsModel = {enable: true,
    duration: 2000, delay: 500};
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


APP.COMPONENT.HTML

```
<div class="stepperAnimation">
  <ejs-stepper [animation]="animateStepper">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

Globalization in Angular Stepper component

Localization

The Localization library allows you to localize the default text content of the Stepper. You can change the static text content used for the **optional** property to other cultures (Arabic, Deutsch, French, etc.) by defining the **locale** value and its translation object.

| Locale key | en-US (default) |

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

| optional | Optional |

In this example, the **French** culture is set to Stepper and the default text is updated with the content defined by the locale key.

APP.COMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { L10n } from '@syncfusion/ej2-base';
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  ngOnInit() {
    // Load French culture for stepper optional property
    L10n.load({
      'fr-BE': {
        'stepper': {
          'optional': "facultatif"
        }
      }
    });
  };
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperWithIcon">
  <ejs-stepper locale='fr-BE'>
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment" optional="true"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQAGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUblAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYwlluf5THQAACTgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAAQAArxT6wV8PPPU
ACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAgAAAAMAAAUAAMAAQAAABQABABKAAAADAAIAAIABOC5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQaiwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz800wEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwCHBQUEAw
MBAQEBAwMEBQUHBwCICQgKCQoKCgkKCAgIBGcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwCFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAd3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCGsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCakJCQoLCgkJCQkIBwCGBgUEAwMBAQEBAwM
EBQYGBwCICQkJCQoKCgkKCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1foBAGQFBgCH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAYAqgBaAGwArQDuAAABBz
cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcFBTcfAg8BLwY30wEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCfHITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgYGBxwCAwIBFYGBAgFBgIWA
gQHCPcBAGQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEa8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WfxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4IERESFBQUFQQDAgECAGMEBAUG
BgYIBwgJCQoKCwsLDawMDQ0ODQ4PDgEZawIBAQIGBQMQAQQDBgZqBgOHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAYGawMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXfXyYVFBISEA8NCwo
IBQQBAQQFCAoLDQ8QEhIUFRYXAAAAAAQAAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEB
AQDAgEBAGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH+aAICAgQEBAYFBwYIBwgICAgICAgHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCAGICAgICAcIBGcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBgICakIBwGHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
```

```

wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAAANKA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY
8HDWYdAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWEfBj8HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQEBAikCAGIEAwQFDA0SBwcGAGIBAQICBgcHBxYK
CQkJCAcHBgUFBAMCAQEBAQIDAwQFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhiHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFXcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCAGJCgsLDA0MDQ0NDBk2EQYGGgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBAgQCAQEBAwKRNRIHBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAIIDAwUGBwgJCgICAQENAQEFawIDAgECAGMFawMEBAUDBAwFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAYDIgICAQECAiICBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAW
QEAgEBAQICBAMFBQUGBwCICAgJBwgHBgcGBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQI
DAwUFBgYICAKJCwsMKSckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUFBQqNEAkKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYS EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISiiIiIjIkJAPRwB8AQmCQMBARQuNgsMDgcIJCYnmyZOTyemJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAA
AAADAACADwABAAAAAEEAAcAFgABAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABA
AAAAAALABIAWwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAA
AAAAAAAAAAAAAAAAAAAAAYBAGEDAQQBBQEGAQCADXRYYW5zcG9ydC12YW4ldXNlcil1tb2RpZn
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.stepperWithIcon {
    margin-top: 30px;
    padding: 30px;
}
.stepperWithIcon [class^="sf-icon-"], .stepperWithIcon [class*=" sf-icon-"] {
    font-family: 'Default' !important;
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    line-height: inherit;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.stepperWithIcon .sf-icon-cart:before { content: "\e710"; }
.stepperWithIcon .sf-icon-transport:before { content: "\e702"; }
.stepperWithIcon .sf-icon-payment:before { content: "\e706"; }
.stepperWithIcon .sf-icon-success:before { content: "\e715"; }

```

RTL

RTL provides an option to switch the text direction and layout of the Stepper component from right to left by setting the [enableRtl](#) property to `true`.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
```

```
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepperIcon">
  <ejs-stepper enableRtl=true>
    <e-steps>
      <e-step iconCss="sf-icon-cart"></e-step>
      <e-step iconCss="sf-icon-transport"></e-step>
      <e-step iconCss="sf-icon-payment"></e-step>
      <e-step iconCss="sf-icon-success"></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

APP.COMPONENT.CSS

```
@font-face {
  font-family: 'Default';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+
pwAAAggAAAJQaGVhZCYp2+EAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYb
G9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACtgAAAIlcG9zdJ8LuoMAAA
0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8PPPU
ACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAAAAAAAgAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wbnCOcQ5xX//wA
A5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAAAAAAEQAIwC
3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8NAR8FFSM1J
xEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAkIBwcHBQUEAw
MBAQEBAwMEBQUHBwcICQgKCQoKCgkKCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICakJCgoKCQoICQg
HBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAwD3n0BAwMGBgYI
CAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAYHBQMD/beAAwQFB
QcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCAcHBgYFBAMDAQEBAQMDBAUGBg
cHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwcGBgUEAwMBAQEBAwM
EBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQIGehYJBKYp/10ICAcG
BQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1foBAGQFBGcH/iwNDawLC
woJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAYAqgBaAGwArQDuAAABZ
```

```

cfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY3OwEfaQcVHw8/Dy8PDw4
BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQCFhITE+wQDw8PDg4ODg0N
DQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgYGBxwCAwIBFQYGBAgFBgIWA
gQHCPCBAgQGBgKCgMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQE8PDg
0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBY
WFXcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4IERESFBQUFQQDagECagMEBAUG
BgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZqBgoHBQMDMAMHBwMWAQICBQYKC
hYCB1wICBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBAQE8PDg0MCwoKCAYGAwMBAQMDBg
YICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0PEBESFBUVFhcXFxYVFBISEA8NCwo
IBQQAQQFCAoLDQ8QEHlUFYRYAAAAAQAQAAAAA/QDRwA/AH8AhwCRAAABFR8OPw49AS8NKwEPDQUV
Hw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgMEBAQGBQcGBwgICagICagIBwYHBQYEB
AQDagEBAGMEBAQGBQcGBwgICagICagIBwYHBQYEBAQDagH+aAICagQEBAYFBwYIBwgICagICagHBg
cFBgQEBAMCAQECAwQEBAYFBwYHCagICagICacIBgcFBgQEBaICAsh6jAFKjPpu/Z3NwgJZ/dzDAf8
AAQkICagHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgcHBwgICakIBwgHBwYGBQUEAwMCAQECAwMEBQUG
BgCHCAcICQgICACHBwYGBQUEBAICAQEBAQICBAQFBQYGBwCHCAgICQgHCAcHBgYFBQQDAwIBAQIDA
wQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAAAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy
8HDwYdAR8WDw0daQ8BKwIvAT0BLwc9AT8COWefBj8HLxc/DTU/AwEfdjsBPxEvFiMPFR8BEw8CFR8
HMz8HNS8GIw8ELwQrAQ8BAGoCagENDAwKCgGHBQEBaIkCagIEAwQFDA0SBwcGagIBAQICBgcHBxYK
CQkJCAcHBgUFBAQCAQEBAQIDAQWFBQYGBwCPEQECAhUCAQINDAsLCQgHBQICKQICagQDBAULdhIHB
wYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQDagEBAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFh
YWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQICBAUGCagJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0
NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRIHBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJ
BwYCOAIBAiIDAwUGBwgJCgICAQENAQEFawIDagECagMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHC
AgICQgHBwCGBgYFBQQEBAyDIgICAQECAiICBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAW
QEAgEBAQICBAMFBQUGBwCICagJBwgHBgcGBgUFBAQEBOQIagEBAf6RDAsLCQkICAYGBQUdAwIBAQI
DAwUFBgYICakJCwsMKsckIiAeGxoYFhQTERAPDQwLCgkIDxsJBQUBQqNEakKCwwNDxARExQWGBob
HiAiJCcCoAMDawQECA8XPRcKCgUPFz0REakIBAMDAwMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDN
AA0AAABDwQvAw8EHwQ/ETUnIw8LAYsEJwwGagIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFg
sCAWMhISIiIiIjIkJAPRwB8AQmQMBARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4
RERITFBUVKy0tFgAAABIA3gABAAAAAABAAAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAA
AAADAACADwABAAAAAABAAcAFgABAAAAAABFAAsAHQABAAAAAABGAACAKAABAAAAAABKAwALwABA
AAAAAALABIAwADAAEECQAAAAIAbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiW
ADAAEECQAEAA4AmQADAAEECQAFABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQ
BIyBEZWZhdWx0UmVndWxhcRlZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5l
cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIBEA
GUAZgBhAHUAbAB0AFIAZQBnAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQ
ByAHMAaQBvAG4AIAAxAAC4AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAAYQB0AGU
AZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBk
AGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaA
AAAAAAAAAAAAAAAAAAAAAAYBAgEDAQBBQEGAQcADXRyYw5zcG9ydC12YW4ldXNlci1tb2Rpb2N
kRc2hvcHBpbmctY2FydF8wMS0Lc3BlbmQtbW9uZXXkFY2hly2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
 stepperIcon {
margin-top: 30px;
padding: 30px;
}
 stepperIcon [class^="sf-icon-"], .stepperIcon [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;

```

```

-moz-osx-font-smoothing: grayscale;
}
.stepperIcon .sf-icon-cart:before { content: "\e710"; }
.stepperIcon .sf-icon-transport:before { content: "\e702"; }
.stepperIcon .sf-icon-payment:before { content: "\e706"; }
.stepperIcon .sf-icon-success:before { content: "\e715"; }

```

Accessibility in Angular Stepper component

The Stepper 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 Stepper 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 following ARIA attributes are used in the Stepper component:

Attributes	Purpose
aria-label	Attribute provide a descriptive text that labels the interactive element for accessibility.
aria-current	Attribute denotes the currently active step in the Stepper.
aria-disabled	Indicates when the step is disabled and cannot be interacted.

Keyboard interaction

The following keyboard shortcuts are supported by the Stepper component.

Press	To do this
Up Arrow	Focuses the previous step in a vertical Stepper.
Down Arrow	Focuses the next step in a vertical Stepper.
Left Arrow	Focuses the previous step in a horizontal Stepper.
Right Arrow	Focuses the next step in a horizontal Stepper.
Home	Focuses on the first step of the Stepper.
End	Focuses on the last step of the Stepper.
Enter / Space	Activates the currently focused step.

Ensuring accessibility

The Stepper component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Stepper component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Stepper component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Events in Angular Stepper component

This section describes the Stepper events that will be triggered when an appropriate actions are performed. The following events are available in the Stepper component.

created

The Stepper component triggers the [created](#) event when the component rendering is completed.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public created(): void {
    //Your required action here
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepper">
  <ejs-stepper (created)="created()">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

stepChanged

The Stepper component triggers the [stepChanged](#) event after the active step is changed.

APP.COMPONENT.TS

```
import { Component } from "@angular/core";
import { StepperChangedEventArgs } from "@syncfusion/ej2-navigations";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public stepChanged(args: StepperChangedEventArgs): void {
    alert("Step changed from "+args.previousStep + " to " + args.activeStep)
  }
}
```


MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepper">
  <ejs-stepper (stepChanged)="stepChanged($event)">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

[stepChanging](#)

The Stepper component triggers the [stepChanging](#) event before the active step change.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { StepperChangingEventArgs } from '@syncfusion/ej2-navigations';
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public stepChanging(args: StepperChangingEventArgs): void {
    alert("Step changing from "+args.previousStep + " to " + args.activeStep)
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepper">
  <ejs-stepper (stepChanging)="stepChanging($event)">
    <e-steps>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

```

        <e-step></e-step>
        <e-step></e-step>
        <e-step></e-step>
    </e-steps>
</ejs-stepper>
</div>

```

stepClick

The Stepper component triggers the [stepClick](#) event when the step is clicked.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";
import { StepperClickEventArgs } from "@syncfusion/ej2-navigations";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public stepClick(args: StepperClickEventArgs): void {
    //Your required action here
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

APP.COMPONENT.HTML

```

<div class="stepper">
  <ejs-stepper (stepClick)="stepClick($event)">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>

```

beforeStepRender

The Stepper component triggers the [beforeStepRenderLink to the Video](#) event before rendering each step.

APP.COMPONENT.TS

```

import { Component } from "@angular/core";

```

```
import { StepperRenderingEventArgs } from "@syncfusion/ej2-navigations";
@Component({
  imports: [ StepperAllModule, StepperModule ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public beforeStepRender(args: StepperRenderingEventArgs): void {
    //Your required action here
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

APP.COMPONENT.HTML

```
<div class="stepper">
  <ejs-stepper (beforeStepRender)="beforeStepRender($event)">
    <e-steps>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
      <e-step></e-step>
    </e-steps>
  </ejs-stepper>
</div>
```

Stock Chart

Getting started with Angular Stock chart component

This section explains you the steps required to create a simple StockChart and demonstrate the basic usage of the StockChart component in an Angular environment.

To get start quickly with Angular Stock Charts using CLI and Schematics, you can check on this video:

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
```

```
npm install -g @angular/cli
```

```
`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`
```

Installing Syncfusion StockChart package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-charts](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-charts --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-charts@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-charts@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-charts:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering StockChart Module

Import Chart module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-charts` [`src/app/app.module.ts`].

```
`typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the StockChartModule for the StockChart component
import { StockChartModule } from '@syncfusion/ej2-angular-charts';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of StockChartModule into NgModule
  imports: [ BrowserModule, StockChartModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

- Modify the template in `app.component.ts` file to render the `ej2-angular-charts` component

[src/app/app.component.ts].

```
`javascript
import { Component, ViewEncapsulation } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the Charts component
  template: <ejs-stockchart id='chart-container'></ejs-stockchart>,
  encapsulation: ViewEncapsulation.None
})
export class AppComponent { }
```

<!-- markdownlint-disable MD033 -->

Now use the `<code>app-container</code>` in the index.html instead of default one.

```
<app-container></app-container>
```

- Now run the application in the browser using the below command.

npm start

Module Injection

Chart component are segregated into individual feature-wise modules. In order to use a particular feature, you need to inject its feature service in the AppModule. In the current application, we are going to modify the above basic chart to visualize sales stock value of a company.

For this application we are going to use candle series, tooltip, data label, datetime axis and legend feature of the chart. Please find relevant feature service name and description as follows.

- **CandleSeriesService** - Inject this provider to use candle series.
- **LegendService** - Inject this provider to use legend feature.
- **TooltipService** - Inject this provider to use tooltip feature.
- **DataLabelService** - Inject this provider to use datalabel feature.
- **DateTimeService** - Inject this provider to use datetime feature.

These modules should be injected to the provider section as follows,

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ChartComponent } from '@syncfusion/ej2-angular-charts';
import { DateTimeService, LegendService, TooltipService } from '@syncfusion/ej2-angular-charts';
import { DataLabelService, CandleSeriesService } from '@syncfusion/ej2-angular-charts';
@NgModule({
  imports: [
    BrowserModule,
  ],
  declarations: [AppComponent, ChartComponent],
  bootstrap: [AppComponent],
  providers: [ DateTimeService, LegendService, TooltipService, DataLabelService, CandleSeriesService ]
})
```

Populate Chart with Data

This section explains how to plot below JSON data to the chart.

```
`javascript
export class AppComponent implements OnInit {
```

```
public chartData: Object[];
ngOnInit(): void {
  // Data for chart series
  this.chartData = [{
    "x": new Date('2012-04-02T00:00:00.000Z'),
    "open": 320.705719,
    "high": 324.074066,
    "low": 317.737732,
    "close": 323.783783,
    "volume": 45638000
  }, {
    "x": new Date('2012-04-03T00:00:00.000Z'),
    "open": 323.028015,
    "high": 324.299286,
    "low": 319.639648,
    "close": 321.631622,
    "volume": 40857000
  }, {
    "x": new Date('2012-04-04T00:00:00.000Z'),
    "open": 319.544556,
    "high": 319.819824,
    "low": 315.865875,
    "close": 317.892883,
    "volume": 32519000
  }, {
    "x": new Date('2012-04-05T00:00:00.000Z'),
    "open": 316.436432,
    "high": 318.533539,
    "low": 314.599609,
    "close": 316.476471,
    "volume": 46327000
  }]
}
```

```
}
`
```

Add a series object to the chart by using [series](#) property and then set the JSON data to [dataSource](#) property.

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

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartAllModule, StockChartAllModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartAllModule, StockChartAllModule
  ],
  providers: [ CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container" >
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='stockchartData' type='Candle'
xName='date' High='high' Low='low' Open='open' Close ='close'
Name='Apple'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public stockchartData?: Object[];
  ngOnInit(): void {
    // Title for stock chart
    this.stockchartData = chartData;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add Stock Chart Title

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

APP.COMPONENT.TS


```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartAllModule, StockChartAllModule } from '@syncfusion/ej2-angular-charts'
import { CategoryService, LegendService, TooltipService, DataLabelService,
LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartAllModule, StockChartAllModule
  ],
  providers: [ CategoryService, LegendService, TooltipService,
DataLabelService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[title]='title'>
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='stockchartData' type='Candle'
xName='date' High='high' Low='low' Open='open' Close ='close'
Name='Apple'></e-stockchart-series>
  </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public stockchartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    // Title for chart
    this.title = 'AAPL Stock Price';
    this.stockchartData = chartData;
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add Stock Chart Crosshair

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'

```

```
import { CategoryService, LegendService, CandleSeriesService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, LegendService, CandleSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [title]='title' [crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='stockchartData' type='Candle'
xName='date' High='high' Low='low' Open='open' Close ='close'
Name='Apple'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public stockchartData?: Object[];
  public title?: string;
  crosshair?: Object;
  ngOnInit(): void {
    // Title for chart
    this.title = 'AAPL Historical';
    this.crosshair = { enable: true };
    this.stockchartData = chartData;
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

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 `CrosshairService` and `TooltipService` into the `NgModule.providers`.

Trackball can be enabled by setting the `enable` property of the crosshair to true and `shared` property in `tooltip` to true in chart.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
```

```

import { LegendService, DataLabelService, TooltipService, CrosshairService }
  from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    DataLabelService,
    TooltipService, CrosshairService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair' [tooltip]='tooltip'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y' name='John' width=2 [marker]='marker'></e-stockchart-
series>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y1' name='Andrew' width=2 [marker]='marker'></e-stockchart-
series>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y2' name='Thomas' width=2 [marker]='marker'></e-stockchart-
series>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y3' name='Mark' width=2 [marker]='marker'></e-stockchart-
series>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y4' name='William' width=2 [marker]='marker'></e-stockchart-
series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public crosshair?: Object;
  public title?: string;
  public tooltip?: Object;
  public marker?: Object;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = [
      { x: new Date(2000, 2, 11), y: 15, y1: 39, y2: 60, y3: 75, y4: 85
    },
      { x: new Date(2000, 9, 14), y: 20, y1: 30, y2: 55, y3: 75, y4: 83
    },
      { x: new Date(2001, 2, 11), y: 25, y1: 28, y2: 48, y3: 68, y4: 85
    },
      { x: new Date(2001, 9, 16), y: 21, y1: 35, y2: 57, y3: 75, y4: 87
    },
      { x: new Date(2002, 2, 7), y: 13, y1: 39, y2: 62, y3: 71, y4: 82
    },
      { x: new Date(2002, 9, 7), y: 18, y1: 41, y2: 64, y3: 69, y4: 74
    },
    ],
  }
}

```

```

    { x: new Date(2003, 2, 11), y: 24, y1: 45, y2: 57, y3: 81, y4: 73
    },
    { x: new Date(2003, 9, 14), y: 23, y1: 48, y2: 53, y3: 84, y4: 75
    },
    { x: new Date(2004, 2, 6), y: 19, y1: 54, y2: 63, y3: 85, y4: 73
    },
    { x: new Date(2004, 9, 6), y: 31, y1: 55, y2: 50, y3: 87, y4: 60
    },
    { x: new Date(2005, 2, 11), y: 39, y1: 57, y2: 66, y3: 75, y4: 48
    },
    { x: new Date(2005, 9, 11), y: 50, y1: 60, y2: 65, y3: 70, y4: 55
    },
    { x: new Date(2006, 2, 11), y: 24, y1: 60, y2: 79, y3: 85, y4: 40
    }
  ];
  this.primaryXAxis = {
    title: 'Years',
    minimum: new Date(2000, 1, 1), maximum: new Date(2006, 2, 11),
    intervalType: 'Years',
    valueType: 'DateTime',
  };
  this.tooltip = { enable: true, shared: true, format: '${series.name}
: ${point.x} : ${point.y}' };
  this.crosshair = { enable: true, lineType: 'Vertical' };
  this.marker = { visible: true };
  this.title = 'Average Sales per Person';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Stock Chart](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Stock Chart example](#) that shows you how to present and manipulate data.

<!-- markdownlint-disable MD036 -->

Working with data in Angular Stock chart component

Stock 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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService} from '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [crosshair]='crosshair'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='stockchartData' type='Candle'
          xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public stockchartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  ngOnInit(): void {
    this.stockchartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: {enable:true}
    };
    this.primaryYAxis = {
      majorTickLines: { color: 'transparent', width: 0 },
      crosshairTooltip: {enable:true}
    };
    this.crosshair= {
      enable: true
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { DataManager, Query } from '@syncfusion/ej2-data';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="stockChartSpline"
[enablePeriodSelector]="enable"
[chartArea]="chartArea" [primaryXAxis]="primaryXAxis"
[primaryYAxis]="primaryYAxis" [seriesType]="seriesType"
[indicatorType]="indicatorType">
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]="dataManager" [query]="query"
type="Spline" xName="ShippedDate"
yName="Freight" >
    </e-stockchart-series>
  </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  chartArea: any;
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

  }
  public dataManager: DataManager = new DataManager({
    url: "https://ej2services.syncfusion.com/production/web-
services/api/Orders"
  });
  public query: Query = new Query()
    .take(5)
    .where("Estimate", "lessThan", 50, false);
  public primaryXAxis: Object = {
    valueType: "DateTime",
    crosshairTooltip: { enable: true },
    majorGridLines: { width: 0 }
  };
  public primaryYAxis: Object = {
    lineStyle: { width: 0 },
    majorTickLines: { width: 0 }
  };
  public seriesType: string[] = [];
  public indicatorType: string[] = [];
  public periods: object[] = [];
  public enable: boolean = false;
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Series Types](#)

Chart dimensions in Angular Stock chart component

Size for Container

Stock Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, DateTimeService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, LineSeriesService, DateTimeService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' >
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Size for Stock 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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, DateTimeService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, LineSeriesService, DateTimeService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' width='650' height='350'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

MAIN.TS


```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

In Percentage

By setting value in percentage, Stock Chart gets its dimension with respect to its container. For example, when the height is '50%', Stock Chart renders to half of the container height.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { CategoryService, LineSeriesService, DateTimeService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, LineSeriesService, DateTimeService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' width='80%' height='90%'>
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
  </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Note: When you do not specify the size, it takes 450px as the height and window size as its width.

<!-- markdownlint-disable MD036 -->

Axis types in Angular Stock chart component

DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format. To use DateTime axis, set the [valueType](#) of axis to `DateTime`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
          xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
      </e-stockchart-series-collection>
    </e-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

DateTimeCategory axis

DateTimeCategory axis in the stock chart is used to display only business days. To use DateTimeCategory axis, set the [valueType](#) of axis to `DateTimeCategory`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series2 } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [crosshair]='crosshair' [tooltip]='tooltip'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y'></e-stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public tooltip?: Object;
  public crosshair?: Object;
  ngOnInit(): void {
    this.chartData = series2;
    this.primaryXAxis = {
      valueType: 'DateTimeCategory',
      majorGridLines: { width: 0 },
      crosshairTooltip: { enable: true }
    };
    this.tooltip = { enable: true, header: 'AAPL Stock Price' };
    this.crosshair = { enable: true };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Logarithmic axis

<!-- 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). To use Logarithmic axis, set the [valueType](#) of axis to **Logarithmic**.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
          xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.primaryYAxis = {
      valueType: 'Logarithmic'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See also

- [Axis Customization](#)

Axis customization in Angular Stock chart component

Axis Crossing

An axis can be positioned in the Stock 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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart> <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='chartData' type='Candle'
    xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      crossesAt: 90
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      title: 'AAPL Historical',
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
 StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
    StripLineService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
          xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      //Tick lines customization
      majorTickLines : {
        color : 'blue',
        width : 5
      },
      minorTickLines : {
        color : 'red',
        width : 0
      }
    };
    this.primaryYAxis = {
      //Tick lines customization
      majorTickLines : {
        color : 'blue',
        width : 5
      },
      minorTickLines : {
```

```

        color : 'red',
        width : 0
    }
};
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
  </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      //Tick lines customization

```



```

        majorGridLines : {
            color : 'blue',
            width : 5
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    };
    this.primaryYAxis = {
        //Tick lines customization
        majorGridLines : {
            color : 'blue',
            width : 5
        },
        minorGridLines : {
            color : 'red',
            width : 0
        }
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the Stock Chart. Series can be associated with this [axis](#), by mapping with axis's unique name.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, DateTimeService,
 LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, ColumnSeriesService, DateTimeService,
    LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-stockchart-axes>

```

```

        <e-stockchart-axis rowIndex=1 name='yAxis1'
opposedPosition='true' title='Temperature (Celsius)'
[majorGridLines]='majorGridLines' labelFormat='{value}°C'
        [minimum]='24' [maximum]='36' [interval]='2'
[lineStyle]='lineStyle'>
        </e-stockchart-axis>
    </e-stockchart-axes>
    <e-stockchart-rows>
        <e-stockchart-row height='20%'></e-stockchart-row>
        <e-stockchart-row height='20%'></e-stockchart-row>
    </e-stockchart-rows>
    <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Column'
xName='x' yName='y' name='Germany'></e-stockchart-series>
        <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y1' name='Japan' yAxisName='yAxis1' [marker]='marker'></e-
stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public majorGridLines?: Object;
        public primaryYAxis?: Object;
        public lineStyle?: Object;
        public marker?: Object;
        public rows?: Object;
        ngOnInit(): void {
            this.chartData = [
                { x: new Date(2000, 2, 11), y: 15, y1: 29 },
                { x: new Date(2000, 9, 14), y: 20, y1: 30 },
                { x: new Date(2001, 2, 11), y: 25, y1: 28 },
                { x: new Date(2001, 9, 16), y: 21, y1: 35 },
                { x: new Date(2002, 2, 7), y: 13, y1: 29 },
                { x: new Date(2002, 9, 7), y: 18, y1: 31 },
                { x: new Date(2003, 2, 11), y: 24, y1: 25 },
                { x: new Date(2003, 9, 14), y: 23, y1: 28 },
                { x: new Date(2004, 2, 6), y: 19, y1: 24 },
                { x: new Date(2004, 9, 6), y: 31, y1: 25 },
                { x: new Date(2005, 2, 11), y: 39, y1: 27 },
                { x: new Date(2005, 9, 11), y: 30, y1: 30 },
                { x: new Date(2006, 2, 11), y: 24, y1: 20 }
            ];
            this.primaryXAxis = {
                title: 'Months',
                valueType: 'DateTime',
                interval: 1
            };
            this.primaryYAxis = {
                minimum: 0, maximum: 90, interval: 20,
                lineStyle: { width: 0 },
                title: 'Temperature (Fahrenheit)',
                labelFormat: '{value}°F',
                span: 2
            };
        }
    }

```

```

        this.majorGridLines = { width: 0};
        this.lineStyle = { width: 0};
        this.marker = {
            visible: true, width: 10, height: 10, border: { width: 2, color:
'#F8AB1D' }
        }
        this.title = 'Multiple Axis';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
  </e-stockchart-series-collection>
</e-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  ngOnInit(): void {
    this.chartData = chartData;
  }
}

```

```

        this.title = 'Efficiency of oil-fired power production';
        this.primaryXAxis = {
            valueType: 'DateTime',
            isInversed: true
        };
        this.primaryYAxis = {
            isInversed: true
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Opposed Position

To place an axis opposite from its original position, set [opposedPosition](#) property of the axis to true.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts';
import { DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule, StockChartAllModule, ChartAllModule
    ],
    providers: [ DateTimeService, LineSeriesService, DateTimeCategoryService,
StripLineService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public primaryYAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    ngOnInit(): void {
        this.chartData = chartData;
        this.title = 'Efficiency of oil-fired power production';
        this.primaryXAxis = {

```

```

        valueType: 'DateTime',
        opposedPosition: true
    };
    this.primaryYAxis = {
        opposedPosition: true
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Series types in Angular Stock chart component

Essential JS 2 StockChart supports 7 major types of series namely **Line**, **Spline**, **Area**, **Hilo**, **HiloOpenClose**, **Hollow Candle** and **Candle**. By using the series dropdown button you can navigate between the above listed series types.

<!-- markdownlint-disable MD036 -->

Line

To render a line series, use series **type** as **Line** and inject **LineSeriesService** into the **@NgModule.providers**.

Spline

To render a spline series, use series **type** as **Spline** and inject **SplineSeriesService** into the **@NgModule.providers**.

Area

To render a area series, use series **type** as **Area** and inject **AreaSeriesService** into the **@NgModule.providers**.

Hilo

To render a hilo series, use series **type** as **Hilo** and inject **HiloSeries** into the **@NgModule.providers**.

HiloOpenClose

To render a hiloOpenClose series, use series **type** as **HiloOpenClose** and inject **HiloOpenCloseSeries** into the **@NgModule.providers**.

HollowCandle

To render a hollowcandle series, use series **type** as **Candle** and set **enableSolidCandle** as false. Now inject **CandleSeries** into the **@NgModule.providers**.

Candle

To render a candle series, use series **type** as **Candle** and inject **CandleSeries** into the **@NgModule.providers**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, ChartAllModule, StockChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { LineSeriesService, SplineSeriesService, StepLineSeriesService,
CategoryService, ParetoSeriesService, ColumnSeriesService,
    SplineAreaSeriesService, MultiColoredLineSeriesService, TooltipService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ LineSeriesService, SplineSeriesService, StepLineSeriesService,
CategoryService, SplineAreaSeriesService, ParetoSeriesService,
ColumnSeriesService,
    MultiColoredLineSeriesService, TooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  primaryXAxis: any;
  primaryYAxis: any;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Trend lines in Angular Stock chart component

Trendlines are used to show the direction and speed of price.

StockChart supports 6 types of trendlines namely

Linear, Exponential, Logarithmic, Polynomial, Power, Moving Average. By using trendline dropdown button you can add or remove the required trendline type.

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`.

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`.

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`.

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`.

`polynomialOrder` used to define the polynomial value.

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`.

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`.

`period` property defines the period to find the moving average.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartAllModule, RangeNavigatorAllModule, StockChartAllModule } from '@syncfusion/ej2-angular-charts'
import { ScatterSeriesService, LineSeriesService, DateTimeService, TrendlinesService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    StockChartAllModule, RangeNavigatorAllModule, ChartAllModule,
  ],
  providers: [ ScatterSeriesService, LineSeriesService, DateTimeService, TrendlinesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container" [title]='title'>
    <e-stockchart-series-collection>
```

```

        <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ><e-trendlines>
    <e-trendline type='MovingAverage' width=3 name='Trends'
fill='red'>
        </e-trendline>
    </e-trendlines></e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        public chartData?: Object[];
        public title?: string;
        ngOnInit(): void {
            this.chartData = chartData;
            this.title = 'Efficiency of oil-fired power production';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Technical indicators in Angular Stock chart component

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

StockChart supports 10 types of technical indicators namely Accumulation Distribution, ATR, EMA, SMA, TMA, Momentum, MACD, RSI, Stochastic, Bollinger Band. By using indicator dropdown box you can add and remove the required indicators types.

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 AccumulationDistributionIndicatorService into the @NgModule.providers.

To calculate the signal line [volume](#) field is additionally added with [dataSource](#).

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 AtrIndicatorService into the @NgModule.providers.

Bollinger Band

A StockChart 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 `BollingerBandsService` into the `@NgModule.providers`.

Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine). Bollinger Band default values of the [period](#) is 14 and [standardDeviations](#) is 2.

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 `EMAIndicatorService` into the `@NgModule.providers`.

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 `MomentumIndicatorService` into the `@NgModule.providers`. Momentum indicator will be represented by two lines (upperLine, signalLine). In momentum indicator the upperBand value is always render at the value 100.

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 `MACDIndicatorService` into the `@NgModule.providers`. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

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 `RsiIndicatorService` into the `@NgModule.providers`. 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.

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 `SmaIndicatorService` module using `@NgModule.providers`.

Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [type](#) as `Stochastic` and inject `StochasticIndicatorService` module using `@NgModule.providers` 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.

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 `TmaIndicatorService` module using `@NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
```

```

import { ChartModule, StockChartAllModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-stockchart id='chartcontainer' style="display:block;"
    [title]='title' [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
    [chartArea]= 'chartArea'>
    <e-stockchart-indicators>
      <e-stockchart-indicator type='Tma' [dataSource]='data'
      xName='x' field="Close" high='high' low='low' open='open' fill="blue"
      close='close' volume='volume'> </e-stockchart-indicator>
    </e-stockchart-indicators>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public data: Object[] = chartData;
  public primaryXAxis: Object = {
    title: 'Months',
    valueType: 'DateTime',
    intervalType: 'Months',
    majorGridLines: { width: 0}
  };
  public primaryYAxis: Object = {
    title: 'Price',
    labelFormat: '${value}',
    minimum: 30, maximum: 180,
    interval: 30,
  };
  public title: string = 'AAPL 2012-2017';
  public chartArea : Object = {
    border: { width : 0}
  };
  constructor() {
    //code
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Module Dependencies

To render a Indicator, it is mandatory to inject `LineSeriesService` module..

In addition to that, MACD Indicator requires `ColumnSeriesService` and BollingerBands requires `RangeAreaSeriesService`.

Tool tip in Angular Stock chart component

<!-- markdownlint-disable MD036 -->

stockchart will display details about the points through tooltip, when the mouse is moved over the point.

Default tooltip

By default, tooltip is not visible. Enable the tooltip by setting `enable` property to true and by injecting `TooltipService` into the `NgModule.providers`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LineSeriesService,
 ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, RangeTooltipService, CategoryService
 } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, RangeTooltipService,
 StepLineSeriesService, LegendService, TooltipService, CategoryService,
 ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
 [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
 [tooltip]='tooltip'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Spline'
 xName='date' yName='open' width=2 name='China' [marker]='marker'></e-
 stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
```

```

public tooltip?: Object;
ngOnInit(): void {
  this.chartData = chartData;
  this.primaryXAxis = {
    valueType: 'DateTime',
  };
  this.tooltip = { enable: true };
  this.marker = { visible: true, width: 10, height: 10 };
  this.title = 'Unemployment Rates 1975-2010';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD013 -->

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts';
import { DateTimeService, StepLineSeriesService, LineSeriesService,
 ColumnSeriesService } from '@syncfusion/ej2-angular-charts';
import { LegendService, TooltipService, RangeTooltipService, CategoryService
 } from '@syncfusion/ej2-angular-charts';
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, RangeTooltipService,
    StepLineSeriesService, LegendService, TooltipService, CategoryService,
    ColumnSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
 [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
 [tooltip]='tooltip'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Spline'
 xName='date' yName='open' width=2 name='China' [marker]='marker'></e-
 stockchart-series>

```

```

        </e-stockchart-series-collection>
    </ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public marker?: Object;
        public tooltip?: Object;
        public primaryYAxis?: Object;
        ngOnInit(): void {
            this.chartData = chartData;
            this.primaryXAxis = {
                valueType: 'DateTime'
            };
            this.tooltip = { enable: true, header: 'Unemployment', format:
'<b>${point.x} : ${point.y}</b>' };
            this.marker = { visible: true, width: 10, height: 10 };
            this.title = 'Unemployment Rates 1975-2010';
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD013 -->

Position the tooltip

By default, the tooltip is positioned at the left side of the stock chart. You can move the tooltip along with the mouse by setting **Nearest** to the [position](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, RangeTooltipService, CategoryService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule, StockChartAllModule, ChartAllModule
    ],
    providers: [ DateTimeService, LineSeriesService, RangeTooltipService,
StepLineSeriesService, LegendService, TooltipService, CategoryService,
ColumnSeriesService],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Spline'
xName='date' yName='open' width=2 name='China' [marker]='marker'></e-
stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public marker?: Object;
    public tooltip?: Object;
    public primaryYAxis?: Object;
    ngOnInit(): void {
        this.chartData = chartData;
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.tooltip = { enable: true, shared: true, position: 'Nearest' };
        this.marker = { visible: true, width: 10, height: 10 };
        this.title = 'Unemployment Rates 1975-2010';
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, RangeTooltipService, CategoryService
} from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule, StockChartAllModule, ChartAllModule
    ]
})

```

```

    ],
    providers: [ DateTimeService, LineSeriesService, RangeTooltipService,
    StepLineSeriesService, LegendService, TooltipService, CategoryService,
    ColumnSeriesService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
    [tooltip]='tooltip'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Spline'
        xName='date' yName='open' width=2 name='China' [marker]='marker'></e-
        stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public primaryYAxis?: Object;
    public marker?: Object;
    public tooltip?: Object;
    ngOnInit(): void {
      this.chartData = chartData;
      this.primaryXAxis = {
        valueType: 'DateTime'
      };
      this.tooltip = { enable: true, template: '#Unemployment' };
      this.marker = { visible: true, width: 10, height: 10 };
      this.title = 'Unemployment Rates 1975-2010';
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize the appearance of the 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.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, StepLineSeriesService, LineSeriesService,
ColumnSeriesService } from '@syncfusion/ej2-angular-charts'
import { LegendService, TooltipService, RangeTooltipService, CategoryService
} from '@syncfusion/ej2-angular-charts'

```

```

import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, RangeTooltipService,
    StepLineSeriesService, LegendService, TooltipService, CategoryService,
    ColumnSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[tooltip]='tooltip'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Spline'
xName='date' yName='open' width=2 name='China' [marker]='marker'></e-
stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public primaryYAxis?: Object;
  public marker?: Object;
  public tooltip?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.tooltip = {
      enable: true,
      format: '${series.name} ${point.x} : ${point.y}',
      fill: '#7bb4eb',
      border: {
        width: 2,
        color: 'grey'
      }
    };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


Cross hair in Angular Stock chart component

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
  [primaryXAxis]='primaryXAxis' [title]='title' [crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
  xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.crosshair= {
      enable: true
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Tooltip for axis

Tooltip label for an axis can be enabled by using [enable](#) property of `crosshairTooltip` in the corresponding axis.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
    CrosshairService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: { enable: true }
    };
    this.primaryYAxis = {
      majorTickLines: { color: 'transparent', width: 0 },
      crosshairTooltip: { enable: true }
    };
    this.crosshair = {
      enable: true
    };
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { LegendService, CrosshairService } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, LegendService,
CrosshairService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: {enable:true}
```

```

    };
    this.primaryYAxis = {
      majorTickLines: { color: 'transparent', width: 0 },
      crosshairTooltip: { enable: true }
    };
    this.crosshair = {
      enable: true,
      line: { width: 2, color: 'green' }
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend in Angular Stock chart component

Legend provides information about the series rendered in the Stock Chart. Legend can be added to a Stock Chart by enabling the [visible](#) option in the [legendSettings](#).

Position and Alignment

By using the [position](#) property, legend can be placed at **Left**, **Right**, **Top**, **Bottom** or **Custom** of the Stock Chart. The legend is positioned at the bottom of the Stock Chart, by default.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
    CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
    [trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
        volume='volume' xName='date' low='low' high='high' open='open' close='close'
        name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
  `
})

```

```

    </ejs-stockchart>`
  })
  export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public indicatorType: string[] = [];
    public trendlineType: string[] = [];
    public legendSettings?: Object;
    ngOnInit(): void {
      this.chartData = chartData;
      this.title = 'AAPL Stock Price';
      this.primaryXAxis = {
        valueType: 'DateTime'
      };
      this.legendSettings = {
        visible: true,
        //Legend position as top
        position: 'Top'
      };
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

[Custom](#) position is used to position the legend anywhere in the Stock Chart using x, y coordinates.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
    CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
    [trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>

```

```

        <e-stockchart-series [dataSource]='chartData' type='Candle'
        volume='volume' xName='date' low='low' high='high' open='open' close='close'
        name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public indicatorType: string[] = [];
        public trendlineType: string[] = [];
        public legendSettings?: Object;
        ngOnInit(): void {
            this.chartData = chartData;
            this.title = 'AAPL Stock Price';
            this.primaryXAxis = {
                valueType: 'DateTime'
            };
            this.legendSettings = {
                visible: true,
                //Legend position as custom
                position: 'Custom',
                location: { x: 200, y: 20 }
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Legend Alignment

<!-- markdownlint-disable MD036 -->

The legend can be align as **Center**, **Far** or **Near** to the Stock Chart using [alignment](#) property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [

```

```

        ChartModule, StockChartAllModule, ChartAllModule
    ],
    providers: [ DateTimeService, LineSeriesService, StockLegendService,
        CandleSeries ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
    ))
export class AppComponent implements OnInit {
    public primaryXAxis?: Object;
    public chartData?: Object[];
    public title?: string;
    public indicatorType: string[] = [];
    public trendlineType: string[] = [];
    public legendSettings?: Object;
    ngOnInit(): void {
        this.chartData = chartData;
        this.title = 'AAPL Stock Price';
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.legendSettings = {
            visible: true,
            position: 'Bottom',
            //Legend alignment as near
            alignment: 'Near'
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customization

To change the legend icon shape, [legendShape](#) property in the [series](#) can be used. By default legend icon shape is `seriesType`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'

```

```

import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService, CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL' legendShape='Pentagon'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'AAPL Stock Price';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.legendSettings = { visible: true };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Size

By default, legend takes 20% - 25% of the Stock 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 Stock Chart. The default legend size can be changed by using the [width](#) and [height](#) property of the `legendSettings`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
  CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'AAPL Stock Price';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.legendSettings = {
      visible: true,
      //Legend size for chart
      width: '500', height: '50',
      border: { width: 1, color: 'pink' }
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Legend Item Size

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

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
    CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'AAPL Stock Price';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.legendSettings = {
      visible: true,
      //Legend item size
      shapeHeight: 15, shapeWidth: 15
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Collapsing Legend Item

By default, series name will be displayed as legend. To skip the legend for a particular series, empty string to the series name can be given.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
    CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open'
close='close'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public legendSettings?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'AAPL Stock Price';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

```

        this.legendSettings = { visible: true };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Legend Title

The title for legend can be set using [title](#) property in `legendSettings`. 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`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, LineSeriesService } from '@syncfusion/ej2-angular-
charts'
import { StockLegendService, CandleSeries } from '@syncfusion/ej2-angular-
charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ DateTimeService, LineSeriesService, StockLegendService,
    CandleSeries ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [indicatorType]='indicatorType'
[trendlineType]='trendlineType' [legendSettings]='legendSettings'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
volume='volume' xName='date' low='low' high='high' open='open' close='close'
name='AAPL'></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public legendSettings?: Object;
  ngOnInit(): void {

```

```

        this.chartData = chartData;
        this.title = 'AAPL Stock Price';
        this.primaryXAxis = {
            valueType: 'DateTime'
        };
        this.legendSettings = {
            visible: true,
            title: 'Countries',
            titlePosition: 'Top',
            titleStyle: {
                fontFamily: 'verdana',
                fontStyle: 'Normal',
                fontWeight: 'Normal',
                size: '15px',
                textAlign: 'Center',
                color: 'blue',
                textOverflow: 'None'
            },
            maximumTitleWidth: 150
        };
    };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Note: To use legend feature, we need to inject `StockLegendService` into the `@NgModule.Providers`.

Period selector in Angular Stock chart component

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

Periods

Periods is an array of objects that allows users to specify the range of [periods](#). The `interval` property specifies the count value of the button, and the `text` property specifies the text to be displayed on button. The `intervalType` property allows users to customize the intervals of the buttons. The `intervalType` property supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, RangeNavigatorModule, StockChartAllModule,
ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, DateTimeService, PeriodSelectorService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { series1 } from './datasource';
import { PeriodsModel } from '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, RangeNavigatorModule, StockChartAllModule,
    ChartAllModule
  ],
  providers: [ AreaSeriesService, DateTimeService, PeriodSelectorService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [periods]='periods'
[exportType]='exportType' [seriesType]='seriesType'
[indicatorType]='indicatorType' [trendlineType]='trendlineType'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='x' yName='y' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public periods?: PeriodsModel[];
  public seriesType: string[] = [];
  public indicatorType: string[] = [];
  public trendlineType: string[] = [];
  public exportType: string[] = [];
  ngOnInit(): void {
    this.chartData = series1;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
    this.periods = [
      { intervalType: 'Minutes', interval: 1, text: '1m' },
      { intervalType: 'Minutes', interval: 30, text: '30m' },
      { intervalType: 'Hours', interval: 1, text: '1H' },
      { intervalType: 'Hours', interval: 12, text: '12H', selected:
true },
      { intervalType: 'Auto', text: '1D' }
    ];
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Visibility of period selector

The [enablePeriodSelector](#) property allows users to toggle the visibility of period selector.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, RangeNavigatorModule, StockChartAllModule,
ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, DateTimeService, PeriodSelectorService } from
'@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, RangeNavigatorModule, StockChartAllModule,
    ChartAllModule
  ],
  providers: [ AreaSeriesService, DateTimeService, PeriodSelectorService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[enablePeriodSelector]='enable'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Line'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  public enable: boolean = false;
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime',
      crosshairTooltip: { enable: true }
    };
    this.primaryYAxis = {
      majorTickLines: { color: 'transparent', width: 0 },
      crosshairTooltip: { enable: true }
    };
    this.crosshair = {
      enable: true
    }
  }
}
```

```
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Range selector in Angular Stock chart component

The left and right thumb of RangeNavigator are used to indicate the selected range in the large collection of data. Following are the ways you can select a range.

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and end through Date Range button.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, RangeNavigatorModule, StockChartAllModule,
ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, DateTimeService, RangeTooltipService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, RangeNavigatorModule, StockChartAllModule,
    ChartAllModule
  ],
  providers: [ AreaSeriesService, DateTimeService, RangeTooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public crosshair?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
```



```

        this.title = 'Efficiency of oil-fired power production';
        this.primaryXAxis = {
            valueType: 'DateTime',
            crosshairTooltip: {enable:true}
        };
        this.primaryYAxis = {
            majorTickLines: { color: 'transparent', width: 0 },
            crosshairTooltip: {enable:true}
        };
        this.crosshair= {
            enable: true
        };
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Visibility of range selector

The [enableSelector](#) property allows users to toggle the visibility of range selector.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, RangeNavigatorModule, StockChartAllModule,
ChartAllModule } from '@syncfusion/ej2-angular-charts'
import { AreaSeriesService, DateTimeService, RangeTooltipService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, RangeNavigatorModule, StockChartAllModule,
    ChartAllModule
  ],
  providers: [ AreaSeriesService, DateTimeService, RangeTooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container" [enableSelector]='enable'
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[crosshair]='crosshair'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public chartData?: Object[];

```

```

public title?: string;
public crosshair?: Object;
public enable: boolean = false;
ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
        valueType: 'DateTime',
        crosshairTooltip: {enable:true}
    };
    this.primaryYAxis = {
        majorTickLines: { color: 'transparent', width: 0 },
        crosshairTooltip: {enable:true}
    };
    this.crosshair= {
        enable: true
    };
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Export print in Angular Stock chart component

The rendered stock chart can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format using the export dropdown button in the period selector toolbar. You can choose the required format using the export dropdown button in stock-chart.

The rendered stock chart can be printed directly using print button in period selector toolbar.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LegendService,
 DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ CategoryService, ColumnSeriesService, LegendService,
 DataLabelService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
 [primaryXAxis]='primaryXAxis' [title]='title'>
    <e-stockchart-series-collection>

```

```

        <e-stockchart-series [dataSource]='chartData' type='Candle'
        xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
</ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        public primaryXAxis?: Object;
        public chartData?: Object[];
        public title?: string;
        public crosshair?: Object;
        ngOnInit(): void {
            this.chartData = chartData;
            this.title = 'Efficiency of oil-fired power production';
            this.primaryXAxis = {
                valueType: 'DateTime'
            };
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Disable Export and print

To empty the value of `exportType` for to disable the Export button.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { CategoryService, ColumnSeriesService, LegendService,
DataLabelService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
    imports: [
        ChartModule, StockChartAllModule, ChartAllModule
    ],
    providers: [ CategoryService, ColumnSeriesService, LegendService,
DataLabelService ],
    standalone: true,
    selector: 'app-container',
    template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [title]='title' [exportType]='exportType'>
    <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
        xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
    </ejs-stockchart>`
})

```

```
export class AppComponent implements OnInit {
  public primaryXAxis?: Object;
  public chartData?: Object[];
  public title?: string;
  public exportType: string[] = [];
  ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.primaryXAxis = {
      valueType: 'DateTime'
    };
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Appearance in Angular Stock chart component

Stock Chart Title

StockChart can be given a title using [title](#) property, to show the information about the data plotted.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, CandleSeriesService, LegendService,
 CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ TooltipService, DateTimeService, CandleSeriesService,
    LegendService, CategoryService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
    [primaryXAxis]='primaryYAxis' [primaryYAxis]='primaryYAxis' [title]='title'>
      <e-stockchart-series-collection>
        <e-stockchart-series [dataSource]='chartData' type='Candle'
          xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
      </e-stockchart-series-collection>
    </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
```

```

public primaryYAxis: any;
ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

<!-- markdownlint-disable MD036 -->

Title Customization

The `textStyle` property of stockchart title provides options to customize the `size`, `color`, `fontFamily`, `fontWeight`, `fontStyle`, `opacity`, `textAlignment` and `textOverflow`.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, CandleSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ TooltipService, DateTimeService, CandleSeriesService,
LegendService, CategoryService, LineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryXAxis' [title]='title'
[titleStyle]='titleStyle'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public title?: string;
  titleStyle?: any;
  primaryXAxis: any;
  ngOnInit(): void {
    this.chartData = chartData;

```

```

        this.title = 'Efficiency of oil-fired power production';
        this.titleStyle = {
            fontFamily: "Arial",
            fontStyle: 'italic',
            fontWeight: 'regular',
            color: "#E27F2D",
            size: '23px',
            textOverflow: 'Wrap'
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Stock Chart Theme

Changing Stock Chart theme will affect background color, grid lines, tooltip colors and appearance.

[theme](#) property of Stock chart is shipped with several built-in themes such as [Material](#), [Fabric](#), [Bootstrap](#), [HighContrastLight](#), [MaterialDark](#), [FabricDark](#), [FabricDark](#), [HighContrast](#) and [BootstrapDark](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule } from
'@syncfusion/ej2-angular-charts'
import { DateTimeService, CandleSeriesService, LegendService,
CategoryService, LineSeriesService } from '@syncfusion/ej2-angular-charts'
import { TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule
  ],
  providers: [ TooltipService, DateTimeService, CandleSeriesService,
LegendService, CategoryService, LineSeriesService],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis' [title]='title'
[theme]='theme'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='chartData' type='Candle'
xName='date' yName='open' name='India' width=2 ></e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit {

```

```

public chartData?: Object[];
public title?: string;
theme?: string;
primaryXAxis: any;
primaryYAxis: any;
ngOnInit(): void {
    this.chartData = chartData;
    this.title = 'Efficiency of oil-fired power production';
    this.theme = 'HighContrast';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Axis Customization](#)

Stock events in Angular Stock chart component

<!-- markdownlint-disable MD036 -->

Stock Events visualizes stock events in stock chart. 'SplineSeries' is used to represent selected data value. You can customize the specific data value using `stockEvents` event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule,
RangeNavigatorModule } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, SplineSeriesService, RangeTooltipService,
StepLineSeriesService, CategoryService, ParetoSeriesService,
ColumnSeriesService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
import { IStockChartEventArgs, ChartTheme, ITooltipRenderEventArgs } from
 '@syncfusion/ej2-angular-charts';
@Component({
    imports: [
        ChartModule, StockChartAllModule, ChartAllModule,
        RangeNavigatorModule
    ],
    providers: [ LineSeriesService, SplineSeriesService, RangeTooltipService,
        StepLineSeriesService, CategoryService, SplineAreaSeriesService,
        ParetoSeriesService, ColumnSeriesService,
            MultiColoredLineSeriesService, TooltipService ],
    standalone: true,
    selector: 'app-container',

```

```

    template: `<ejs-stockchart id='stockChartEvents'
[enablePeriodSelector]='enable' [title]='title' [titleStyle]='titleStyle'
    [chartArea]='chartArea' [primaryXAxis]='primaryXAxis'
style="display:block;" [tooltip]='tooltip'
    [crosshair]='crosshair' [primaryYAxis]='primaryYAxis'
(tooltipRender)='tooltipRender($event)' [seriesType]='seriesType'
    [indicatorType]='indicatorType'>
    <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='data1' type='Spline'
xName='date' yName='high' close='close'>
    </e-stockchart-series>
    </e-stockchart-series-collection>
    <e-stockchart-stockevents>
    <e-stockchart-stockevent [date]="date1" text="Q2"
description="2012 Quarter2"
        type="Flag" background='#6c6d6d' [border]="border1"></e-
stockchart-stockevent>
    <e-stockchart-stockevent [date]="date2" text="Open"
description="Markets opened"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date3" text="Q3"
description="2013 Quarter3"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date4" text="Q4"
description="2013 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date5" text="G"
description="Google Stock"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date6" text="Y"
description="Yahoo Stock"
        type="Square" [textStyle]="textStyle"
background="#841391" [border]="border5">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date7" text="Y2"
description="Year 2013" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date8" text="Q2"
description="2013 Quarter2"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date9" text="Q2"
description="Surge in Stocks"
        type="ArrowUp" [textStyle]="textStyle"
background="#3ab0f9" [border]="border4">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date10" text="Q3"
description="2013 Quarter3"

```



```

        type="Flag" [textStyle]="textStyle" background="#f48a21"
[border]="border2">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date11" text="Q4"
description="2013 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date12" text="Y3"
description="Year 2014" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date13" text="Q2"
description="2014 Quarter2"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date14" text="Q3"
description="2014 Quarter3"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date15" text="Q4"
description="2014 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date16" text="Y4"
description="Year 2015" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date17" text="End"
description="Markets closed"
        type="ArrowDown" [textStyle]="textStyle"
background="#3ab0f9" [border]="border4">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date18" text="A"
description="Amazon Stock"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date19" text="Q1"
description="AAPL Stock"
        [textStyle]="textStyle" background="#dd3c9f"
[border]="border7" type="Text">
    </e-stockchart-stockevent>
    <e-stockchart-stockevent [date]="date20" text="Close"
description="Markets closed"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
    </e-stockchart-stockevents>
    </ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        tooltipRender($event: any) {
            throw new Error('Method not implemented.');
```

```

public primaryXAxis?: Object;
public data1?: Object[];
public title?: string;
public primaryYAxis?: Object;
public marker?: Object;
public tooltip?: Object;
public date1: Date = new Date(2012, 3, 1);
public date2: Date = new Date(2012, 3, 20);
public date3: Date = new Date(2012, 6, 1);
public date4: Date = new Date(2012, 9, 1);
public date5: Date = new Date(2012, 7, 30);
public date6: Date = new Date(2012, 10, 1);
public date7: Date = new Date(2012, 12, 0);
public date8: Date = new Date(2013, 3, 1);
public date9: Date = new Date(2013, 3, 20);
public date10: Date = new Date(2013, 6, 1);
public date11: Date = new Date(2013, 9, 1);
public date12: Date = new Date(2013, 12, 0);
public date13: Date = new Date(2014, 3, 1);
public date14: Date = new Date(2014, 6, 1);
public date15: Date = new Date(2014, 9, 1);
public date16: Date = new Date(2014, 12, 0);
public date17: Date = new Date(2014, 2, 2);
public date18: Date = new Date(2015, 1, 7);
public date19: Date = new Date(2015, 1, 2);
public date20: Date = new Date(2015, 2, 12);
public border1: object = { color: '#6c6d6d' };
public textStyle: object = { color: 'white' };
public border2: object = { color: '#f48a21' };
public border3: object = { color: '#6c6d6d' };
public border4: object = { color: '#3ab0f9' };
public border5: object = { color: '#841391' };
public border6: object = { color: '#6322e0' };
public border7: object = { color: '#dd3c9f' };
enable: any;
titleStyle: any;
chartArea: any;
crosshair: any;
seriesType: any;
indicatorType: any;
onSeries: any;
ngOnInit(): void {
    this.data1 = chartData;
    this.primaryXAxis = {
        valueType: 'DateTime',
    };
    this.tooltip = { enable: true };
    this.marker = { visible: true, width: 10, height: 10 };
    this.title = 'Unemployment Rates 1975-2010';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Stock Events for individual series

<!-- markdownlint-disable MD036 -->

By default, stock events will be showed for all series. Now, you can set the stock events for particular series using `seriesIndexes` property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartModule, StockChartAllModule, ChartAllModule,
RangeNavigatorModule } from '@syncfusion/ej2-angular-charts'
import { LineSeriesService, SplineSeriesService, RangeTooltipService,
StepLineSeriesService, CategoryService, ParetoSeriesService,
ColumnSeriesService,
    SplineAreaSeriesService, MultiColoredLineSeriesService,
TooltipService } from '@syncfusion/ej2-angular-charts'
import { Component, OnInit } from '@angular/core';
import { chartData } from './datasource';
import { IStockChartEventArgs, ChartTheme, ITooltipRenderEventArgs } from
 '@syncfusion/ej2-angular-charts';
@Component({
  imports: [
    ChartModule, StockChartAllModule, ChartAllModule,
    RangeNavigatorModule
  ],
  providers: [ LineSeriesService, SplineSeriesService, RangeTooltipService,
    StepLineSeriesService, CategoryService, SplineAreaSeriesService,
    ParetoSeriesService, ColumnSeriesService,
    MultiColoredLineSeriesService, TooltipService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id='stockChartEvents'
[enablePeriodSelector]='enable' [title]='title' [titleStyle]='titleStyle'
[chartArea]='chartArea' [primaryXAxis]='primaryXAxis'
style="display:block;" [tooltip]='tooltip'
[crosshair]='crosshair' [primaryYAxis]='primaryYAxis'
(tooltipRender)='tooltipRender($event)' [seriesType]='seriesType'
[indicatorType]='indicatorType'>
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='data1' type='Spline'
xName='date' yName='high'>
      </e-stockchart-series>
      <e-stockchart-series [dataSource]='data1' type='Spline'
xName='date' yName='low'>
      </e-stockchart-series>
    </e-stockchart-series-collection>
    <e-stockchart-stockevents>
      <e-stockchart-stockevent [date]="date1" text="Q2"
description="2012 Quarter2"
type="Flag" background='#6c6d6d' [border]="border1"
[seriesIndexes]="seriesIndex1"></e-stockchart-stockevent>
```

```

        <e-stockchart-stockevent [date]="date2" text="Open"
description="Markets opened"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date3" text="Q3"
description="2013 Quarter3"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date4" text="Q4"
description="2013 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3" [seriesIndexes]="seriesIndex2">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date5" text="G"
description="Google Stock"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date6" text="Y"
description="Yahoo Stock"
        type="Square" [textStyle]="textStyle"
background="#841391" [border]="border5" [seriesIndexes]="seriesIndex3">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date7" text="Y2"
description="Year 2013" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date8" text="Q2"
description="2013 Quarter2"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date9" text="Q2"
description="Surge in Stocks"
        type="ArrowUp" [textStyle]="textStyle"
background="#3ab0f9" [border]="border4" [seriesIndexes]="seriesIndex4">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date10" text="Q3"
description="2013 Quarter3"
        type="Flag" [textStyle]="textStyle" background="#f48a21"
[border]="border2">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date11" text="Q4"
description="2013 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date12" text="Y3"
description="Year 2014" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6" [seriesIndexes]="seriesIndex5">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date13" text="Q2"
description="2014 Quarter2"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3" [seriesIndexes]="seriesIndex6">

```

```

        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date14" text="Q3"
description="2014 Quarter3"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date15" text="Q4"
description="2014 Quarter4"
        type="Flag" [textStyle]="textStyle" background="#6c6d6d"
[border]="border3">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date16" text="Y4"
description="Year 2015" type="Pin"
        [showOnSeries]="onSeries" [textStyle]="textStyle"
background="#6322e0" [border]="border6" [seriesIndexes]="seriesIndex7">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date17" text="End"
description="Markets closed"
        type="ArrowDown" [textStyle]="textStyle"
background="#3ab0f9" [border]="border4">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date18" text="A"
description="Amazon Stock"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date19" text="Q1"
description="AAPL Stock"
        [textStyle]="textStyle" background="#dd3c9f"
[border]="border7" type="Text" [seriesIndexes]="seriesIndex8">
        </e-stockchart-stockevent>
        <e-stockchart-stockevent [date]="date20" text="Close"
description="Markets closed"
        [textStyle]="textStyle" background="#f48a21"
[border]="border2"></e-stockchart-stockevent>
        </e-stockchart-stockevents>
    </ejs-stockchart>`
    })
    export class AppComponent implements OnInit {
        indicatorType: any;
        crosshair: any;
        chartArea: any;
        onSeries: any;
        titleStyle: any;
        seriesType: any;
        tooltipRender($event: any) {
            throw new Error('Method not implemented. ');
        }
        public primaryXAxis?: Object;
        public data1?: Object[];
        public title?: string;
        public primaryYAxis?: Object;
        public marker?: Object;
        public tooltip?: Object;
        public date1: Date = new Date(2012, 3, 1);
        public date2: Date = new Date(2012, 3, 20);
        public date3: Date = new Date(2012, 6, 1);
        public date4: Date = new Date(2012, 9, 1);
        public date5: Date = new Date(2012, 7, 30);
    }

```

```

public date6: Date = new Date(2012, 10, 1);
public date7: Date = new Date(2012, 12, 0);
public date8: Date = new Date(2013, 3, 1);
public date9: Date = new Date(2013, 3, 20);
public date10: Date = new Date(2013, 6, 1);
public date11: Date = new Date(2013, 9, 1);
public date12: Date = new Date(2013, 12, 0);
public date13: Date = new Date(2014, 3, 1);
public date14: Date = new Date(2014, 6, 1);
public date15: Date = new Date(2014, 9, 1);
public date16: Date = new Date(2014, 12, 0);
public date17: Date = new Date(2014, 2, 2);
public date18: Date = new Date(2015, 1, 7);
public date19: Date = new Date(2015, 1, 2);
public date20: Date = new Date(2015, 2, 12);
public border1: object = { color: '#6c6d6d' };
public textStyle: object = { color: 'white' };
public border2: object = { color: '#f48a21' };
public border3: object = { color: '#6c6d6d' };
public border4: object = { color: '#3ab0f9' };
public border5: object = { color: '#841391' };
public border6: object = { color: '#6322e0' };
public border7: object = { color: '#dd3c9f' };
public seriesIndex1: number[] = [0];
public seriesIndex2: number[] = [1];
public seriesIndex3: number[] = [0];
public seriesIndex4: number[] = [1];
public seriesIndex5: number[] = [0];
public seriesIndex6: number[] = [1];
public seriesIndex7: number[] = [0];
public seriesIndex8: number[] = [1];
enable: any;
ngOnInit(): void {
  this.data1 = chartData;
  this.primaryXAxis = {
    valueType: 'DateTime',
  };
  this.tooltip = { enable: true };
  this.marker = { visible: true, width: 10, height: 10 };
  this.title = 'Unemployment Rates 1975-2010';
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Accessibility in Angular Stock chart component

The Stock chart 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 Stock chart 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 Stock chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Stock chart component:

- img (role)
- button (role)
- region (role)

- `aria-label` (attribute)
- `aria-hidden` (attribute)
- `aria-pressed` (attribute)

Keyboard interaction

The Stock chart 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 Stock chart component.

| **Press** | **To do this** |

| --- | --- |

| **Alt + J** | Moves the focus to the Stock chart element. |

| **Tab** | Moves the focus to the next element in the Stock chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Stock 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. |

| **ESC** | Cancel the tooltip for the data point. |

| **Ctrl + P** | Prints the Stock chart. |

Ensuring accessibility

The Stock chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Stock chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Stock chart component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Internationalization in Angular Stock chart component

Chart provide supports for internationalization for below chart elements.

- Axis label.
- Tooltip.
- Crosshair

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 labels, tooltip and crosshair labels are globalized to EUR.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ChartAllModule, StockChartAllModule } from '@syncfusion/ej2-angular-charts'
import { CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService } from '@syncfusion/ej2-angular-charts'
import { Component, ViewEncapsulation, OnInit } from '@angular/core';
import { chartData } from './datasource';
import { setCurrencyCode, L10n, setCulture } from '@syncfusion/ej2-base';
import { ITooltipRenderEventArgs } from '@syncfusion/ej2-angular-charts';
setCurrencyCode("EUR");
@Component({
  imports: [
    ChartAllModule, StockChartAllModule
  ],
  providers: [ CandleSeriesService, LineSeriesService, TmaIndicatorService,
DateTimeService ],
  standalone: true,
  selector: 'app-container',
  template:
`<ejs-stockchart id='chart-container' [primaryXAxis]='primaryXAxis'
style="display:block;"
[primaryYAxis]='primaryYAxis' [crosshair]='crosshair'
(tooltipRender)='tooltipRender($event)' [tooltip]='tooltip'>
  <e-stockchart-series-collection>
    <e-stockchart-series [dataSource]='data1' type='Candle' xName='x'
yName='high' high='high' low='low'>
    </e-stockchart-series>
  </e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public data1: Object[] = chartData;
  public primaryXAxis: Object = { majorGridLines: { color: 'transparent' },
crosshairTooltip: { enable: true } };
  public primaryYAxis: Object = {
    lineStyle: { color: 'transparent' },
    majorTickLines: { color: 'transparent', width: 0 },
    labelFormat: 'c',
    crosshairTooltip: { enable: true }
  };
  public crosshair: Object = {
    enable: true
  };
  public tooltipRender(args: ITooltipRenderEventArgs | any): void {
    if (args.text.split('<br/>')[4]) {
      let target: number =
parseInt(args.text.split('<br/>')[4].split('<b>')[1].split('</b>')[0], 10);
      let value: string = (target / 100000000).toFixed(1) + 'B';
```

```

        args.text =
args.text.replace(args.text.split('<br/>')[4].split('<b>')[1].split('</b>')[0]
, value);
    };
    public tooltip: object = { enable: true };
    constructor() {
        //code
    }
    ngOnInit(): void {
        throw new Error('Method not implemented.');
```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Localization in Angular Stock chart component

Localization library allows to localize the default text content of StockChart. In stock 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](#)

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { StockChartAllModule, ChartAllModule, RangeNavigatorAllModule } from
 '@syncfusion/ej2-angular-charts'
import { DateTimeService, SplineSeriesService } from '@syncfusion/ej2-
angular-charts'
import { Component, OnInit } from '@angular/core';
```

```

import { chartData } from './datasource';
import { L10n, setCulture } from '@syncfusion/ej2-base';
setCulture('ar-AR');
L10n.load({
  'ar-AR': {
    'chart': {
      ZoomIn: 'تكبير',
      ZoomOut: 'تصغير',
      Zoom: 'زوم',
      Pan: 'مقلاة',
      Reset: 'إعادة تعيين',
      ResetZoom: 'زوم إعادة تعيين'
    }
  }
});
@Component({
  imports: [
    StockChartAllModule, ChartAllModule, RangeNavigatorAllModule
  ],
  providers: [ DateTimeService, SplineSeriesService ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-stockchart id="chart-container"
[primaryXAxis]='primaryXAxis' [primaryYAxis]='primaryYAxis'
[crosshair]='crosshair' [zoomSettings]='zoomSettings' locale='ar-AR'>
<e-stockchart-series-collection>
<e-stockchart-series [dataSource]='chartData' type='Spline'
xName='x' yName='open' [marker]='marker'></e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent implements OnInit {
  public chartData?: Object[];
  public primaryXAxis?: Object;
  public primaryYAxis?: Object;
  public crosshair?: Object;
  public zoomSettings?: Object;
  public marker?: Object;
  ngOnInit(): void {
    this.chartData = chartData;
    this.primaryXAxis = {
      valueType: 'DateTime',
      majorGridLines: { color: 'transparent' }, crosshairTooltip: { enable:
true }
    };
    this.primaryYAxis = {
      lineStyle: { color: 'transparent' },
      majorTickLines: { color: 'transparent', width: 0 },
      crosshairTooltip: { enable: true }
    };
    this.zoomSettings = {
      enableMouseWheelZooming: true,
      mode: 'XY'
    };
    this.crosshair = { enable: true };
    this.marker = { visible: true, width: 10, height: 10 };
  }
}

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

How To**Live data in Angular Stock chart component**

You can update a stockchart 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 { Component } from '@angular/core';
@Component({
  selector: 'app-container',
  // specifies the template string for the StockChart component
  template: `<ejs-stockchart id="chart-container" #chart [primaryXAxis]='primaryXAxis'
[periods]='periods'>
<e-stockchart-series-collection>
<e-stockchart-series [dataSource]='series1' type='Candle' xName='x' yName='high' high='high' low='low'
name='India' width=2 >
</e-stockchart-series>
</e-stockchart-series-collection>
</ejs-stockchart>`
})
export class AppComponent {
}
`

```

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.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'

```

```

import { BrowserModule } from '@angular/platform-browser'
import { StockChartModule, DateTimeService, CandleSeriesService } from
 '@syncfusion/ej2-angular-charts'
import { Component, OnInit, OnDestroy, ViewChild } from '@angular/core';
import { IStockChartEventArgs, getElement, PeriodsModel, StockChartComponent
 } from '@syncfusion/ej2-angular-charts';
import { StockChart, StockChartAxisModel } from '@syncfusion/ej2-charts';
@Component({
  imports: [
    StockChartModule
  ],
  providers: [ DateTimeService, CandleSeriesService ],
  standalone: true,
  selector: 'app-container',
  template:
    `<ejs-stockchart id="chart-container" #chart
[primaryXAxis]='primaryXAxis' [periods]='periods' title="Live Stock Chart">
    <e-stockchart-series-collection>
      <e-stockchart-series [dataSource]='series1' type='Candle'
xName='x' yName='high' high='high' low='low' name='India' width=2 >
    </e-stockchart-series>
    </e-stockchart-series-collection>
  </ejs-stockchart>`
})
export class AppComponent implements OnInit, OnDestroy {
  public chartData?: Object[];
  public title?: string;
  public point1?: Object;
  public range?: boolean;
  public period?: boolean;
  public primaryXAxis: StockChartAxisModel = { valueType: 'DateTime' };
  public series1: Object[] = [];
  public value: number = 10;
  public intervalId?: any;
  public setTimeoutValue?: number;
  public i: number = 0;
  public date: Date = new Date('2019-09-16');
  public periods?: PeriodsModel[];
  @ViewChild('chart', { static: true })
  public stock?: StockChartComponent | StockChartComponent;
  ngOnInit() {
    this.title = 'Efficiency of oil-fired power production';
    this.range = false;
    this.period = false;
    this.periods = [
      { intervalType: 'Minutes', interval: 1, text: '1m' },
      { intervalType: 'Minutes', interval: 30, text: '30m' },
      { intervalType: 'Hours', interval: 1, text: '1H', selected: true },
      { intervalType: 'Hours', interval: 2, text: '2H' },
    ];
    this.setTimeoutValue = 5000;
    this.intervalId = setInterval(
      () => {
        let i: number;
        if (getElement('chart-container') === null) {
          clearInterval(this.intervalId);
        } else {

```

```

        this.date = new Date(this.date.getTime() + (1 * 60 * 1000));
        // this.series1.push( { x: this.date, high: 100, low: 50, open :
60,
        //   close : 70 });
        this.series1.push( {
            x: this.date,
            high: Math.floor(Math.random() * (100 - 90 + 1) + 90),
            low: Math.floor(Math.random() * (60 - 50 + 1) + 50),
            close: Math.floor(Math.random() * (99 - 51 + 1) + 51),
            open: Math.floor(Math.random() * (99 - 51 + 1) + 51)
        });
        this.i++;
        (this.stock as StockChartComponent ).series[0].dataSource =
this.series1;
        this.stock?.refresh();
    }
    },
    3000);
}
ngOnDestroy() {
    clearInterval(this.intervalId);
}
constructor() {
    for ( ; this.i < 60; this.i++) {
        if (Math.random() > .5) {
            if (this.value < 25) {
                this.value += Math.random() * 2.0;
            } else {
                this.value -= 2.0;
            }
        }
        this.date = new Date(2019, 8, 16, 10, this.i);
        //this.date = new Date(this.date.setDate(this.date.getDate() + 1));
        this.series1[this.i] = {
            x: this.date,
            high: Math.floor(Math.random() * (100 - 90 + 1) + 90),
            low: Math.floor(Math.random() * (60 - 50 + 1) + 50),
            close: Math.floor(Math.random() * (99 - 51 + 1) + 51),
            open: Math.floor(Math.random() * (99 - 51 + 1) + 51)
        };
    }
}
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Toggle Switch Button

Getting started with Angular Switch component

This section explains how to create a simple Switch, and demonstrate the basic usage of the Switch module in an Angular environment.

Dependencies

The following list of dependencies are required to use the Switch module in your application.

```
`typescript
|-- @syncfusion/ej2-angular-buttons
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
`,`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`npm install -g @angular/cli
`,`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`ng new my-app
cd my-app
`,`
```

Installing Syncfusion Switch package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-buttons](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-buttons --save
```

```
,
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the ngcc package use the below.

Add [@syncfusion/ej2-angular-buttons@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-buttons@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-buttons:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Switch module

Import Switch module into Angular application(`app.module.ts`) from the package `@syncfusion/ej2-angular-buttons`.

```
`typescript
```

```
import { NgModule } from '@angular/core';
```

```
import { BrowserModule } from '@angular/platform-browser';
```

```
// Imported Syncfusion Switch module from buttons package.
```

```
import { SwitchModule } from '@syncfusion/ej2-angular-buttons';
```

```
import { AppComponent } from './app.component';
```

```
@NgModule({
```

```
imports: [ BrowserModule, SwitchModule ], // Registering EJ2 Switch Module.
```

```
declarations: [ AppComponent ],
```

```
bootstrap: [ AppComponent ]
```

```
})
```

```
export class AppModule { }
```

```
,
```

Adding Syncfusion Switch component

Modify the template in `app.component.ts` file to render the Switch component.

```
`typescript
```



```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render Switch with checked state. -->
<ejs-switch [checked]="true"></ejs-switch>`
})
export class AppComponent { }
,
```

Adding CSS reference

Add Switch component's styles as given below in `style.css`.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
,
```

Running the application

Run the application in the browser using the following command:

```
,
ng serve
,
```

The below example shows a basic Switch component,

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render Switch with checked state. -->
    <ejs-switch [checked]="true"></ejs-switch></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set text on Switch

This section explains how to set [onLabel](#)

and [offLabel](#) texts on Switch. In the following example, `onLabel` is set as

`ON` and `offLabel` is set as `OFF`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render Switch. -->
    <ejs-switch onLabel="ON" offLabel="OFF"
[checked]="true"></ejs-switch>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Switch does not have text support for material themes, and does not support long custom text.

Accessibility in Angular Switch component

The Switch 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 Switch 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 Switch component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Switch component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the switch component. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Switch 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 Switch component.

| Press | To do this |

| --- | --- |

| Space | When the switch has focus, pressing the Space key changes the state of the switch. |

Ensuring accessibility

The Switch component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Switch component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Switch component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Bind data using two way binding in Angular Switch component

Switch component supports two way binding.

In this following example, two way binding for Switch is illustrated with CheckBox component. The steps to achieve two way binding in Switch are as follows,

- Initialize Switch component and bind the checked value using `ngModel` as in the below code using "banana in a box" syntax,

`typescript

```
<ejs-switch #switch [(ngModel)]="checked"></ejs-switch>
```

,

- Initialize Checkbox component and assign the [checked](#) property value like the below code,

`typescript

```
<ejs-checkbox #checkbox [(checked)]="checked"></ejs-checkbox>
```

,

- Now, the changes made in Switch will reflect in CheckBox (i.e When the state of Switch is changed to checked state then the CheckBox state will also change to checked state) and vice versa.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule, CheckBoxModule } from '@syncfusion/ej2-angular-buttons'
import { FormsModule } from '@angular/forms'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
```

```

@Component({
  imports: [

    SwitchModule,
    CheckBoxModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id='container'>
      <table class='tablealign'>
        <thead>
          <th></th>
          <th>
            <h4>Switch</h4>
          </th>
          <th>
            <h4>Checkbox</h4>
          </th>
        </thead>
        <tr>
          <td class='dataalign'>
            <label>Wi-Fi</label>
          </td>
          <td class='dataalign'>
            <ejs-switch #wswitch
[ (ngModel) ]="checkedwifi">
              </ejs-switch>
            </td>
          <td class='dataalign'>
            <ejs-checkbox #wcheckbox
[ (checked) ]="checkedwifi">
              </ejs-checkbox>
            </td>
        </tr>
        <tr>
          <td class='dataalign'>
            <label>Bluetooth</label>
          </td>
          <td class='dataalign'>
            <ejs-switch #bswitch
[ (ngModel) ]="checkedbluetooth">
              </ejs-switch>
            </td>
          <td class='dataalign'>
            <ejs-checkbox #bcheckbox
[ (checked) ]="checkedbluetooth">
              </ejs-checkbox>
            </td>
        </tr>
      </table>
    </div>
  </div>`
})
export class AppComponent {
  public checkedwifi: boolean = true;

```

```
public checkedbluetooth: boolean = false;
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Change size in Angular Switch component

The different Switch sizes available are default and small. To reduce the size of default Switch to small, set the [cssClass](#) property to `e-small`.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <table class='size'>
      <tr>
        <td class='lSize'>Small</td>
        <td>
          <ejs-switch cssClass="e-small"></ejs-switch>
        </td>
      </tr>
      <tr>
        <td class='lSize'>Default</td>
        <td>
          <ejs-switch></ejs-switch>
        </td>
      </tr>
    </table>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize the appearance of a switch in Angular Switch component

You can customize the appearance of the Switch component using the CSS rules. Define your own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

Customize Switch bar and handle

Switch bar and handle can be customized as per requirement using CSS rules. Switch bar and handle customized using `cssClass` property. In the following sample, the `border-radius` CSS property for the Switch bar (`e-switch-inner`) and handle (`e-switch-handle`) elements was changed border radius circle to square shape.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id='container'>
      <div class="container switch-control">
        <div>
          <h3>Customizing Shape</h3>
        </div>
        <div>
          <label for="switch1" style="padding: 10px 82px 10px 7px">
Square Switch </label>
          <ejs-switch id="switch1" cssClass="square" ></ejs-switch>
        </div>
        <div>
          <label for="switch2" style="padding: 10px 76px 10px 7px">
Bar and handle </label>
          <ejs-switch id="switch2" cssClass="custom-switch"
[checked]= "true" ></ejs-switch>
        </div>
        <div>
          <label for="switch3" style="padding: 10px 96px 10px 7px">
Handle text </label>
          <ejs-switch id="switch3" cssClass="handle-text" ></ejs-
switch>
        </div>
      </div>
    </div>
  `
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

STYLES.CSS

```
@import 'node_modules/@syncfusion/ej2-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-buttons/styles/material.css';
.e-section-control {
    margin-top: 150px;
}
#container {
    margin-left: 10px;
}
.switch-control div {
    display: flex;
    align-items: center;
}
.switch-control h3 {
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    text-indent: 23px;
}
.switch-control {
    margin: 90px auto;
    width: 240px;
}
.switch-control label {
    -moz-user-select: none;
    cursor: pointer;
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    font-size: 13px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
/* Square Switch */
.e-switch-wrapper.square .e-switch-inner,
.e-switch-wrapper.square .e-switch-handle {
    border-radius: 0;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.custom-switch {
    width: 50px;
    height: 24px;
}
.e-switch-wrapper.custom-switch .e-switch-handle {
    width: 20px;
    height: 16px;
```



```

}
.e-switch-wrapper.custom-switch .e-switch-inner,
.e-switch-wrapper.custom-switch .e-switch-handle {
  border-radius: 0;
}
.e-switch-wrapper.custom-switch .e-switch-handle.e-switch-active {
  left: 42px;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.handle-text {
  width: 58px;
  height: 24px;
}
.e-switch-wrapper.handle-text .e-switch-handle {
  width: 26px;
  height: 20px;
  left: 2px;
  background-color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-handle {
  border-radius: 0;
}
.e-switch-wrapper.handle-text .e-switch-handle.e-switch-active {
  left: 46px;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-text:hover .e-switch-inner.e-switch-active .e-switch-on {
  background-color: #4d841d;
  border-color: #4d841d;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-off {
  background-color: #e3165b;
  border-color: #e3165b;
}
.e-switch-wrapper.handle-text .e-switch-inner:after,
.e-switch-wrapper.handle-text .e-switch-inner:before {
  font-size: 10px;
  position: absolute;
  line-height: 21px;
  font-family: "Helvetica", sans-serif;
  z-index: 1;
  height: 100%;
  transition: all 200ms cubic-bezier(0.445, 0.05, 0.55, 0.95);
}
.e-switch-wrapper.handle-text .e-switch-inner:before {
  content: "OFF";
  color: #e3165b;
  left: 3px;
}
.e-switch-wrapper.handle-text .e-switch-inner:after {
  content: "ON";
  right: 5px;
  color: #fff;
}

```

```
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:before {
  color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:after {
  color: #4d841d;
}
.e-switch-wrapper.handle-text:not(.e-switch-disabled):hover .e-switch-
handle:not(.e-switch-active) {
  background-color: #fff;
}
```

Color the Switch

Switch colors can be customized as per the requirement using CSS rules. Switch bar and handle colors customized using `cssClass` property. In the following sample, the Switch bar (`e-switch-inner`) element background and border colors were changed from default colors.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { Component } from '@angular/core';
@Component({
  imports: [

    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id='container'>
      <div class="container switch-control">
        <div>
          <h3>Customizing Color</h3>
        </div>
        <div>
          <label for="switch1" style="padding: 10px 85px 10px 7px">
Custom color </label>
          <ejs-switch id="switch1" cssClass="bar-color" ></ejs-
switch>
        </div>
        <div>
          <label for='switch2' style="padding: 10px 88px 10px 7px">
Handle color </label>
          <ejs-switch id="switch2" cssClass="handle-color"
[checked]= "true" ></ejs-switch>
        </div>
        <div>
          <label for='switch3' style="padding: 10px 102px 10px
7px"> iOS Switch </label>
          <ejs-switch id="switch3" cssClass="custom-iOS"
[checked]="true" ></ejs-switch>
        </div>
      </div>
    </div>
  `
```

```

})
export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

STYLES.CSS

```

@import 'node_modules/@syncfusion/ej2-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-base/styles/material.css';
@import 'node_modules/@syncfusion/ej2-angular-buttons/styles/material.css';
.e-section-control {
    margin-top: 150px;
}
#container {
    margin-left: 10px;
}
.switch-control div {
    display: flex;
    align-items: center;
}
.switch-control h3 {
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    text-indent: 23px;
}
.switch-control {
    margin: 90px auto;
    width: 240px;
}
.switch-control label {
    -moz-user-select: none;
    user-select: none;
    cursor: pointer;
    font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
    font-size: 13px;
}
#loader {
    color: #008cff;
    height: 40px;
    width: 30%;
    position: absolute;
    top: 45%;
    left: 45%;
}
/* Custom color Switch */
.e-switch-wrapper.bar-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.bar-color:hover .e-switch-inner.e-switch-active .e-switch-on {
    background-color: #4d841d;
}

```

```

    border-color: #4d841d;
  }
  .e-switch-wrapper.bar-color .e-switch-inner,
  .e-switch-wrapper.bar-color .e-switch-off {
    background-color: #e3165b;
    border-color: #e3165b;
  }
  .e-switch-wrapper.bar-color .e-switch-handle {
    background-color: #fff;
  }
  /* handle color Switch */
  .e-switch-wrapper.handle-color .e-switch-handle,
  .e-switch-wrapper.handle-color:not(.e-switch-disabled):hover .e-switch-
  handle:not(.e-switch-active) {
    background-color: #e3165b;
  }
  .e-switch-wrapper.handle-color .e-switch-handle.e-switch-active {
    background-color: #4d841d
  }
  .e-switch-wrapper.handle-color .e-switch-inner.e-switch-active,
  .e-switch-wrapper.handle-color:hover .e-switch-inner.e-switch-active .e-
  switch-on {
    background-color: #fff;
    border-color: #ccc;
  }
  .e-switch-wrapper.handle-color .e-switch-inner,
  .e-switch-wrapper.handle-color .e-switch-off {
    background-color: #fff;
    border-color: #ccc;
  }
  }
  /* iOS Switch */
  .e-switch-wrapper.custom-iOS .e-switch-inner.e-switch-active,
  .e-switch-wrapper.custom-iOS:hover .e-switch-inner.e-switch-active .e-switch-
  on {
    background-color: #3df865;
    border-color: #3df665;
  }
  .e-switch-wrapper.custom-iOS {
    width: 42px;
    height: 24px;
  }
  .e-switch-wrapper.custom-iOS .e-switch-handle {
    width: 20px;
    height: 20px;
  }
  .e-switch-wrapper.custom-iOS .e-switch-handle.e-switch-active {
    margin-left: -22px;
  }
}

```

Enable ripple for switch label in Angular Switch component

By default, label with ripple effect is not available in Switch. You can achieve this by using `rippleMouseHandler` method.

The following example illustrates how to enable ripple effect for labels in Switch component.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component, OnInit } from '@angular/core';
import { rippleMouseHandler } from '@syncfusion/ej2-buttons';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <div id='container'>
      <table class="ripple">
        <tr>
          <td class="lRipple"><label for='switch1'>USB
Tethering</label></td>
          <td>
            <ejs-switch id="switch1" [checked]="true"></ejs-
switch>
          </td>
        </tr>
      </table>
    </div>`
  })
export class AppComponent implements OnInit {
  parentElement: any;
  ngOnInit(): void {
    const rippleHandler = (e: MouseEvent): void => {
      let rippleSpan: Element =
this.parentElement.nextElementSibling.querySelector('.e-ripple-container') as
Element;
      if (rippleSpan) {
        rippleMouseHandler(e, rippleSpan);
      }
      (document as any).querySelector('.lRipple
label')!.addEventListener('mouseup', rippleHandler);
      (document as any).querySelector('.lRipple
label')!.addEventListener('mousedown', rippleHandler);
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enable rtl in Angular Switch component

Switch 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 Switch component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ejs-switch [enableRtl]="true"></ejs-switch></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set disabled state in Angular Switch component

Switch can be disabled by setting the [disabled](#) property to `true`.

The following example illustrates how to disable support in Switch component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <ejs-switch [disabled]="true"></ejs-switch></div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Submit name and value in form in Angular Switch component

The [name](#) attribute of the Switch is used to group Switches. When the Switches are grouped in form, the checked items [value](#) attribute will post to the server on form submit. The disabled and unchecked Switch values will not be sent to the server on form submit.

In the following code snippet, USB and Wi-Fi in the [checked](#) state, and Bluetooth is in disabled state. Values that are in checked state only be sent on form submit.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule, ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { enableRipple } from '@syncfusion/ej2-base'
import { Component } from '@angular/core';
@Component({
  imports: [
    SwitchModule,
    ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <form><table class='size'>
      <tr>
        <td class='lSize'>USB</td>
        <td>
          <ejs-switch name= "Tethering" value= "USB"
[checked]="true" ></ejs-switch>
        </td>
      </tr>
      <tr>
        <td class='lSize'>Wi-Fi</td>
        <td>
          <ejs-switch name= "Hotspot" value= "Wi-Fi"
[checked]="true" ></ejs-switch>
        </td>
      </tr>
      <tr>
        <td class='lSize'>Bluetooth</td>
        <td>
          <ejs-switch name= "Tethering" value= "Bluetooth"
[disabled]="true" ></ejs-switch>
        </td>
      </tr>
      <tr>
        <td>
          <button ejs-button [isPrimary]="true">Submit</button>
        </td>
      </tr>
    </table></form>
  </div>`
```

```

    })
    export class AppComponent { }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change switch state using toggle method in Angular Switch component

This section explains about how to toggle between the switch states using [toggle](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { SwitchModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { SwitchComponent } from '@syncfusion/ej2-angular-buttons';
@Component({
  imports: [
    SwitchModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="e-section-control">
    <!-- To Render Switch. -->
    <ejs-switch #switch onLabel="ON" offLabel="OFF"
    (created)= 'created()' ></ejs-switch>
  </div>`
})
export class AppComponent {
  @ViewChild('switch')
  public switch: SwitchComponent | any;
  public created() {
    this.switch.toggle();
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Switch triggers [change](#) event on every state stage to perform custom operations.

Tab

Getting started with Angular Tab component

This section briefly explains about how to create a simple Tab using Angular by configuring the Tab header content.

Dependencies

The following is the list of dependencies required to use the Angular Tab component in your application.

```
`javascript
|-- @syncfusion/ej2-angular-navigations
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-popups
`,`
```

Setup Angular Environment

You can use [Angular CLI](#) to setup your Angular applications.

To install Angular CLI use the following command.

```
`bash
npm install -g @angular/cli
`,`
```

Create an Angular Application

Start a new Angular application using below Angular CLI command.

```
`bash
ng new my-app
cd my-app
`,`
```

Installing Syncfusion Tab Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-navigations](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-navigations --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-navigations@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-navigations@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-navigations:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering Tab Module

Import Tab module into Angular application(`app.module.ts`) from the package **@syncfusion/ej2-angular-navigations** [`src/app/app.module.ts`].

```
`javascript
import { NgModule }    from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the TabModule for the Tab component
import { TabModule } from '@syncfusion/ej2-angular-navigations';
import { AppComponent } from './app.component';
@NgModule({
  //declaration of ej2-angular-navigations module into NgModule
  imports:    [ BrowserModule, TabModule ],
  declarations: [ AppComponent ],
  bootstrap:  [ AppComponent ]
})
```

```

})
export class AppModule { }
`

```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```

`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-navigations/styles/material.css';
`

```

Add Tab component

Modify the template in `[src/app/app.component.ts]` file to render the Angular Tab component.

Add the Angular Tab by using `<ejs-tab>` selector in **template** section of the `app.component.ts` file.

```

`typescript
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-root',
  // specifies the template string for the Tab component
  template: <ejs-tab> </ejs-tab>
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
  }
}
`

```

Initialize the Tab using JSON items collection

The Tab can be rendered by defining a JSON array. The item is rendered with header [text](#) and [content](#) for each Tab.

```

`typescript
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
/

```

- Adaptive Tab Component

```

*/
@Component({
  selector: 'app-root',
  template: `<ejs-tab id="element">
    <e-tabitems>
    <e-tabitem [header]='headerText[0]' [content]="content0"></e-tabitem>
    <e-tabitem [header]='headerText[1]' [content]="content1"></e-tabitem>
    <e-tabitem [header]='headerText[2]' [content]="content2"></e-tabitem>
    </e-tabitems>
  </ejs-tab>`
})
export class AppComponent {
  public headerText: Object[] = [{ 'text': 'Twitter' }, { 'text': 'Facebook' }, { 'text': 'WhatsApp' }];
  public content0: string = 'Twitter is an online social networking service that enables users to send and read short 140-character ' +
    'messages called "tweets". Registered users can read and post tweets, but those who are unregistered can only read ' +
    'them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is based in San ' +
    'Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, ' +
    'Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, ' +
    'with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion ' +
    'search queries per day.';
  public content1: string = '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 and fellow students Eduardo ' +
    'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website\'s ' +
    'membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford ' +

```

'University. It gradually added support for students at various other universities and later to high-school students.';

public content2: string = 'WhatsApp Messenger is a proprietary cross-platform instant messaging client for smartphones that operates ' +

'under a subscription business model. It uses the Internet to send text messages, images, video, user location and ' +

'audio media messages to other users using standard cellular mobile numbers. As of February 2016, WhatsApp had a user ' +

'base of up to one billion,[10] making it the most globally popular messaging application. WhatsApp Inc., based in ' +

'Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, for approximately US\$19.3 billion.';

}

,

Run the application

After completing the configuration required to render a basic Tab, run the following command to display the output in your default browser.

`shell

npm start

,

The following example illustrates the output in your browser.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * Tab Component
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-tab id="element">
    <e-tabitems>
      <e-tabitem [header]='headerText[0]' [content]="content0"></e-
tabitem>
      <e-tabitem [header]='headerText[1]' [content]="content1"></e-
tabitem>
      <e-tabitem [header]='headerText[2]' [content]="content2"></e-
tabitem>`
})
export class TabComponent {
  headerText: string[] = ['headerText[0]', 'headerText[1]', 'headerText[2]'];
  content0: string = 'content0';
  content1: string = 'content1';
  content2: string = 'content2';
}
```

```

        </e-tabitems>
    </ejs-tab>`
    })
    export class AppComponent {
        public headerText: Object[] = [{ 'text': 'Twitter' }, { 'text':
'Facebook' }, { 'text': 'WhatsApp' }];
        public content0: string = 'Twitter is an online social networking service
that enables users to send and read short 140-character ' +
'messages called "tweets". Registered users can read and post
tweets, but those who are unregistered can only read ' +
'them. Users access Twitter through the website interface, SMS or
mobile device app Twitter Inc. is based in San ' +
'Francisco and has more than 25 offices around the world. Twitter
was created in March 2006 by Jack Dorsey, ' +
'Evan Williams, Biz Stone, and Noah Glass and launched in July
2006. The service rapidly gained worldwide popularity, ' +
'with more than 100 million users posting 340 million tweets a
day in 2012. The service also handled 1.6 billion ' +
'search queries per day.';
        public content1: string = '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 and fellow students Eduardo ' +
'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
founders had initially limited the website\'\'s ' +
'membership to Harvard students, but later expanded it to
colleges in the Boston area, the Ivy League, and Stanford ' +
'University. It gradually added support for students at various
other universities and later to high-school students.';
        public content2: string = 'WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates ' +
'under a subscription business model. It uses the Internet to
send text messages, images, video, user location and ' +
'audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user ' +
'base of up to one billion, [10] making it the most globally
popular messaging application. WhatsApp Inc., based in ' +
'Mountain View, California, was acquired by Facebook Inc. on
February 19, 2014, for approximately US$19.3 billion.';
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

In the above sample code, `#element` is the `id` of the HTML element in a page to which the Tab is initialized.

Initialize the Tab using template

The Tab component can also be rendered using the Angular template `ng-template`. The Tab consists this template support for both its header and content of the item property.

You need to follow the below structure of `ng-template` to render the Tab,

```
`html
<ejs-tab id="element">
<e-tabitems>
<e-tabitem>
<ng-template #headerText> --> Tab header
<div> --> Header Item
</div>
</ng-template>
<ng-template #content> --> Tab content
--> Content Item
</ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>
`
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * Adaptive Tab Component
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-tab id="element">
    <e-tabitems>
      <e-tabitem>
        <ng-template #headerText>
          <div> Twitter </div>
        </ng-template>
        <ng-template #content>
          Twitter is an online social networking service
that enables users to send and read short 140-character messages called
"tweets".
          Registered users can read and post tweets, but
those who are unregistered can only read them. Users access Twitter
        </ng-template>
      </e-tabitem>
    </e-tabitems>
  </ng-template>`
```

```

        through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
        offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
        and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting
        340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per day.
        </ng-template>
    </e-tabitem>
    <e-tabitem>
        <ng-template #headerText>
            <div> Facebook </div>
        </ng-template>
        <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
            Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s membership to Harvard students,
            but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford University. It gradually added support
            for students at various other universities and
later to high-school students.
        </ng-template>
    </e-tabitem>
    <e-tabitem>
        <ng-template #headerText>
            <div> WhatsApp </div>
        </ng-template>
        <ng-template #content>
            WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
            business model. It uses the Internet to send text
messages, images, video, user location and audio media messages to
            other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user base of up to one
billion,[10]
            making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
            by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>`
    })
    export class AppComponent {
    }

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```



```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Initialize the Tab using HTML elements

The Angular Tab component can be rendered based on the given HTML element using `<ejs-tab>` tag.

Header section must be enclosed with in a wrapper element using `e-tab-header` class and corresponding content must be mapped with `e-content` class.

You need to follow the below structure of HTML elements to render the Tab,

```
`html
```

```
<ejs-tab id="element"> --> Root Tab element
```

```
<div class="e-tab-header"> --> Tab header
```

```
<div> --> Header Item
```

```
</div>
```

```
</div>
```

```
<div class="e-content"> --> Tab content
```

```
<div> --> Content Item
```

```
</div>
```

```
</div>
```

```
</ejs-tab>
```

```
,
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-tab id="element">
    <div class="e-tab-header">
      <div>Twitter </div>
      <div>Facebook </div>
      <div>WhatsApp </div>
    </div>
    <div class="e-content">
      <div>
        Twitter is an online social networking service that
        enables users to send and read short 140-character messages called 'tweets'.
      </div>
    </div>
  `
```

```

Registered users can read and post tweets, but those who are unregistered can
only read them. Users access Twitter through the website interface, SMS or
mobile device app Twitter Inc. is based in San Francisco and has more than 25
offices around the world. Twitter was created in March 2006 by Jack Dorsey,
Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The
service rapidly gained worldwide popularity, with more than 100 million users
posting 340 million tweets a day in 2012.The service also handled 1.6 billion
search queries per day.
    </div>
    <div>
        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 and fellow
        students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris
        Hughes.The founders had initially limited the website's membership to Harvard
        students, but later expanded it to colleges in the Boston area, the Ivy
        League, and Stanford University. It gradually added support for students at
        various other universities and later to high-school students.
    </div>
    <div>
        WhatsApp Messenger is a proprietary cross-platform
        instant messaging client for smartphones that operates under a subscription
        business model. It uses the Internet to send text messages, images, video,
        user location and audio media messages to other users using standard cellular
        mobile numbers. As of February 2016, WhatsApp had a user base of up to one
        billion, [10] making it the most globally popular messaging application.
        WhatsApp Inc., based in Mountain View, California, was acquired by Facebook
        Inc. on February 19, 2014, for approximately US$19.3 billion.
    </div>
    </div>
</ejs-tab>`
})
export class AppComponent {
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

You can refer to our [Angular Tab](#) feature tour page for its groundbreaking feature representations. You can also explore our [Angular Tab example](#) that shows you how to render the Tabs in Angular.

See Also

- [How to load tab with DataSource](#)

Adaptive in Angular Tab component

The following section explains about rendering Tab when its width exceeds the viewable area or particularly in a given [width](#).

The available modes are as follows:

- Scrollable
- Popup

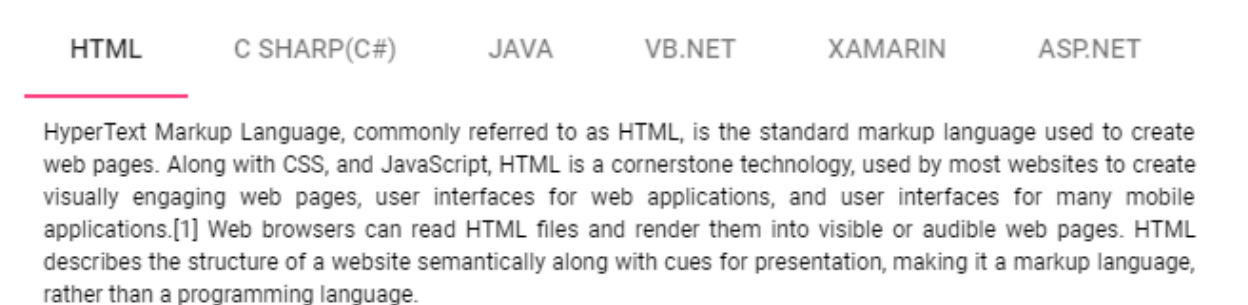
Scrollable

The default overflow mode is Scrollable. Scrollable display mode supports displaying the Tab header items in a single line with horizontal scrolling enabled, when the item overflows to the available space.

- The right and left navigation arrow is added at the start and end of the Tab header through which user can navigate towards overflowed items of the Tab header.
- You can also see the overflowed items using touch and swipe action on the header and content section.
- By default, navigation icon in the left direction is disabled, you can see the overflowed items by moving in the right direction.
- By clicking the arrow or by holding the arrow continuously, you can see the overflowed items.



- In devices the navigation icons are not available. You can touch and swipe to see the overflowed items of the Tab header.



APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
```

```

/**
 * Adaptive Tab Component
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" heightAdjustMode='Auto'
overflowMode='Scrollable' width='300px'>
    <e-tabitems>
      <e-tabitem [header]='headerText[0] '>
        <ng-template #content>
          HyperText Markup Language, commonly referred
to as HTML, is the standard markup language used to create web pages. Along
with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging web
pages, user interfaces for web applications,
and user interfaces for many mobile applications. Web browsers can read
HTML files and render them into visible or
audible web pages. HTML describes the structure of a website semantically
along with cues for presentation, making it a
markup language, rather than a programming language.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1] '>
        <ng-template #content>
          C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led
          by Anders Hejlsberg. The most recent version
is C# 5.0, which was released on August 15, 2012.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[2] '>
        <ng-template #content>
          Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle
Corporation,
          that provides a system for developing
application software and deploying it in a cross-platform computing
environment.
          Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to enterprise servers
and supercomputers. While less common, Java
applets run in secure, sandboxed environments to provide many features
of native applications and can be embedded in
HTML pages.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[3] '>
        <ng-template #content>
          The command-line compiler, VBC.EXE, is
installed as part of the freeware .NET Framework SDK. Mono also includes a
command-line
    </e-tabitems>
  `
})

```

```

        VB.NET compiler. The most recent version is
        VB 2012, which was released on August 15, 2012.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[4] '>
    <ng-template #content>
        Xamarin is a San Francisco, California based
        software company created in May 2011[3] by the engineers that created
        Mono, [4]
        Mono for Android and MonoTouch that are
        cross-platform implementations of the Common Language Infrastructure (CLI)
        and Common Language Specifications (often
        called Microsoft .NET). With a C#-shared codebase, developers can use Xamarin
        tools to write native Android, iOS, and
        Windows apps with native user interfaces and share code across multiple
        platforms.[5]
        Xamarin has over 1 million developers in more
        than 120 countries around the World as of May 2015.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[5] '>
    <ng-template #content>
        ASP.NET is an open-source server-side web
        application framework designed for web development to produce dynamic web
        pages.
        It was developed by Microsoft to allow
        programmers to build dynamic web sites, web applications and web services.
        It was first released in January 2002 with
        version 1.0 of the .NET Framework, and is the successor to Microsoft\'\'s
        Active Server Pages (ASP) technology. ASP.NET
is built on the Common Language Runtime (CLR), allowing programmers
        to write ASP.NET code using any supported
        .NET language. The ASP.NET SOAP extension framework allows ASP.NET components
        to process SOAP messages.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[6] '>
    <ng-template #content>
        The ASP.NET MVC is a web application
        framework developed by Microsoft, which implements the model-view-controller
        (MVC) pattern.
        It is open-source software, apart from the
        ASP.NET Web Forms component which is proprietary. In the later versions
        of ASP.NET, ASP.NET MVC, ASP.NET Web API, and
        ASP.NET Web Pages (a platform using only Razor pages) will merge into
        a unified MVC 6.The project is called ASP.NET
        vNext.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[7] '>
    <ng-template #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.[5]
More recently, however, it has become common in both game development
        and the creation of desktop applications.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>`
}))
export class AppComponent {
    public headerText: Object = [{ text: 'HTML' }, { text: 'C Sharp(C#)' }, {
text: 'Java' }, { text: 'VB.Net' },
        { text: 'Xamarin' }, { text: 'ASP.NET' }, { text: 'ASP.NET MVC' }, {
text: 'JavaScript' }];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Popup

The Popup is the another type of [overflowMode](#) in which the Tab container holds the items that can be placed within the available space. The rest of the overflowing items for which there is no space to fit within the viewing area are moved to overflow popup container.

- The items placed in popup can be viewed by opening the popup with the help of drop-down icon given at the end of the Tab header.
- If the popup height exceeds the height of the visible area, you can scroll through the popup items and select one.

HTML C SHARP(C#) JAVA VB.NET XAMARIN ▾

HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';

```

```

/**
 * Adaptive Tab Component
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" heightAdjustMode='Auto'
overflowMode='Popup' width='300px'>
    <e-tabitems>
      <e-tabitem [header]='headerText[0] '>
        <ng-template #content>
          HyperText Markup Language, commonly referred
to as HTML, is the standard markup language used to create web pages. Along
with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging web
pages, user interfaces for web applications,
and user interfaces for many mobile applications. Web browsers can read
HTML files and render them into visible or
audible web pages. HTML describes the structure of a website semantically
along with cues for presentation, making it a
markup language, rather than a programming language.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1] '>
        <ng-template #content>
          C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led
          by Anders Hejlsberg. The most recent version
is C# 5.0, which was released on August 15, 2012.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[2] '>
        <ng-template #content>
          Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle
Corporation,
          that provides a system for developing
application software and deploying it in a cross-platform computing
environment.
          Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to enterprise servers
and supercomputers. While less common, Java
applets run in secure, sandboxed environments to provide many features
of native applications and can be embedded in
HTML pages.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[3] '>
        <ng-template #content>
          The command-line compiler, VBC.EXE, is
installed as part of the freeware .NET Framework SDK. Mono also includes a
command-line
    </e-tabitems>
  `
})

```

```

        VB.NET compiler. The most recent version is
        VB 2012, which was released on August 15, 2012.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[4] '>
    <ng-template #content>
        Xamarin is a San Francisco, California based
        software company created in May 2011[3] by the engineers that created
        Mono, [4]
        Mono for Android and MonoTouch that are
        cross-platform implementations of the Common Language Infrastructure (CLI)
        and Common Language Specifications (often
        called Microsoft .NET). With a C#-shared codebase, developers can use Xamarin
        tools to write native Android, iOS, and
        Windows apps with native user interfaces and share code across multiple
        platforms.[5]
        Xamarin has over 1 million developers in more
        than 120 countries around the World as of May 2015.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[5] '>
    <ng-template #content>
        ASP.NET is an open-source server-side web
        application framework designed for web development to produce dynamic web
        pages.
        It was developed by Microsoft to allow
        programmers to build dynamic web sites, web applications and web services.
        It was first released in January 2002 with
        version 1.0 of the .NET Framework, and is the successor to Microsoft\'\'s
        Active Server Pages (ASP) technology. ASP.NET
is built on the Common Language Runtime (CLR), allowing programmers
        to write ASP.NET code using any supported
        .NET language. The ASP.NET SOAP extension framework allows ASP.NET components
        to process SOAP messages.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[6] '>
    <ng-template #content>
        The ASP.NET MVC is a web application
        framework developed by Microsoft, which implements the model-view-controller
        (MVC) pattern.
        It is open-source software, apart from the
        ASP.NET Web Forms component which is proprietary. In the later versions
        of ASP.NET, ASP.NET MVC, ASP.NET Web API, and
        ASP.NET Web Pages (a platform using only Razor pages) will merge into
        a unified MVC 6.The project is called ASP.NET
        vNext.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[7] '>
    <ng-template #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.[5]
More recently, however, it has become common in both game development
        and the creation of desktop applications.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>`
}))
export class AppComponent {
    public headerText: Object = [{ text: 'HTML' }, { text: 'C Sharp(C#)' }, {
text: 'Java' }, { text: 'VB.Net' },
        { text: 'Xamarin' }, { text: 'ASP.NET' }, { text: 'ASP.NET MVC' }, {
text: 'JavaScript' }];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [How to prevent content swipe selection](#)
- [Collapsible Tab](#)

Header in Angular Tab component

This section explains about modifying the style of Tab header, and configuring its icons and positions.

Styles

You can customize header styles by adding predefined classes in the Tab root element. The pre-defined CSS class names are as follows:

- **e-fill**: The Selected Tab header background is set as solid fill.
- **e-background**: Tab header has a solid fill background, and the selected header has a highlighted border.
- **e-background e-accent**: Tab header has a solid fill background, and the selected header has a highlighted border with accent color.

If the above custom style classes are not included in the root element, the default style is applied to the Tab items.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';

```

```

import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-angular-dropdowns';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TabModule, DropDownListModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template url path
  templateUrl: 'app/app.component.html'
})
export class AppComponent {
  @ViewChild('element') tabObj?: TabComponent;
  @ViewChild('headerStyles') listObj?: DropDownListComponent;
  public headerText: Record<string, any>[] = [{ 'text': 'Twitter' }, { 'text': 'Facebook' }, { 'text': 'WhatsApp' }];
  public headerData: Object[] = [
    { Id: 'header1', headerStyle: 'default', text: 'Default' },
    { Id: 'header2', headerStyle: 'fill', text: 'Fill' },
    { Id: 'header3', headerStyle: 'background', text: 'Background' },
    { Id: 'header4', headerStyle: 'accent', text: 'Accent' }
  ];
  public fields: Object = { text: 'text', value: 'headerStyle' };
  public height: string = '220px';
  public value: string = 'default';
  public onChange(args: ChangeEventArgs): void {
    this.removeStyleClass();
    if (this.listObj?.value === 'fill') {
      (this.tabObj as TabComponent).element.classList.add('e-fill');
    } else if (this.listObj?.value === 'background') {
      (this.tabObj as TabComponent).element.classList.add('e-background');
    } else if (this.listObj?.value === 'accent') {
      (this.tabObj as TabComponent).element.classList.add('e-background');
      (this.tabObj as TabComponent).element.classList.add('e-accent');
    }
  }
  public removeStyleClass(): void {
    (this.tabObj as TabComponent).element.classList.remove('e-fill');
    (this.tabObj as TabComponent).element.classList.remove('e-background');
    (this.tabObj as TabComponent).element.classList.remove('e-accent');
  }
}

```

APP.COMPONENT.HTML

```

<div id="wrapper" style='margin-top: 20px'>
  <div id='content' style="margin: 0px auto">
    <div id="default" style="padding-top:20px;width:250px">
      <div class='row'>
        <div>
          <label> Header Style </label>
        </div><br />
      </div>
    </div>
  </div>
</div>

```

```

        <div>
            <ejs-dropdownlist id='headerStyles' #headerStyles
[dataSource]='headerData'
                (change)='onChange($event)' [value]='value'
[fields]='fields' [popupHeight]='height'>
            </ejs-dropdownlist>
        </div>
    </div>
</div>
<br />
<div>
    <ejs-tab id="element" #element>
        <e-tabitems>
            <e-tabitem [header]='headerText[0] '>
                <ng-template #content>
                    Twitter is an online social networking service
that enables users to send and read short
140-character messages called "tweets".
                    Registered users can read and post tweets, but
those who are unregistered can only read
                    them. Users access Twitter
                    through the website interface, SMS or mobile
device app Twitter Inc. is based in San
                    Francisco and has more than 25
                    offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams,
                    Biz Stone, and Noah Glass
                    and launched in July 2006. The service rapidly
gained worldwide popularity, with more than
                    100 million users posting
                    340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per
                    day.
                </ng-template>
            </e-tabitem>
            <e-tabitem [header]='headerText[1] '>
                <ng-template #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 and fellow students Eduardo
                    Saverin, Andrew McCollum,
                    Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s
                    membership to Harvard students,
                    but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford
                    University. It gradually added support
                    for students at various other universities and
later to high-school students.
                </ng-template>
            </e-tabitem>
            <e-tabitem [header]='headerText[2] '>
                <ng-template #content>
                    WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones

```

```

        that operates under a subscription
        business model. It uses the Internet to send text
messages, images, video, user location and
        audio media messages to
        other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user
        base of up to one billion, [10]
        making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain
        View, California, was acquired
        by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>
</div>
</div>
</div>

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Icon positions

You can customize the position of the Tab header icons using the [iconPosition](#) property. This property depends on the header items [iconCss](#) property. By default, Tab header icon is placed on left position. The position values are as follows:

- **Left:** Icon is placed on the left of the Tab header item.
- **Right:** Icon is placed on the right of the Tab header item.
- **Top:** Icon is placed on the top of the Tab header item.
- **Bottom:** Icon is placed on the bottom of the Tab header item.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabComponent, TabItemsDirective, TabItemDirective } from
'@syncfusion/ej2-angular-navigations'
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { DropDownListModel, ChangeEventArgs } from '@syncfusion/ej2-
dropdowns';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
/**
 * Adaptive Tab Component
 */

```

```

@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template url path
  template: ` <div id="wrapper" style='margin-top: 20px'><div id='content'
style="margin: 0px auto">
  <div id="default" style="padding-top:20px;width:250px">
    <div class='row'>
      <div>
        <label> Icon Position </label>
      </div><br/>
      <div>
        <ejs-dropdownlist id='iconPosition' #iconPosition
[dataSource]='positionData' (change)='onChange($event)' [value]='value'
[fields]='fields' [popupHeight]='height'></ejs-dropdownlist>
      </div>
    </div>
  </div>
  <br/><div><ejs-tab id="element" #element>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]'>
        <ng-template #content>
          Twitter is an online social networking
service that enables users to send and read short 140-character messages
called "tweets".

          Registered users can read and post tweets,
but those who are unregistered can only read them. Users access Twitter
          through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
          offices around the world. Twitter was created
in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
          and launched in July 2006. The service
rapidly gained worldwide popularity, with more than 100 million users posting
          340 million tweets a day in 2012.The service
also handled 1.6 billion search queries per day.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1]'>
        <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
          Dustin Moskovitz and Chris Hughes.The
founders had initially limited the website's membership to Harvard
students,

          but later expanded it to colleges in the
Boston area, the Ivy League, and Stanford University. It gradually added
support
          for students at various other universities
and later to high-school students.
        </ng-template>
      </e-tabitem>
    </e-tabitems>
  </div>
</div>`
})

```

```

        <e-tabitem [header]='headerText[2] '>
            <ng-template #content>
                WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
                business model. It uses the Internet to send
text messages, images, video, user location and audio media messages to
                other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user base of up to one
billion, [10]
                making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
                by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
            </ng-template>
        </e-tabitem>
    </e-tabitems>
</ejs-tab>
</div>
</div>
</div>`
    ))
export class AppComponent {
    @ViewChild('element') tabObj?: TabComponent;
    @ViewChild('iconPosition') listObj?: DropDownListComponent;
    public positionData: Object[] = [
        { position: 'left', text: 'Left' },
        { position: 'right', text: 'Right' },
        { position: 'top', text: 'Top' },
        { position: 'bottom', text: 'Bottom' }
    ];
    public fields: Object = { text: 'text', value: 'position' };
    public height: string = '220px';
    public value: string = 'left';
    public onChange(ChangeEventArgs: any): void {
        let items: any = (this.tabObj as TabComponent).items;
        for(let i: number = 0; i < items.length; i++) {
            items[i].header.iconPosition = this.listObj?.value;
        }
    }
    public headerText: Object = [{ 'text': 'Twitter', 'iconCss': 'e-twitter'
}, { 'text': 'Facebook', 'iconCss': 'e-facebook' }, { 'text': 'WhatsApp',
'iconCss': 'e-whatsapp' }];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [How to customize selected tab styles](#)

Localization in Angular Tab component

Localization library allows to localize the default text content of the Tab to different cultures using the [locale](#) property. In Tab, the close button's tooltip text alone will be localized based on culture. The close button is shown on tab header when enabled [showCloseButton](#) property.

| Locale key | en-US (default) |

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

| closeButtonTitle | 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 Tab and change the close button's tooltip text.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild, OnInit, ElementRef } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { L10n } from '@syncfusion/ej2-base';
import { EmitType } from '@syncfusion/ej2-base';
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-tab id="element" locale="fr-BE" [showCloseButton]='true'>
      <e-tabitems>
        <e-tabitem [header]='headerText[0]' [content]="content0"></e-
tabitem>
        <e-tabitem [header]='headerText[1]' [content]="content1"></e-
tabitem>
        <e-tabitem [header]='headerText[2]' [content]="content2"></e-
tabitem>
      </e-tabitems>
    </ejs-tab>
  `
})
export class AppComponent implements OnInit {
  ngOnInit() {
    // Load French culture for Tab close button tooltip text
    L10n.load({
      'fr-BE': {
        'tab': {
          'closeButtonTitle': "Fermer"
        }
      }
    });
  }
}
```

```

    public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
    }, { 'text': 'WhatsApp' }];
    public content0: string = 'Twitter is an online social networking service
    that enables users to send and read short 140-character ' +
    'messages called "tweets". Registered users can read and post
    tweets, but those who are unregistered can only read ' +
    'them. Users access Twitter through the website interface, SMS or
    mobile device app Twitter Inc. is based in San ' +
    'Francisco and has more than 25 offices around the world. Twitter
    was created in March 2006 by Jack Dorsey, ' +
    'Evan Williams, Biz Stone, and Noah Glass and launched in July
    2006. The service rapidly gained worldwide popularity, ' +
    'with more than 100 million users posting 340 million tweets a
    day in 2012. The service also handled 1.6 billion ' +
    'search queries per day.';
    public content1: string = '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 and fellow students Eduardo ' +
    'Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
    founders had initially limited the website's ' +
    'membership to Harvard students, but later expanded it to
    colleges in the Boston area, the Ivy League, and Stanford ' +
    'University. It gradually added support for students at various
    other universities and later to high-school students.';
    public content2: string = 'WhatsApp Messenger is a proprietary cross-
    platform instant messaging client for smartphones that operates ' +
    'under a subscription business model. It uses the Internet to
    send text messages, images, video, user location and ' +
    'audio media messages to other users using standard cellular
    mobile numbers. As of February 2016, WhatsApp had a user ' +
    'base of up to one billion, [10] making it the most globally
    popular messaging application. WhatsApp Inc., based in ' +
    'Mountain View, California, was acquired by Facebook Inc. on
    February 19, 2014, for approximately US$19.3 billion.';
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Orientation in Angular Tab component

This section explains about modifying the position and modes of Tab header.

It allows placing the header section inside the Tab component at different positions by using the [headerPlacement](#) property. The available positions are as follows:

- **Top:** Tab header items can be arranged horizontally, and their content can be placed after the header.
- **Bottom:** Tab header items can be arranged horizontally, and their content can be placed before the header.

- **Left:** Tab header items can be arranged vertically, and their content can be placed after the header.
- **Right:** Tab header items can be arranged vertically, and their content can be placed before the header.

It is also adaptable to the available space when the tab items exceed the view space. You can customize the modes by using [overflowMode](#) property. The available modes are as follows:

- Scrollable
- Popup

APP.COMPONENT.HTML

```
<div id='container'>
  <div id="default" style="padding-top:20px">
    <div class='row' style="float:left">
      <label> Header Position </label>
      <div>
        <ejs-dropdownlist #dropDownPosition
(change)="positionChange($event)" id='headerPosition'
[dataSource]='PositionDB' [value]='positionValue' [fields]='positionFields'
index='0'>
          </ejs-dropdownlist>
        </div>
      </div>
      <div class='row' style="float:right">
        <label> Mode </label>
        <div>
          <ejs-dropdownlist #dropDownMode (change)="modeChange($event)"
id='Mode' [dataSource]='ModeDB' [value]='modeValue' [fields]='modeFields'
index='0'>
            </ejs-dropdownlist>
          </div>
        </div>
      </div>
    <div>
      <ejs-tab #element id="element" heightAdjustMode='None' height='150px'
width='700px'>
        <e-tabitems>
          <e-tabitem [header]='headerText[0]' [content]="content0"></e-
tabitem>
          <e-tabitem [header]='headerText[1]' [content]="content1"></e-
tabitem>
          <e-tabitem [header]='headerText[2]' [content]="content2"></e-
tabitem>
          <e-tabitem [header]='headerText[3]' [content]="content3"></e-
tabitem>
          <e-tabitem [header]='headerText[4]' [content]="content4"></e-
tabitem>
          <e-tabitem [header]='headerText[5]' [content]="content5"></e-
tabitem>
          <e-tabitem [header]='headerText[6]' [content]="content6"></e-
tabitem>
          <e-tabitem [header]='headerText[7]' [content]="content7"></e-
tabitem>
        </e-tabitems>
      </div>
    </div>
  </div>
</div>
```

```

        </ejs-tab>
        <br/>
        <br/>
    </div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabAllModule } from '@syncfusion/ej2-angular-navigations'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild, OnInit, ViewEncapsulation } from
 '@angular/core';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { rippleEffect } from '@syncfusion/ej2-base';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TabAllModule, DropDownListAllModule
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: 'app/app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent implements OnInit {
  ngOnInit(): void {
  }
  @ViewChild('element') tabObj?: TabComponent;
  @ViewChild('dropDownMode') dropMode?: DropDownListComponent;
  @ViewChild('dropDownPosition') dropPosition?: DropDownListComponent;
  public ModeDB = [{ id: 'scrollable', mode: 'Scrollable' },
    { id: 'popup', mode: 'Popup' }];
  public PositionDB = [{ id: 'top', position: 'Top' }, { id: 'bottom',
position: 'Bottom' },
    { id: 'left', position: 'Left' }, { id: 'right', position: 'Right'
}];
  public positionFields: Object = { text: 'position', value: 'id' };
  public modeFields: Object = { text: 'mode', value: 'id' };
  public positionValue: string = 'top';
  public modeValue: string = 'scrollable';
  public headerText: Object = [{ 'text': 'HTML' }, { 'text': 'C Sharp(C#)'
},{ 'text': 'Java' }, { 'text': 'VB.Net' }, { 'text': 'Xamarin' }, { 'text':
'ASP.NET' },
    { 'text': 'ASP.NET MVC' }, { 'text': 'JavaScript' }];
  public content0: string = 'HyperText Markup Language, commonly referred
to as HTML, is the standard markup language used to create web ' +
    'pages. Along with CSS, and JavaScript, HTML is a cornerstone
technology, used by most websites to create visually ' +
    'engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web ' +
    'browsers can read HTML files and render them into visible or
audible web pages. HTML describes the structure of a ' +
    'website semantically along with cues for presentation,
making it a markup language, rather than a programming language.';

```

```

    public content1: string = 'C# is intended to be a simple, modern,
    general-purpose, object-oriented programming language. Its development ' +
    'team is led by Anders Hejlsberg. The most recent version is
    C# 5.0, which was released on August 15, 2012.';
    public content2: string = 'Java is a set of computer software and
    specifications developed by Sun Microsystems, later acquired by Oracle ' +
    'Corporation, that provides a system for developing
    application software and deploying it in a cross-platform computing ' +
    'environment. Java is used in a wide variety of computing
    platforms from embedded devices and mobile phones to ' +
    'enterprise servers and supercomputers. While less common,
    Java applets run in secure, sandboxed environments to ' +
    'provide many features of native applications and can be
    embedded in HTML pages.';
    public content3: string = 'The command-line compiler, VBC.EXE, is
    installed as part of the freeware .NET Framework SDK. Mono also ' +
    'includes a command-line VB.NET compiler. The most recent
    version is VB 2012, which was released on August 15, 2012.';
    public content4: string = 'Xamarin is a San Francisco, California based
    software company created in May 2011[3] by the engineers that ' +
    'created Mono,[4] Mono for Android and MonoTouch that are
    cross-platform implementations of the Common Language ' +
    'Infrastructure (CLI) and Common Language Specifications
    (often called Microsoft .NET). With a C#-shared codebase, ' +
    'developers can use Xamarin tools to write native Android,
    iOS, and Windows apps with native user interfaces and share ' +
    'code across multiple platforms.[5] Xamarin has over 1
    million developers in more than 120 countries around the World ' +
    'as of May 2015.';
    public content5: string = 'ASP.NET is an open-source server-side web
    application framework designed for web development to produce ' +
    'dynamic web pages. It was developed by Microsoft to allow
    programmers to build dynamic web sites, web applications ' +
    'and web services. It was first released in January 2002 with
    version 1.0 of the .NET Framework, and is the successor ' +
    'to Microsoft\'s Active Server Pages (ASP) technology.
    ASP.NET is built on the Common Language Runtime (CLR), allowing ' +
    'programmers to write ASP.NET code using any supported .NET
    language. The ASP.NET SOAP extension framework allows ' +
    'ASP.NET components to process SOAP messages.';
    public content6: string = 'The ASP.NET MVC is a web application framework
    developed by Microsoft, which implements the ' +
    'model-view-controller (MVC) pattern. It is open-source
    software, apart from the ASP.NET Web Forms component which is ' +
    'proprietary. In the later versions of ASP.NET, ASP.NET MVC,
    ASP.NET Web API, and ASP.NET Web Pages (a platform using ' +
    'only Razor pages) will merge into a unified MVC 6.The
    project is called ASP.NET vNext.';
    public content7: string = '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.[5] More recently, however, it has become common in ' +
    'both game development and the creation of desktop
    applications.';
    public modeChange(e: Object): void {

```

```

        (this.tabObj as TabComponent).overflowMode = this.dropMode?.text as
any;
        (this.tabObj as TabComponent).dataBind();
    }
    public positionChange(e: Object): void {
        (this.tabObj as TabComponent).headerPlacement =
this.dropPosition?.text as any;
        (this.tabObj as TabComponent).dataBind();
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Drag and drop in Angular Tab component

The Tab component allows you to drag and drop any item by setting [allowDragAndDrop](#) to **true**. Items can be reordered to any place by dragging and dropping them onto the desired location.

- If you need to prevent dragging action for a particular item, the [onDragStart](#) event can be used which will trigger when the item drag is started. If you need to prevent dropping action for a particular item, the [dragged](#) event can be used which will trigger when the drag action is stopped.
- The [dragArea](#) defines the area in which the draggable element movement will be occurring. Outside that area will be restricted for the draggable element movement.
- The [onDragStart](#) event will be triggered before dragging the item from Tab.
- The [dragging](#) event will be triggered when the Tab item is being dragged.
- The [dragged](#) event will be triggered when the Tab item is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TabAllModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
/**
 * Tab Component
 */
@Component({
  imports: [
    FormsModule,
    TabAllModule,
  ],
  standalone: true,
  selector: 'app-container',

```

```

    template: `<div id='tabpanel'><ejs-tab id="draggableTab"
heightAdjustMode='Auto' [allowDragAndDrop]='true' dragArea='#tabpanel'>
    <e-tabitems>
        <e-tabitem [header]='headerText[0]' [content]="content0"></e-
tabitem>
        <e-tabitem [header]='headerText[1]' [content]="content1"></e-
tabitem>
        <e-tabitem [header]='headerText[2]' [content]="content2"></e-
tabitem>
        <e-tabitem [header]='headerText[3]' [content]="content3"></e-
tabitem>
    </e-tabitems>
    </ejs-tab></div>`
  })
  export class AppComponent {
    public headerText: Object = [{ 'text': 'India' }, { 'text': 'Australia'
}, { 'text': 'USA' }, { 'text': 'France' }];
    public content0: string = 'India officially the Republic of India, is a
country in South Asia. It is the seventh-largest country by area, the second-
most populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the Arabian
Sea on the south-west, and the Bay of Bengal on the south-east, it shares
land borders with Pakistan to the west;China, Nepal, and Bhutan to the north-
east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in
the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and
Nicobar Islands share a maritime border with Thailand and Indonesia.';
    public content1: string = 'Australia, officially the Commonwealth of
Australia, is a country comprising the mainland of the Australian continent,
the island of Tasmania and numerous smaller islands. It is the world sixth-
largest country by total area. Neighboring countries include Indonesia, East
Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.';
    public content2: string = 'The United States of America (USA or U.S.A.),
commonly called the United States (US or U.S.) and America, is a federal
republic consisting of fifty states and a federal district. The 48 contiguous
states and the federal district of Washington, D.C. are in central North
America between Canada and Mexico. The state of Alaska is west of Canada and
east of Russia across the Bering Strait, and the state of Hawaii is in the
mid-North Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.';
    public content3: string = 'France, officially the French Republic is a
sovereign state comprising territory in western Europe and several overseas
regions and territories. The European part of France, called Metropolitan
France, extends from the Mediterranean Sea to the English Channel and the
North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679
square kilo metres and as of August 2015 has a population of 67 million,
counting all the overseas departments and territories.';
  }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Drag and drop item between tabs

It is possible to drag and drop the tab items between two tabs, by manually saving those dropped items as new tab item data through the `addTab` method of Tab and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tab control as an external source, and the item from the tab component is dragged and dropped onto another Tab. Therefore, it is necessary to use the `onDragStart` and `dragged` event of the Tab component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged item through `removeTab` method of Tab using the dragged item index.

APP.COMPONENT.HTML

```
<div id='tabpanel'>
  <ejs-tab #firstTabObj id="firstTab" heightAdjustMode='Auto'
[allowDragAndDrop]='true' dragArea='#tabpanel'
  (onDragStart)='firstTabdragStart($event)'
  (dragged)='firstTabDragStop($event)'>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]' [content]="content0"></e-tabitem>
      <e-tabitem [header]='headerText[1]' [content]="content1"></e-tabitem>
      <e-tabitem [header]='headerText[2]' [content]="content2"></e-tabitem>
      <e-tabitem [header]='headerText[3]' [content]="content3"></e-tabitem>
    </e-tabitems>
  </ejs-tab>
  <ejs-tab #secondTabObj id="secondTab" heightAdjustMode='Auto'
[allowDragAndDrop]='true' dragArea='#tabpanel'
  (onDragStart)='secondTabDragStart($event)'
  (dragged)='secondTabDragStop($event)'>
    <e-tabitems>
      <e-tabitem [header]='headerText[4]' [content]="content4"></e-tabitem>
      <e-tabitem [header]='headerText[5]' [content]="content5"></e-tabitem>
      <e-tabitem [header]='headerText[6]' [content]="content6"></e-tabitem>
      <e-tabitem [header]='headerText[7]' [content]="content7"></e-tabitem>
    </e-tabitems>
  </ejs-tab>
</div>
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TabAllModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { TabComponent, DragEventArgs } from '@syncfusion/ej2-angular-navigations';
import { isNullOrUndefined } from "@syncfusion/ej2-base";
/**
 * Tab Component
 */
@Component({
  imports: [
    FormsModule,
    TabAllModule,
```

```

    ],
    standalone: true,
    selector: 'app-container',
    templateUrl: './app.component.html',
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent {
    @ViewChild('firstTabObj') public firstTabObj?: TabComponent;
    @ViewChild('secondTabObj') public secondTabObj?: TabComponent;
    public headerText: Object[] = [{ 'text': 'India' }, { 'text': 'Australia' },
    { 'text': 'USA' }, { 'text': 'France' }, { 'text': 'HTML' }, { 'text': 'C Sharp(C#)' }, { 'text': 'Java' }, { 'text': 'VB.Net' }]];
    public firstTabitem: Object[] = [];
    public secondTabitem: Object[] = [];
    public dragItemIndex?: number;
    public dragItemContainer?: HTMLElement;
    public content0: string = 'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west;China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.';
    public content1: string = 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.';
    public content2: string = 'The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of Hawaii is in the mid-North Pacific. The country also has five populated and nine unpopulated territories in the Pacific and the Caribbean.';
    public content3: string = 'France, officially the French Republic is a sovereign state comprising territory in western Europe and several overseas regions and territories. The European part of France, called Metropolitan France, extends from the Mediterranean Sea to the English Channel and the North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo metres and as of August 2015 has a population of 67 million, counting all the overseas departments and territories.';
    public content4: string = 'HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.';
  }

```

```

    public content5: string = 'C# is intended to be a simple, modern,
    general-purpose, object-oriented programming language. Its development team
    is led by Anders Hejlsberg. The most recent version is C# 5.0, which was
    released on August 15, 2012.';

    public content6: string = 'Java is a set of computer software and
    specifications developed by Sun Microsystems, later acquired by Oracle
    Corporation, that provides a system for developing application software and
    deploying it in a cross-platform computing environment. Java is used in a
    wide variety of computing platforms from embedded devices and mobile phones
    to enterprise servers and supercomputers. While less common, Java applets run
    in secure, sandboxed environments to provide many features of native
    applications and can be embedded in HTML pages.';

    public content7: string = 'The command-line compiler, VBC.EXE, is
    installed as part of the freeware .NET Framework SDK. Mono also includes a
    command-line VB.NET compiler. The most recent version is VB 2012, which was
    released on August 15, 2012.';

    firstTabDragStart(args: DragEventArgs): void {
        this.firstTabitem = [(this.firstTabObj as
        TabComponent).items[args.index]];
        args.draggedItem.style.visibility = 'hidden';
        this.dragItemContainer = <HTMLElement>args.draggedItem.closest('.e-
        tab');
    }

    firstTabDragStop(args: DragEventArgs): void {
        if (!isNullOrUndefined((args.target as HTMLElement).closest('.e-tab')
        as Element) && !(this.dragItemContainer as
        HTMLElement).isSameNode(args.target.closest('.e-tab'))) {
            args.cancel = true;
            let TabElement: HTMLElement =
            <HTMLElement>args.target.closest('.e-tab');
            let dropItem: HTMLElement = <HTMLElement>args.target.closest('.e-
            toolbar-item');
            if (TabElement != null && dropItem != null) {
                const childrenArray = Array.from((this.firstTabObj as
                TabComponent).element.children[0].children[0].children);
                const toolbarItem: Element[] = childrenArray.filter(el =>
                el.classList.contains('e-toolbar-item'));
                this.dragItemIndex = toolbarItem.indexOf(args.draggedItem);
                let dropItemContainer: Element = args.target.closest('.e-
                toolbar-items') as Element;
                let dropChildArray: Element[] =
                (Array.from(dropItemContainer.children)).filter(el =>
                el.classList.contains('e-toolbar-item'));
                let dropItemIndex: number = (dropItemContainer != null) ?
                dropChildArray.indexOf(dropItem) as number : '' as any;
                (this.secondTabObj as TabComponent).addTab(this.firstTabitem,
                dropItemIndex);
                (this.firstTabObj as
                TabComponent).removeTab(this.dragItemIndex);
            }
        }
    }

    secondTabDragStart(args: DragEventArgs): void {
        this.secondTabitem = [(this.secondTabObj as
        TabComponent).items[args.index]];
        args.draggedItem.style.visibility = 'hidden';

```



```

        this.dragItemContainer = <HTMLElement>args.draggedItem.closest('.e-
tab');
    }
    secondTabDragStop(args: DragEventArgs): void {
        if (!isNullOrUndefined(args.target.closest('.e-tab') as Element) &&
        !(this.dragItemContainer as HTMLElement).isSameNode(args.target.closest('.e-
tab'))) {
            args.cancel = true;
            let TabElement: HTMLElement =
<HTMLElement>args.target.closest('.e-tab');
            let dropItem: HTMLElement = <HTMLElement>args.target.closest('.e-
toolbar-item');
            if (TabElement != null && dropItem != null) {
                const childrenArray = Array.from((this.secondTabObj as
TabComponent).element.children[0].children[0].children);
                const toolbarItem: Element[] = childrenArray.filter(el =>
el.classList.contains('e-toolbar-item'));
                this.dragItemIndex = toolbarItem.indexOf(args.draggedItem);
                let dropItemContainer: Element = args.target.closest('.e-
toolbar-items') as Element;
                let dropChildArray: Element[] =
(Array.from(dropItemContainer.children)).filter(el =>
el.classList.contains('e-toolbar-item'));
                let dropItemIndex: number = (dropItemContainer != null) ?
dropChildArray.indexOf(dropItem) as number : '' as any;
                (this.firstTabObj as TabComponent).addTab(this.secondTabitem,
dropItemIndex);
                (this.secondTabObj as
TabComponent).removeTab(this.dragItemIndex);
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Drag and drop items to external source

It is possible to drag and drop the items to any of the external sources from the Tab, by manually saving those dropped items as new node data through the `addNodes` method of Treeview and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tree view control as an external source, and the item from the tab component is dragged and dropped onto the child nodes of the tree view component. Therefore, it is necessary to use the `dragged` event of the Tab component, where we can form an event object and save it using the `addNodes` method of the Treeview and remove the dragged item through the `removeTab` method of Tab using the dragged item index.

APP.COMPONENT.HTML

```

<div id='tabpanel'>
  <ejs-tab #tabObj id="draggableTab" heightAdjustMode='Auto'
[allowDragAndDrop]='true' dragArea='#tabpanel'
  (created)='onTabCreate()' (dragged)='tabDragStop($event)'>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]' [content]="content0"></e-tabitem>
      <e-tabitem [header]='headerText[1]' [content]="content1"></e-tabitem>
      <e-tabitem [header]='headerText[2]' [content]="content2"></e-tabitem>
      <e-tabitem [header]='headerText[3]' [content]="content3"></e-tabitem>
    </e-tabitems>
  </ejs-tab>
  <ejs-treeview #treeObj id='draggableTreeview' [fields]='field'
cssClass='treeview-external-drop-tab'>
  </ejs-treeview>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TabAllModule, TreeViewModel } from '@syncfusion/ej2-angular-
navigations'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { TabComponent, DragEventArgs, TabItemModel, TreeViewComponent,
HeaderModel } from '@syncfusion/ej2-angular-navigations';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
/**
 * Tab Component
 */
@Component({
  imports: [
    FormsModule,
    TabAllModule,

    TreeViewModel
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('tabObj') tabObj?: TabComponent;
  @ViewChild('treeObj') treeObj?: TreeViewComponent;
  public headerText: Object[] = [{ 'text': 'India' }, { 'text': 'Australia'
}, { 'text': 'USA' }, { 'text': 'France' }];
  public content0: string = 'India officially the Republic of India, is a
country in South Asia. It is the seventh-largest country by area, the second-
most populous country with over 1.2 billion people, and the most populous
democracy in the world. Bounded by the Indian Ocean on the south, the Arabian
Sea on the south-west, and the Bay of Bengal on the south-east, it shares
land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-
east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in
the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and
Nicobar Islands share a maritime border with Thailand and Indonesia.';

```

```

    public content1: string = 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.';

    public content2: string = 'The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of Hawaii is in the mid-North Pacific. The country also has five populated and nine unpopulated territories in the Pacific and the Caribbean.';

    public content3: string = 'France, officially the French Republic is a sovereign state comprising territory in western Europe and several overseas regions and territories. The European part of France, called Metropolitan France, extends from the Mediterranean Sea to the English Channel and the North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo metres and as of August 2015 has a population of 67 million, counting all the overseas departments and territories.';

    public field: Object = {
      dataSource: [
        { text: 'Hennessey Venom', id: 'list-01' },
        { text: 'Bugatti Chiron', id: 'list-02' },
        { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
        { text: 'SSC Ultimate Aero', id: 'list-04' },
        { text: 'Koenigsegg CCR', id: 'list-05' },
        { text: 'McLaren F1', id: 'list-06' },
        { text: 'Aston Martin One- 77', id: 'list-07' },
        { text: 'Jaguar XJ220', id: 'list-08' },
        { text: 'McLaren P1', id: 'list-09' },
        { text: 'Ferrari LaFerrari', id: 'list-10' },
      ],
      id: "id", text: "text"
    }
    public i: number = 0;
    onTabCreate(): void {
      let tabElement: HTMLElement = document.getElementById('draggableTab')
as HTMLElement;
      if (!isNullOrUndefined(tabElement)) {
        (this.tabObj as
TabComponent).element.children[0].classList.add('e-droppable');
        (this.tabObj as
TabComponent).element.children[1].classList.add('tab-content');
      }
    }
    tabDragStop(args: DragEventArgs): void {
      const childrenArray = Array.from((this.tabObj as
TabComponent).element.children[0].children[0].children);
      const toolbarItem: Element[] = childrenArray.filter(el =>
el.classList.contains('e-toolbar-item'));
      let dragTabIndex: number = toolbarItem.indexOf(args.draggedItem);
      let dragItem: TabItemModel = (this.tabObj as
TabComponent).items[dragTabIndex];
      let dropNode: HTMLElement =
<HTMLElement>args.target.closest('#draggableTreeview .e-list-item');

```

```

    if (dropNode !== null && !args.target.closest('#draggableTab .e-
toolbar-item')) {
        args.cancel = true;
        let dropContainer = Array.from((this.treeObj as
TreeViewComponent).element.children[0].children);
        const treeViewItem: Element[] = dropContainer.filter(el =>
el.classList.contains('e-list-item'));
        let dropIndex: number =
Array.prototype.indexOf.call(treeViewItem, dropNode);
        let newNode: { [key: string]: Object }[] = [{ id: 'list' +
this.i, text: (dragItem.header as HeaderModel).text as any }];
        (this.tabObj as TabComponent).removeTab(dragTabIndex);
        this.treeObj?.addNodes(newNode, 'Treeview', dropIndex);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Drag and drop items from external source

It is possible to drag and drop the items from any of the external sources into the Tab, by manually saving those dropped items as new item data through the `addTab` method of Tab and removing the dragged node through the `removeNodes` method of Treeview.

In this example, we have used the tree view control as an external source, and the child nodes from the tree view component are dragged and dropped onto the Tab. Therefore, it is necessary to use the `nodeDragStop` event of the Treeview component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged node through the `removeNodes` method of Treeview.

APP.COMPONENT.HTML

```

<div id='tabpanel'>
  <ejs-tab #tabObj id="draggableTab" heightAdjustMode='Auto'
dragArea='#tabpanel'>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]' [content]="content0"></e-
tabitem>
      <e-tabitem [header]='headerText[1]' [content]="content1"></e-
tabitem>
      <e-tabitem [header]='headerText[2]' [content]="content2"></e-
tabitem>
      <e-tabitem [header]='headerText[3]' [content]="content3"></e-
tabitem>
    </e-tabitems>
  </ejs-tab>
  <ejs-treeview #treeObj id='draggableTreeview' [fields]='field'
cssClass='treeview-external-drop-tab' dragArea='#container'
[allowDragAndDrop]='true' (nodeDragStop)='onNodeDragStop($event)'

```

```
(nodeDragging)='onNodeDrag($event) '>
</ejs-treeview>
</div>
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TabAllModule, TreeViewModel } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation, ViewChild } from '@angular/core';
import { TabComponent, TabItemModel, TreeViewComponent, DragAndDropEventArgs } from '@syncfusion/ej2-angular-navigations';
import { isNullOrUndefined } from '@syncfusion/ej2-base';
/**
 * Tab Component
 */
@Component({
  imports: [
    FormsModule,
    TabAllModule,

    TreeViewModel
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  @ViewChild('tabObj') tabObj?: TabComponent;
  @ViewChild('treeObj') treeObj?: TreeViewComponent;
  public headerText: Object[] = [{ 'text': 'India' }, { 'text': 'Australia' }, { 'text': 'USA' }, { 'text': 'France' }];
  public content0: string = 'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.';
  public content1: string = 'Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.';
  public content2: string = 'The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of Hawaii is in the
```

```

mid-North Pacific. The country also has five populated and nine unpopulated
territories in the Pacific and the Caribbean.';
    public content3: string = 'France, officially the French Republic is a
sovereign state comprising territory in western Europe and several overseas
regions and territories. The European part of France, called Metropolitan
France, extends from the Mediterranean Sea to the English Channel and the
North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679
square kilo metres and as of August 2015 has a population of 67 million,
counting all the overseas departments and territories.';
    public field: Object = {
        dataSource: [
            { text: 'Hennessey Venom', id: 'list-01' },
            { text: 'Bugatti Chiron', id: 'list-02' },
            { text: 'Bugatti Veyron Super Sport', id: 'list-03' },
            { text: 'SSC Ultimate Aero', id: 'list-04' },
            { text: 'Koenigsegg CCR', id: 'list-05' },
            { text: 'McLaren F1', id: 'list-06' },
            { text: 'Aston Martin One- 77', id: 'list-07' },
            { text: 'Jaguar XJ220', id: 'list-08' },
            { text: 'McLaren P1', id: 'list-09' },
            { text: 'Ferrari LaFerrari', id: 'list-10' },
        ],
        id: "id", text: "text"
    }
    onNodeDragStop(args: DragAndDropEventArgs): void {
        let dropElement: HTMLElement =
<HTMLElement>args.target.closest('#draggableTab .e-toolbar-item');
        if (dropElement != null) {
            const childrenArray = Array.from((this.tabObj as
TabComponent).element.children[0].children[0].children);
            const toolbarItem: Element[] = childrenArray.filter(el =>
el.classList.contains('e-toolbar-item'));
            let dropItemIndex: number = toolbarItem.indexOf(dropElement as
never);

            let newTabItem: TabItemModel[] = [{
                header: { 'text': args.draggedNodeData['text'].toString() },
                content: args.draggedNodeData['text'].toString() + ' Content'
            }];
            (this.tabObj as TabComponent).addTab(newTabItem, dropItemIndex);
            this.treeObj?.removeNodes([args.draggedNode]);
            args.cancel = true;
        } else {
            let dropNode: HTMLElement =
<HTMLElement>args.target.closest('#draggableTreeview .e-list-item ');
            if (!isNullOrUndefined(dropNode) && args.dropIndicator === 'e-
drop-in') {
                args.cancel = true;
            }
        }
    }
    onNodeDrag(args: DragAndDropEventArgs): void {
        if (!isNullOrUndefined((args.target as HTMLElement).closest('.tab-
content') as Element)) {
            args.dropIndicator = 'e-no-drop';
        } else if (!isNullOrUndefined(args.target.closest('#draggableTab .e-
tab-header') as Element)) {
            args.dropIndicator = 'e-drop-in';
        }
    }

```

```

    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Accessibility in Angular Tab component

The Tab 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 Tab 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>
```

ARIA attributes

Tab component is designed by considering [WAI-ARIA](#) standard. Tab is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices. The following list of attributes are added in the Tab.

| Roles and Attributes | Functionalities |

```
| --- | --- |
```

```
| tablist | Attribute is set to the Tab header element that describes actual role of the element. |
```

```
| tab | Attribute is set to the Tab items element to indicates an interactive element inside a tablist that, when activated, displays its associated tabpanel. |
```

```
| tabpanel | Attribute is set to the Tab content that describes the role for viewing the active content. |
```

```
| aria-orientation | Attribute is set to the Tab header element indicates the Tab header orientation. Default value of this attribute is horizontal. |
```

```
| aria-selected | Attribute set to the Tab items to indicates the selection state for Tab items. Active Tab is set to true for this attribute. |
```

```
| aria-labelledby | Attribute is set to the Tab content element to indicates the associated Tab header for the content. |
```

```
| aria-controls | Attribute is set to the Tab items element to indicates the associated tabpanel for the header. |
```

```
| aria-haspopup | Attribute is set to the Popup element to indicates the popup mode in the Tab. The default value of this attribute is false. If popup mode is enabled, the attribute value is set to true. |
```

```
| aria-disabled | Attribute set to the Tab items to It indicates the disabled state of the Tab. |
```

Keyboard interaction

By default, keyboard navigation is enabled. This component implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Tab element, you can use the following key combination for interacting with the Tab.

```
| Key | Description |
|-----|-----|
```

```
| Left | Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab. |
```

```
| Right | Moves focus to the next Tab. If focus is on the last Tab element, the focus will not move to any Tab. |
```


Enter or Space	Selects the Tab if it is not selected. Opens the popup dropdown icon if it is focussed. Select the Tab item as active when popup item is focussed.	
Esc(Escape)	Closes the popup if popup is in opened state.	
Down or Up	When the popup is open and focused, it will move to previous/next Tab items of the popup in the vertical direction.	
Home	Moves focus to the first Tab.	
End	Moves focus to the last Tab.	
Shift + F10	If popup mode is enabled, it opens the popup when the Tab is focused.	
Delete	Deletes the Tab, if close button is enabled in Tab header.	
Tab	To Move focus through the interactive elements.	
Shift + Tab	To Move focus through the interactive elements.	

Ensuring accessibility

The Tab component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tab component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tab component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style in Angular Tab component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing Tab

Use the following CSS to customize the Tab.

```
`CSS
.e-tab {
border: 5px solid rgb(173, 255, 47);
}
```

Customizing the Tab items

Use the following CSS to customize the header items of Tab.

```
`CSS
.e-tab .e-tab-header .e-toolbar-items {
background: #9faed8;
border: 2px solid blue;
```

```
}  
,
```

Use the following CSS to customize the content items of Tab.

```
`CSS
```

```
.e-tab .e-content .e-item {  
  color: #a78515;  
  font-size: 14px;  
}  
,
```

[Customizing Tab's header](#)

Use the following CSS to customize the header of Tab control.

```
`CSS
```

```
.e-tab .e-tab-header {  
  background: #badfba !important;  
}  
,
```

[Customizing Tab's header icon](#)

Use the following CSS to customize the header item icon of Tab control.

```
`CSS
```

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {  
  color: #badfba !important;  
}  
,
```

[Customizing Tab's content](#)

Use the following CSS to customize the content of Tab control.

```
`CSS
```

```
.e-tab .e-content {  
  background: #d1f6d1 !important;  
}  
,
```

[Customizing the hover state of Tab control](#)

Use the following CSS to customize the tab item when hovering.

```
`CSS
```

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-wrap:hover {
```

```
background: #d1f6d1 !important;
}
`
```

Use the following CSS to customize the tab item popup icon when hovering.

```
`CSS
.e-tab .e-tab-header .e-hor-nav .e-popup-up-icon:hover,
.e-tab .e-tab-header .e-hor-nav .e-popup-down-icon:hover {
background: #d1f6d1 !important;
}
`
```

Customizing selected item of Tab control

Use the following CSS to customize the selected tab item.

```
`CSS
.e-tab .e-tab-header .e-toolbar-item.e-active {
background: #d1f4d1;
}
`
```

Use the following CSS to customize the selected tab item text and icon.

```
`CSS
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-text,
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-icon {
color: green !important;
}
`
```

How To

Load content through post in Angular Tab component

The Tab supports to load external contents through **AJAX** library. Refer to the following steps.

- Import the **Ajax** module from **ej2-base** and initialize with URL path.
- Get the data from Ajax **Success** event, then initialize the Tab with retrieved external path data.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild, OnInit } from '@angular/core';
```

```

import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { enableRipple } from '@syncfusion/ej2-base';
import { Ajax } from '@syncfusion/ej2-base';
/**
 * Load content through Ajax
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="tab_html_markup" #tab_html_markup>
    <e-tabitems>
      <e-tabitem [header]='headerText[0] '>
        <ng-template #content>
          Twitter is an online social networking service
that enables users to send and read short 140-character messages called
"tweets".

          Registered users can read and post tweets, but
those who are unregistered can only read them. Users access Twitter
          through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
          offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
          and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting
          340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per day.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1] '>
        <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
          Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s membership to Harvard students,
          but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford University. It gradually added support
          for students at various other universities and
later to high-school students.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[2] ' content="#content">
        </e-tabitem>
    </e-tabitems>
  </ejs-tab>`
})
export class AppComponent {
  @ViewChild('tab_html_markup') tabObj?: TabComponent;
  public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
}, { 'text': 'WhatsApp' }];
  ngOnInit() {
    let ajax: Ajax = new Ajax('./ajax.html', 'GET', true);
    ajax.send().then();
  }
}

```

```

    ajax.onSuccess = (data: string): void => {
      (this.tabObj as TabComponent).items[2].content = data;
      (this.tabObj as TabComponent).refresh();
    };
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Prevent content swipe selection in Angular Tab component

We can prevent the tab selection on touch swipe action by using the Tab [selecting](#) event.

Refer the below sample with preventing swipe selection.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { SelectEventArgs } from '@syncfusion/ej2-navigations';
/**
 * Prevent content swipe selection
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" (selecting)="select($event)">
    <e-tabitems>
      <e-tabitem [header]='headerText[0]'>
        <ng-template #content>
          Twitter is an online social networking service
          that enables users to send and read short 140-character messages called
          "tweets".
          Registered users can read and post tweets, but
          those who are unregistered can only read them. Users access Twitter
          through the website interface, SMS or mobile
          device app Twitter Inc. is based in San Francisco and has more than 25
          offices around the world. Twitter was created in
          March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
          and launched in July 2006. The service rapidly
          gained worldwide popularity, with more than 100 million users posting
          340 million tweets a day in 2012. The service also
          handled 1.6 billion search queries per day.
        </ng-template>
      </e-tabitem>
    </e-tabitems>
  `
})

```

```

        <e-tabitem [header]='headerText[1] '>
            <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
                Dustin Moskovitz and Chris Hughes. The founders
had initially limited the website\'\'s membership to Harvard students,
                but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford University. It gradually added support
                for students at various other universities and
later to high-school students.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[2] '>
            <ng-template #content>
                WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
                business model. It uses the Internet to send text
messages, images, video, user location and audio media messages to
                other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user base of up to one
billion, [10]
                making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
                by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
            </ng-template>
        </e-tabitem>
    </e-tabitems>
</ejs-tab>`
))
export class AppComponent {
    @ViewChild('element') tabInstance?: TabComponent;
    public select (e: SelectEventArgs) {
        if (e.isSwiped) {
            e.cancel = true;
        }
    }
    public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
}, { 'text': 'WhatsApp' }];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize selected tab styles in Angular Tab component

You can customize the Tab style by overriding its header and active tab CSS classes. Define HTML string for adding animation and customizing the Tab header and pass it to [text](#) property. Now you can override the style using custom CSS classes added to the Tab elements.

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

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';

/**
 * Customize selected tab styles
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" [items]='items'></ejs-tab>`
})
export class AppComponent {
  public items: Object[] = [
    {
      header: { 'text': '<div id="template-wrap"><div class="e-image e-andrew"></div><div class="e-title fade-in">Andrew</div></div>' },
      content: 'Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.He is fluent in French and Italian and reads German.He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993.Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.'
    },
    {
      header: { 'text': '<div id="template-wrap"><div class="e-image e-margaret"></div><div class="e-title fade-in">Margaret</div></div>' },
      content: 'Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966).She was assigned to the London office temporarily from July through November 1992.'
    },
    {
      header: { 'text': '<div id="template-wrap"><div class="e-image e-janet"></div><div class="e-title fade-in">Janet</div></div>' },
      content: 'Janet has a BS degree in chemistry from Boston College (1984).She has also completed a certificate program in food retailing management.Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992.'
    }
  ]
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```

```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize tab scroll step in Angular Tab component

Tab supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize **ScrollStep** property for scrolling distance. Refer to the following code example.

- By using Tab scrollStep property, pass a required value to customize tab scrollStep.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabComponent, TabItemsDirective, TabItemDirective } from
'@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" scrollStep="50">
    <e-tabitems>
      <e-tabitem [header]='headerText[0] '>
        <ng-template #content>
          HyperText Markup Language, commonly referred
to as HTML, is the standard markup language used to create web pages. Along
with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging web
pages, user interfaces for web applications,
and user interfaces for many mobile applications. Web browsers can read
HTML files and render them into visible or
audible web pages. HTML describes the structure of a website semantically
along with cues for presentation, making it a
markup language, rather than a programming language.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1] '>
        <ng-template #content>
          C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led
          by Anders Hejlsberg. The most recent version
is C# 5.0, which was released on August 15, 2012.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[2] '>
        <ng-template #content>
          Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle
Corporation,
```


that provides a system **for** developing application software and deploying it **in** a cross-platform computing environment.

Java **is** used **in** a wide variety of computing platforms **from** embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run **in** secure, sandboxed environments to provide many features of native applications and can be embedded **in** HTML pages.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[3] '>
  <ng-template #content>
```

The command-line compiler, VBC.EXE, **is** installed **as** part of the freeware .NET Framework SDK. Mono also includes a command-line

VB.NET compiler. The most recent version **is** VB **2012**, which was released on August **15, 2012**.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[4] '>
  <ng-template #content>
```

Xamarin **is** a San Francisco, California based software company created **in** May **2011**[3] by the engineers that created Mono, [4]

Mono **for** Android and MonoTouch that are cross-platform implementations of the Common Language Infrastructure (CLI) and Common Language **Specifications** (often called Microsoft .NET). With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms.[5]

Xamarin has over **1** million developers **in** more than **120** countries around the World **as** of May **2015**.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[5] '>
  <ng-template #content>
```

ASP.NET **is** an open-source server-side web application framework designed **for** web development to produce **dynamic** web pages.

It was developed by Microsoft to allow programmers to build **dynamic** web sites, web applications and web services.

It was first released **in** January **2002** with version **1.0** of the .NET Framework, and **is** the successor to Microsoft\'\'s Active Server **Pages** (ASP) technology. ASP.NET **is** built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code **using any** supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[6] '>
  <ng-template #content>
```

The ASP.NET MVC **is** a web application framework developed by Microsoft, which implements the model-view-controller (MVC) pattern.

It **is** open-source software, apart **from** the ASP.NET Web Forms component which **is** proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform **using only** Razor pages) will merge **into** a unified MVC **6**. The project **is** called ASP.NET vNext.

```

        </ng-template>
    </e-tabitem>
    <e-tabitem [header]='headerText[7] '>
        <ng-template #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.[5]
More recently, however, it has become common in both game development
            and the creation of desktop applications.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>`
    })
    export class AppComponent {
        public headerText: Object = [{ text: 'HTML' }, { text: 'C Sharp(C#)' }, {
text: 'Java' }, { text: 'VB.Net' }, { text: 'Xamarin' }, { text: 'ASP.NET' },
{ text: 'ASP.NET MVC' }, { text: 'JavaScript' }];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Create wizard using tab in Angular Tab component

Tab items can be disabled dynamically by passing the index and boolean value with the [enableTab](#) method and also passing index or HTML element to select an item from the tab using [select](#) method.

Create the following contents for each tab in the wizard.

1. Search tab:

Created with [DropDownList](#) to select the source, destination and type of ticket. A [DatePicker](#) for choosing the date of journey.

2. Train tab:

Based on the selected start and end point, populated Grid with random list of available seats and train list. Initially define the columns and row selected event for validating, after the source and destination chosen update the [dataSource](#) for the Grid.

3. Passenger tab:

A table with Textbox, Numeric, DropDownList for adding passenger name, age, gender and preferred berth/seat. Add validation on entering passenger details to proceed.

4. Payment tab:

Calculate the ticket cost based on location, passenger count and ticket type. Generate data for Grid with passenger details, train number and ticket cost summary.

You can go back on each tab using buttons available in it and tabs are [disabled](#) to navigate through tab header click actions. Once you end the wizard all the data get cleared and wizard goes back to starting tab.

In the below demo, designed for simple train reservation module that enable/disable tab items based on sequential validation of each Tab content.

APP.COMPONENT.HTML

```
<ejs-tab #tab id="tab_wizard" heightAdjustMode="None" height=390
showCloseButton=false (selected)="tabSelected($event)" ">
  <e-tabitems>
    <e-tabitem [header]='headerText[0] '>
      <ng-template #content>
        <div id='booking'>
          <div class="wizard-title">Plan your journey</div>
          <div class="responsive-align">
            <div class='row'>
              <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6
search-item">
                <ejs-dropdownlist #startPoint width="100%"
[dataSource]='cities'
                [fields]='autoCompleteFields'
floatLabelType='Auto' placeholder='From'>
              </ejs-dropdownlist>
            </div>
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6
search-item">
              <ejs-dropdownlist #endPoint width="100%"
[dataSource]='cities'
              [fields]='autoCompleteFields'
floatLabelType='Auto' placeholder='To'>
            </ejs-dropdownlist>
          </div>
        </div>
        <div class="row">
          <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6
search-item">
            <ejs-datepicker #date width='100%'
placeholder='Journey Date' floatLabelType='Auto'
            [min]='dateMin' [max]='dateMax'
[.value]='date'></ejs-datepicker>
          </div>
          <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6
search-item">
```

```

                                <ejs-dropdownlist #quota
[dataSource]='quotas' placeholder='Ticket type'
                                floatLabelType='Auto'
[fields]='fields'></ejs-dropdownlist>
                                </div>
                                </div>
                                <div class="btn-container">
                                    <button id='searchNext' class='e-btn'
(click)='btnClicked($event)'>Search Train</button>
                                </div>
                                    <span id="err1">{{err1}}</span>
                                </div>
                                </div>
                                </ng-template>
                            </e-tabitem>
                            <e-tabitem [header]='headerText[1]' disabled=true>
                                <ng-template #content>
                                    <div id='selectTrain'>
                                        <div class="wizard-title">Select the train from the list
</div>
                                        <ejs-grid #availableTrain width="100%"
(created)="availableTrainGridcreated()"
(rowSelected)='trainSelected($event)'>
                                            <e-columns>
                                                <e-column field='TrainNo' headerText='Train No'
width=120 type='number'></e-column>
                                                <e-column field='Name' headerText='Name'
width=140></e-column>
                                                <e-column field='Departure'
headerText='Departure' width=120></e-column>
                                                <e-column field='Arrival' headerText='Arrival'
width=140></e-column>
                                                <e-column field='Availability'
headerText='Availability' width=140 type='number'></e-column>
                                            </e-columns>
                                        </ejs-grid>
                                        <br />
                                        <div class="btn-container">
                                            <button id="goToSearch" class='e-btn'
(click)='btnClicked($event)'>Back</button>
                                            <button id="bookTickets" class='e-btn'
(click)='btnClicked($event)'>Continue</button>
                                        </div>
                                            <span id="err2">{{err2}}</span>
                                        </div>
                                    </ng-template>
                                </e-tabitem>
                                <e-tabitem [header]='headerText[2]' disabled=true>
                                    <ng-template #content>
                                        <div id='details'>
                                            <div class="details-page wizard-title">Enter the
passenger details</div>
                                            <div id="PassengersList">
                                                <table id="passenger-table">
                                                    <colgroup>
                                                        <col>
                                                        <col>
                                                    </colgroup>

```

```

        <col>
        <col>
        <col>
        <col>
    </colgroup>
    <thead>
        <tr>
            <th class="name-header">Name</th>
            <th class="age-header">Age</th>
            <th class="gender-header">Gender</th>
            <th class="type-header">Berth
Preference</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>
                <input #pass_name1 class="e-input"
type="text" placeholder="Passenger Name">
            </td>
            <td>
                <ejs-numerictextbox #pass_age1
showSpinButton=false min=1 max=100 value=18
                format=n0></ejs-numerictextbox>
            </td>
            <td>
                <ejs-dropdownlist #pass_gender1
[dataSource]='gender' text="Male"
                [fields]='fields'></ejs-
dropdownlist>
            </td>
            <td>
                <ejs-dropdownlist #pass_berth1
[dataSource]='berths' placeholder="Optional"
                [fields]='fields'></ejs-
dropdownlist>
            </td>
        </tr>
        <tr>
            <td>
                <input #pass_name2 class="e-input"
type="text" placeholder="Passenger Name">
            </td>
            <td>
                <ejs-numerictextbox
showSpinButton=false min=1 max=100 value=18 format=n0>
                </ejs-numerictextbox>
            </td>
            <td>
                <ejs-dropdownlist #pass_gender2
[dataSource]='gender' text="Male"
                [fields]='fields'></ejs-
dropdownlist>
            </td>
            <td>
                <ejs-dropdownlist #pass_berth2
[dataSource]='berths' placeholder="Optional"

```

```

[fields]='fields'></ejs-
dropdownlist>
        </td>
    </tr>
    <tr>
        <td>
            <input #pass_name3 class="e-input"
type="text" placeholder="Passenger Name">
        </td>
        <td>
            <ejs-numerictextbox
showSpinButton=false min=1 max=100 value=18 format=n0>
            </ejs-numerictextbox>
        </td>
        <td>
            <ejs-dropdownlist #pass_gender3
[dataSource]='gender' text="Male"
            [fields]='fields'></ejs-
dropdownlist>
        </td>
        <td>
            <ejs-dropdownlist #pass_berth3
[dataSource]='berths' placeholder="Optional"
            [fields]='fields'></ejs-
dropdownlist>
        </td>
    </tr>
</tbody>
</table>
</div>
<br />
<div class="btn-container">
    <button id="goBackToBook" class='e-btn'
(click)='btnClicked($event)'>Back</button>
    <button id="confirmTickets" class='e-btn'
(click)='btnClicked($event)'>Continue</button>
</div>
<span id="err3">{{err3}}</span>
</div>
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[3]' disabled=true>
    <ng-template #content>
        <div id='confirm'>
            <div class="tab-title1 wizard-title">Confirm the details
and proceed</div>
            <ejs-grid #ticketDetailGrid width="100%"
(created)="ticketDetailGridcreated()">
                <e-columns>
                    <e-column field='TrainNo' headerText='Train No'
width=120 type='number'></e-column>
                    <e-column field='PassName' headerText='Name'
width=140></e-column>
                    <e-column field='Gender' headerText='Gender'
width=120></e-column>
                    <e-column field='Berth' headerText='Berth'
width=140></e-column>

```

```

        </e-columns>
    </ejs-grid>
    <br />
    <div id="amount" #amount>{{displayAmt}}</div>
    <br />
    <div class="btn-container">
        <button id="goBackDetails" class='e-btn '
(click)='btnClicked($event)'>Back</button>
        <button id="makePayment" class='e-btn '
(click)='btnClicked($event)'>Pay</button>
    </div>
    </div>
</ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>
<div>
    <ejs-dialog #alertDialog header='Success' width=250 isModal=true
showCloseIcon=true
        content='Your payment successfully processed' [target]='dlgTarget'
[buttons]='dlgButtons'
        (created)='dlgCreated()' [visible]=false></ejs-dialog>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { GridAllModule } from '@syncfusion/ej2-angular-grids'
import { DialogAllModule } from '@syncfusion/ej2-angular-popups'
import { TabAllModule } from '@syncfusion/ej2-angular-navigations'
import { DatePickerAllModule } from '@syncfusion/ej2-angular-calendars'
import { NumericTextBoxAllModule } from '@syncfusion/ej2-angular-inputs'
import { DropDownListAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild, OnInit, ViewEncapsulation, ElementRef } from
 '@angular/core';
import { DialogComponent } from '@syncfusion/ej2-angular-popups';
import { TabComponent, SelectEventArgs } from '@syncfusion/ej2-angular-
navigations';
import { DatePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { NumericTextBoxComponent } from '@syncfusion/ej2-angular-inputs';
import { GridComponent, RowSelectEventArgs } from '@syncfusion/ej2-angular-
grids';
import { isNullOrUndefined as isNOU } from '@syncfusion/ej2-base';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
/**
 * Create wizard using Tab
 */
// tslint:disable:max-line-length
@Component({
imports: [
    FormsModule,
    TabAllModule,

    GridAllModule,

```

```

        DialogAllModule,
        DatePickerAllModule,
        DropDownListAllModule,
        NumericTextBoxAllModule
    ],
    standalone: true,
    selector: 'app-container',
    templateUrl: './app.component.html',
    encapsulation: ViewEncapsulation.None
  })
  export class AppComponent implements OnInit {
    @ViewChild('tab') tab?: TabComponent;
    @ViewChild('pass_name1') input1?: ElementRef;
    @ViewChild('pass_name2') input2?: ElementRef;
    @ViewChild('pass_name3') input3?: ElementRef;
    @ViewChild('alertDialog') alertDlg?: DialogComponent;
    @ViewChild('date') journeyDate?: DatePickerComponent;
    @ViewChild('quota') ticketType?: DropDownListComponent;
    @ViewChild('endPoint') endPoint?: DropDownListComponent;
    @ViewChild('startPoint') startPoint?: DropDownListComponent;
    @ViewChild('availableTrain') availTrainGrid?: GridComponent;
    @ViewChild('pass_age1') passagel?: NumericTextBoxComponent;
    @ViewChild('ticketDetailGrid') ticketDetailGrid?: GridComponent;
    @ViewChild('pass_gender1') passgender1?: DropDownListComponent;
    @ViewChild('pass_gender2') passgender2?: DropDownListComponent;
    @ViewChild('pass_gender3') passgender3?: DropDownListComponent;
    @ViewChild('pass_berth1') passBerth1?: DropDownListComponent;
    @ViewChild('pass_berth2') passBerth2?: DropDownListComponent;
    @ViewChild('pass_berth3') passBerth3?: DropDownListComponent;
    public fields: Object = {};
    public quotas: Object[] = [];
    public gender: Object[] = [];
    public berths: Object[] = [];
    public today: Date = new Date();
    public cities?: any;
    public locations?: any;
    public headerText: Object[] = [];
    public dlgButtons: Object[] = [];
    public selectedTrain?: any;
    public autoCompleteFields: Object = {};
    public dlgTarget: HTMLElement = document.querySelector('#container') as
    HTMLElement;
    public dateMin?: Date;
    public dateMax?: Date;
    public result: Object[] = [];
    public reserved: Object[] = [];
    public err1: string = '';
    public err2: string = '';
    public err3: string = '';
    public displayAmt: string = '';
    public ngOnInit(): void {
      document.body.style.visibility = 'hidden';
      this.dateMin = new Date(this.today.getTime());
      this.dateMax = new Date(this.today.getTime() + 60 * 24 * 60 * 60 *
1000);
      this.headerText = [
        { 'text': 'New Booking' },

```



```

        { 'text': 'Train List' },
        { 'text': 'Add Passenger' },
        { 'text': 'Make Payment' }]];
    this.quotas = [
        { id: '1', text: 'Business Class' },
        { id: '2', text: 'Economy Class' },
        { id: '4', text: 'Common Class' }
    ];
    this.gender = [
        { id: '1', text: 'Male' },
        { id: '2', text: 'Female' }
    ];
    this.berths = [
        { id: '1', text: 'Upper' },
        { id: '2', text: 'Lower' },
        { id: '3', text: 'Middle' },
        { id: '4', text: 'Window' },
        { id: '5', text: 'Aisle' }
    ];
    this.cities = [
        { name: 'Chicago', fare: 300 },
        { name: 'San Francisco', fare: 125 },
        { name: 'Los Angeles', fare: 175 },
        { name: 'Seattle', fare: 250 },
        { name: 'Florida', fare: 150 }
    ];
    this.fields = { id: 'id', text: 'text', value: 'text' };
    this.autoCompleteFields = { text: 'name', value: 'name' };
    this.dlgButtons = [{
        buttonModel: { content: 'Ok', isPrimary: true },
        click: () => {
            (this.alertDlg as DialogComponent).hide();
            (this.tab as TabComponent).enableTab(0, true);
            (this.tab as TabComponent).enableTab(1, false);
            (this.tab as TabComponent).enableTab(2, false);
            (this.tab as TabComponent).enableTab(3, false);
            (this.tab as TabComponent).select(0);
        }
    }]];
}

public ngAfterViewInit(): void {
    document.body.style.visibility = 'visible';
}

public showDate(): void {
    (this.journeyDate as DatePickerComponent).show();
}

public trainSelected(args: RowSelectEventArgs): void {
    this.selectedTrain = args.data;
}

public tabSelected(e: SelectEventArgs): void {
    if (e.isSwiped) {
        e.cancel = true;
    }
}

public dlgCreated(): void {
    (this.alertDlg as DialogComponent).hide();
}

```

```

public btnClicked(e: any): void {
    switch (e.target.id) {
        case 'searchNext':
            /* Validate the Source, Destination, Date and Class chosen
            and proceed only if all the fields are selected */
            if (!isNOU((this.startPoint as DropDownListComponent).value)
            && !isNOU((this.endPoint as DropDownListComponent).value) &&
                !isNOU((this.ticketType as DropDownListComponent).value)
            && !isNOU((this.journeyDate as DatePickerComponent).value)) {
                if (!isNOU((this.startPoint as
                DropDownListComponent).value) && (this.startPoint as
                DropDownListComponent).value == (this.endPoint as
                DropDownListComponent).value) {
                    this.err1 = '* Arrival point cannot be same as
Departure';
                } else {
                    (this.tab as TabComponent).enableTab(0, false);
                    (this.tab as TabComponent).enableTab(1, true);
                    this.filterTrains(e);
                    (this.tab as TabComponent).select(1);
                    this.err1 = '';
                    this.err2 = '';
                }
            } else {
                this.err1 = '* Please fill all the details before
proceeding';
            }
            break;
        case 'bookTickets':
            /* Based on the selected station generate Grid content to
            display trains available */
            if ((this.availTrainGrid as
            GridComponent).getSelectedRecords() === undefined || (this.availTrainGrid as
            GridComponent).getSelectedRecords().length === 0) {
                this.err2 = '* Select your convenient train';
            } else {
                (this.tab as TabComponent).enableTab(2, true);
                (this.tab as TabComponent).select(2);
                (this.tab as TabComponent).enableTab(1, false);
                this.err2 = '';
            }
            break;
        case 'confirmTickets':
            /* Get the Passenger details and validate the fields must not
            be left empty */
            if ((this.input1 as any).nativeElement.value === '' ||
            isNOU((this.passgender1 as DropDownListComponent).value) ||
            isNOU((this.passage1 as NumericTextBoxComponent).value)) {
                this.err3 = '* Please enter passenger details';
            } else {
                (this.tab as TabComponent).enableTab(3, true);
                (this.tab as TabComponent).select(3);
                (this.tab as TabComponent).enableTab(2, false);
                this.err3 = '';
                this.finalizeDetails(e);
            }
            break;
    }
}

```

```

        case 'makePayment':
            this.alertDlg?.show();
            break;
        case 'goToSearch':
            /* Go back to change class, date or boarding places */
            this.selectedTrain = [];
            (this.tab as TabComponent).enableTab(0, true);
            (this.tab as TabComponent).select(0);
            (this.tab as TabComponent).enableTab(1, false);
            break;
        case 'goBackToBook':
            /* Change the preferred train chosen already */
            (this.tab as TabComponent).enableTab(1, true);
            (this.tab as TabComponent).select(1);
            (this.tab as TabComponent).enableTab(2, false);
            break;
        case 'goBackDetails':
            /* Update passenger detail before confirming the payment */
            (this.tab as TabComponent).enableTab(2, true);
            (this.tab as TabComponent).select(2);
            (this.tab as TabComponent).enableTab(3, false);
            break;
    }
}

public filterTrains(args: any): void {
    /* Generating trains based on source and destination chosen */
    let fromCity: string = <string>this.startPoint?.value;
    let toCity: string = <string>this.endPoint?.value;
    let count: number = Math.floor((Math.random() * 3) + 2);
    for (let i: number = 0; i < count; i++) {
        let details = <Details>{};
        details.TrainNo = Math.floor((Math.random() * 20) + 19000);
        details.Name = 'Train ' + i;
        details.Departure = fromCity;
        details.Arrival = toCity;
        details.Availability = Math.floor((Math.random() * 20) + 20);
        this.result.push(details);
    }
}

public availableTrainGridcreated(): void {
    (this.availTrainGrid as GridComponent).dataSource = this.result;
}

public finalizeDetails(args: any): void {
    /* Get the passenger details and update table with name and other details for confirmation */
    let passCount: any = 0;
    for (let i: number = 1; i <= 3; i++) {
        if ((this.input1 as any).nativeElement.value !== '') {
            let details = <Details>{};
            let gender: any = (i === 1) ? (this.passgender1 as DropDownListComponent).value : (i === 2) ?
                (this.passgender2 as DropDownListComponent).value :
                (this.passgender3 as DropDownListComponent).value;
            let berth: any = (i === 1) ? (this.passBerth1 as DropDownListComponent).value : (i === 2) ?
                (this.passBerth2 as DropDownListComponent).value :
                (this.passBerth3 as DropDownListComponent).value;

```

```

        details.TrainNo = this.selectedTrain.TrainNo.toString();
        details.PassName = (i === 1) ? (this.input1 as
any).nativeElement.value : (i === 2) ?
        (this.input2 as any).nativeElement.value : (this.input3
as any).nativeElement.value;
        details.Gender = (gender === '') ? 'Male' : gender;
        details.Berth = (berth === '') ? 'Any' : berth;
        if (details.PassName !== '') { this.reserved.push(details); }
        passCount++;
    }
    let calcFare: any = 0;
    for (let i: number = 0; i < this.cities; i++) {
        if ((this.startPoint as DropDownListComponent).value ===
this.cities[i].name) { calcFare = calcFare + this.cities[i].fare; }
        if ((this.endPoint as DropDownListComponent).value ===
this.cities[i].name) { calcFare = calcFare + this.cities[i].fare; }
    }
    if ((this.ticketType as DropDownListComponent).value === 'Economy
Class') {
        this.displayAmt = "Total payable amount: $" + passCount *
(300 + calcFare)
    } else if ((this.ticketType as DropDownListComponent).value ===
'Business Class') {
        this.displayAmt = "Total payable amount: $" + passCount *
(500 + calcFare)
    } else if ((this.ticketType as DropDownListComponent).value ===
'Common Class') {
        this.displayAmt = "Total payable amount: $" + passCount *
(150 + calcFare)
    }
    }
    public ticketDetailGridcreated(): void {
        (this.ticketDetailGrid as GridComponent).dataSource = this.reserved;
    }
}
interface Details {
    TrainNo: number;
    Name: string;
    Departure: string;
    Arrival: string;
    Availability: number;
    PassName: string;
    Gender: any;
    Berth: string;
    SeatNo: number;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Load tab with data source in Angular Tab component

You can bind any data object to Tab items, by mapping it to [header](#) and [Link to the Video](#) property.

You can watch the following video to learn more about loading tab items from a remote data source in the Angular Tabs component:

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 Tab.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, OnInit, ViewChild } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { DataManager, Query, ODataV4Adaptor, ReturnOption } from '@syncfusion/ej2-data';
const SERVICE_URI: string =
'https://services.odata.org/V4/Northwind/Northwind.svc/Employees';
/**
 * Load tab with DataSource
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab #tab></ejs-tab>`
})
export class AppComponent implements OnInit {
  @ViewChild('tab') tabObj?: TabComponent;
  public itemsData: any = [];
  public mapping = { header: 'FirstName', content: 'Notes' };
  public ngOnInit(): void {
    new DataManager({ url: SERVICE_URI, adaptor: new ODataV4Adaptor })
      .executeQuery(new Query().range(1, 4)).then((e: ReturnOption) => {
        let result: any = e.result;
        for(let i: number = 0; i < result.length; i++) {
          this.itemsData.push({ header: {text:
result[i][this.mapping.header]}, content: result[i][this.mapping.content] });
        }
        (this.tabObj as TabComponent).items = this.itemsData;
        (this.tabObj as TabComponent).dataBind();
      });
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add nested tabs in Angular Tab component

Tab supports to render the nested level of Tabs by using [content](#) property.

You can add the nested Tab element inside the parent Tab [content](#) property.

To render the nested Tab, initialize the component using the id of Tab from [selected](#) event handler.

APP.COMPONENT.HTML

```
<ejs-tab id="element">
  <e-tabitems>
    <e-tabitem>
      <ng-template #headerText>
        <div>
          USA
        </div>
      </ng-template>
      <ng-template #content>
        <ejs-tab>
          <e-tabitems>
            <e-tabitem>
              <ng-template #headerText>
                <div>
                  New York
                </div>
              </ng-template>
              <ng-template #content>
                New York City comprises 5 boroughs sitting
where the Hudson River meets the Atlantic
Ocean. At its core is Manhattan, a densely
populated borough that's among the world's
major commercial, financial and cultural
centers. Its iconic sites include skyscrapers
such as the Empire State Building and
sprawling Central Park. Broadway theater is staged
in neon-lit Times Square.
              </ng-template>
            </e-tabitem>
            <e-tabitem>
              <ng-template #headerText>
                <div>
                  Los Angeles
                </div>
              </ng-template>
              <ng-template #content>
                Los Angeles is a sprawling Southern
California city and the center of the nation's film
and television industry. Near its iconic
Hollywood sign, studios such as Paramount
Pictures, Universal and Warner Brothers offer
behind-the-scenes tours. On Hollywood
Boulevard, TCL Chinese Theatre displays
celebrities' hand- and footprints, the Walk of
Fame honors thousands of luminaries and
vendors sell maps to stars' homes.
              </ng-template>
            </e-tabitem>
          </e-tabitems>
        </ejs-tab>
      </ng-template>
    </e-tabitem>
  </e-tabitems>
</ejs-tab>
```

```

        </ng-template>
      </e-tabitem>
    <e-tabitem>
      <ng-template #headerText>
        <div>
          Chicago
        </div>
      </ng-template>
      <ng-template #content>
        Chicago, on Lake Michigan in Illinois, is
among the largest cities in the U.S. Famed for
its bold architecture, it has a skyline
punctuated by skyscrapers such as the iconic
John Hancock Center, 1,451-ft. Willis Tower
(formerly the Sears Tower) and the
neo-Gothic Tribune Tower. The city is also
renowned for its museums, including the Art
Institute of Chicago with its noted
Impressionist and Post-Impressionist works.
      </ng-template>
    </e-tabitem>
  </e-tabitems>
</ejs-tab>
</ng-template>
</e-tabitem>
<e-tabitem>
  <ng-template #headerText>
    <div>
      FRANCE
    </div>
  </ng-template>
  <ng-template #content>
    <ejs-tab>
      <e-tabitems>
        <e-tabitem>
          <ng-template #headerText>
            <div>
              Paris
            </div>
          </ng-template>
          <ng-template #content>
            Paris, France capital, is a major European
city and a global center for art, fashion,
gastronomy and culture. Its 19th-century
cityscape is crisscrossed by wide boulevards
and the River Seine. Beyond such landmarks as
the Eiffel Tower and the 12th-century,
Gothic Notre-Dame cathedral, the city is
known for its cafe culture and designer
boutiques along the Rue du Faubourg Saint-
Honoré.
          </ng-template>
        </e-tabitem>
      </e-tabitems>
    </e-tabitem>
  <ng-template #headerText>
    <div>
      Marseille

```

```

        </div>
    </ng-template>
    <ng-template #content>
        Marseille, a port city in southern France,
has been a crossroads of immigration and
        trade since its founding by the Greeks circa
600 B.C. At its heart is the Vieux-Port
        (Old Port), where fishmongers sell their
catch along the boat-lined quay. Basilique
        Notre-Dame-de-la-Garde is a Romanesque-
Byzantine church. Modern landmarks include Le
        Corbusier's influential Cité Radieuse complex
and Zaha Hadid's CMA CGM Tower.
    </ng-template>
</e-tabitem>
<e-tabitem>
    <ng-template #headerText>
        <div>
            Lyon
        </div>
    </ng-template>
    <ng-template #content>
        Lyon, the capital city in France's Auvergne-
Rhône-Alpes region, sits at the junction of
        the Rhône and Saône rivers. Its center
reflects 2,000 years of history from the Roman
        Amphithéâtre des Trois Gaules, medieval and
Renaissance architecture in Vieux (Old)
        Lyon, to the modern Confluence district on
Presquîle peninsula. Traboules, covered
        passageways between buildings, connect Vieux
Lyon and La Croix-Rousse hill.
    </ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>
</ng-template>
</e-tabitem>
<e-tabitem>
    <ng-template #headerText>
        <div>
            AUSTRALIA
        </div>
    </ng-template>
    <ng-template #content>
        <ejs-tab>
            <e-tabitems>
                <e-tabitem>
                    <ng-template #headerText>
                        <div>
                            Sydney
                        </div>
                    </ng-template>
                    <ng-template #content>
                        Sydney, capital of New South Wales and one of
Australia largest cities, is best known

```



```

        for its harbourfront Sydney Opera House, with
a distinctive sail-like design. Massive
        Darling Harbour and the smaller Circular Quay
port are hubs of waterside life, with the
        arched Harbour Bridge and esteemed Royal
Botanic Garden nearby. Sydney Tower's outdoor
        platform, the Skywalk, offers 360-degree
views of the city and suburbs.
        </ng-template>
    </e-tabitem>
    <e-tabitem>
        <ng-template #headerText>
            <div>
                Melbourne
            </div>
        </ng-template>
        <ng-template #content>
            Melbourne is the coastal capital of the
southeastern Australian state of Victoria. At
            the city centre is the modern Federation
Square development, with plazas, bars, and
            restaurants by the Yarra River. In the
Southbank area, the Melbourne Arts Precinct is
            the site of Arts Centre Melbourne - a
performing arts complex - and the National Gallery
            of Victoria, with Australian and indigenous
art.
        </ng-template>
    </e-tabitem>
    <e-tabitem>
        <ng-template #headerText>
            <div>
                Brisbane
            </div>
        </ng-template>
        <ng-template #content>
            Brisbane, capital of Queensland, is a large
city on the Brisbane River. Clustered in its
            South Bank cultural precinct are the
Queensland Museum and Sciencentre, with noted
            interactive exhibitions. Another South Bank
cultural institution is Queensland Gallery
            of Modern Art, among Australia major
contemporary art museums. Looming over the city is
            Mt. Coot-tha, site of Brisbane Botanic
Gardens.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>
</ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>

```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
/**
 * Add nested Tabs
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  templateUrl: './app.component.html',
})
export class AppComponent {
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set state persistence of the tab component in Angular Tab component

State persistence allows the Tab to retain the current modal value in the browser cookies for state maintenance.

This action is handled through the [enablePersistence](#) property which is set to false by default.

When it is set to true, some of the Tab component model values will be retained even after refreshing the page.

The following sample demonstrates how to set state persistence of the Tab component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabComponent, TabItemsDirective, TabItemDirective } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" enablePersistence=true>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]`>
```

```

        <ng-template #content>
            Twitter is an online social networking service
            that enables users to send and read short 140-character messages called
            "tweets".

            Registered users can read and post tweets, but
            those who are unregistered can only read them. Users access Twitter
            through the website interface, SMS or mobile
            device app Twitter Inc. is based in San Francisco and has more than 25
            offices around the world. Twitter was created in
            March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
            and launched in July 2006. The service rapidly
            gained worldwide popularity, with more than 100 million users posting
            340 million tweets a day in 2012. The service also
            handled 1.6 billion search queries per day.
        </ng-template>
    </e-tabitem>
    <e-tabitem [header]='headerText[1] '>
        <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
            Dustin Moskovitz and Chris Hughes. The founders
            had initially limited the website\'s membership to Harvard students,
            but later expanded it to colleges in the Boston
            area, the Ivy League, and Stanford University. It gradually added support
            for students at various other universities and
            later to high-school students.
        </ng-template>
    </e-tabitem>
    <e-tabitem [header]='headerText[2] '>
        <ng-template #content>
            WhatsApp Messenger is a proprietary cross-
            platform instant messaging client for smartphones that operates under a
            subscription
            business model. It uses the Internet to send text
            messages, images, video, user location and audio media messages to
            other users using standard cellular mobile
            numbers. As of February 2016, WhatsApp had a user base of up to one
            billion, [10]
            making it the most globally popular messaging
            application. WhatsApp Inc., based in Mountain View, California, was acquired
            by Facebook Inc. on February 19, 2014, for
            approximately US$19.3 billion.
        </ng-template>
    </e-tabitem>
</e-tabitems>
</ejs-tab>`
})
export class AppComponent {
    public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
    }, { 'text': 'WhatsApp' } ]};
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set custom animation in Angular Tab component

Tab supports custom animations for both previous and next actions from the provided animation option of **Animation** library.

The [animation](#) property also allows you to set [easing](#), [duration](#), and various other [effect](#).

Default animation is given as **SlideLeftIn** for [previous](#) tab animation and **SlideRightIn** for [next](#) tab animation.

You can also disable the animation by setting the animation effect as **None**. Also, please use the following CSS to disable indicator animation for animation effect as **None**.

`CSS

```
.e-tab .e-tab-header:not(.e-vertical) .e-indicator, .e-tab .e-tab-header.e-vertical .e-indicator {
  transition: none;
}
```

The sample demonstrates some types of animation that suits Tab. You can check all the animation effects [here](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TabComponent, TabItemsDirective, TabItemDirective } from
 '@syncfusion/ej2-angular-navigations'
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { TabActionSettingsModel, TabComponent } from '@syncfusion/ej2-
angular-navigations';
@Component({
  imports: [
    ],
  standalone: true,
  selector: 'app-container',
  template: `
    <div style='padding-top: 25px'>
    <div class='row'>
    <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
      <label> Previous Animation </label></div>
      <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <div class='custom_drop'><ejs-dropdownlist #previousAnimation
        (change)='previousAnimationChange()' index='0' [dataSource]='animationData'
        placeholder='Previous Animation'></ejs-dropdownlist></div>
      </div></div>
```

```

<div class='row'>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<label> Next Animation </label></div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
<div class='custom_drop'><ejs-dropdownlist #nextAnimation
(change)='nextAnimationChange()' index='1' [dataSource]='animationData'
placeholder='Next Animation'></ejs-dropdownlist></div>
</div></div>
<div style='padding-top: 25px'>
<ejs-tab id="element" #element>
    <e-tabitems>
        <e-tabitem [header]='headerText[0] '>
            <ng-template #content>
                Twitter is an online social networking service
that enables users to send and read short 140-character messages called
"tweets".
                Registered users can read and post tweets, but
those who are unregistered can only read them. Users access Twitter
                through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
                offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
                and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting
                340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per day.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[1] '>
            <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
                Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s membership to Harvard students,
                but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford University. It gradually added support
                for students at various other universities and
later to high-school students.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[2] '>
            <ng-template #content>
                WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
                business model. It uses the Internet to send text
messages, images, video, user location and audio media messages to
                other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user base of up to one
billion,[10]
                making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
                by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
            </ng-template>
    </e-tabitems>
</e-tab>
</div>

```

```

        </e-tabitem>
    </e-tabitems>
</ejs-tab>
</div></div>

}))
export class AppComponent {
    @ViewChild('element') tabInstance?: TabComponent;
    @ViewChild('previousAnimation') previousInstance?: DropDownListComponent;
    @ViewChild('nextAnimation') nextInstance?: DropDownListComponent;
    public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
}], { 'text': 'WhatsApp' }];
    public animationData: string[] = ['SlideLeftIn', 'SlideRightIn',
'FadeIn', 'FadeOut', 'FadeZoomIn', 'FadeZoomOut', 'ZoomIn', 'ZoomOut',
'None'];
    public previousAnimationChange(): void {
        ((this.tabInstance as TabComponent).animation.previous as
TabActionSettingsModel).effect = (this.previousInstance as
DropDownListComponent).value as any;
    }
    public nextAnimationChange(): void {
        ((this.tabInstance as TabComponent).animation.next as
TabActionSettingsModel).effect = (this.nextInstance as
DropDownListComponent).value as any;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Disable default tab animation effects in Angular Tab component

You can disable the default Tab animations with the help of the [animation](#) property of the Tab.

The sample demonstrates Tab without the default animation effects. You can check the Tab without any animation effects here.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TabComponent, TabItemsDirective, TabItemDirective } from
'@syncfusion/ej2-angular-navigations'
import { Component, ViewEncapsulation } from "@angular/core";
@Component({
imports: [

    ],
standalone: true,
    selector: 'app-container',
    template: `<ejs-tab id="element" heightAdjustMode='Auto'
[animation]="animation">
        <e-tabitems>

```

```

        <e-tabitem [header]='headerText[0] '>
            <ng-template #content>
                Twitter is an online social networking service
that enables users to send and read short 140-character messages called
"tweets".

                Registered users can read and post tweets, but
those who are unregistered can only read them. Users access Twitter
                through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
                offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
                and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting
                340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per day.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[1] '>
            <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
                Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s membership to Harvard students,
                but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford University. It gradually added support
                for students at various other universities and
later to high-school students.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[2] '>
            <ng-template #content>
                WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
                business model. It uses the Internet to send text
messages, images, video, user location and audio media messages to
                other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user base of up to one
billion, [10]
                making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain View, California, was acquired
                by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
            </ng-template>
        </e-tabitem>
    </e-tabitems>
</ejs-tab>`
    })
    export class AppComponent {
        public headerText: Object = [{ text: "Twitter"}, { text: "Facebook" }, {
text: "WhatsApp" }];

        // Disabling tab animation
        public animation: object = {
            previous: { effect: "", duration: 0, easing: "" },

```

```

    next: { effect: "", duration: 0, easing: "" }
  };
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Load tab items dynamically in Angular Tab component

Tabs can be added dynamically by passing array of items and index value to the [addTab](#) method.

`typescript

// New tab title and content inputs are fetched and stored in local variable

```
let title: string = document.getElementById('tab-title').value;
```

```
let content: string = document.getElementById('tab-content').value;
```

// Required tab item object formed by using textbox inputs

```
let item: Object = { header: { text: title }, content: createElement('pre', { innerHTML:
content.replace(/\n/g, '<br>\n') }).outerHTML };
```

// Item object and the index argument passed into the addTab method to add a new tab

```
this.tabInstance.addTab([item], this.totalItems-1);
```

,

In the following demo, you can add the tab content by clicking the +. This + icon is added on the tab header using [iconCss](#) property. Enter the new Tab heading and content details in the available text boxes, click 'Add Tab' button to pass the details as an array and here last index is calculated to append the new tab at the end.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Tooltip } from '@syncfusion/ej2-popups';
import { Component, ViewChild } from '@angular/core';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
import { SelectEventArgs, TabComponent } from '@syncfusion/ej2-angular-
navigations';
enableRipple(true);
/**
 * Add new tabs dynamically
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,

```



```

        selector: 'app-container',
        template: `<ejs-tab #element id="element" (created)="tabCreated()"
(selected)="tabSelected($event)">
            <e-tabitems>
                <e-tabitem [header]='headerText[0]'
content="#tab1_content"></e-tabitem>
                <e-tabitem [header]='headerText[1]' content="#form-
container"></e-tabitem>
            </e-tabitems>
        </ejs-tab>`
    })
    export class AppComponent {
        @ViewChild('element') tabInstance?: TabComponent;
        public totalItems?: number;
        public headerText: Object = [{ 'text': 'Tab1' }, { 'iconCss': 'e-add-
icon' }];
        public tabCreated(): void {
            let tooltip: Tooltip = new Tooltip({
                content: 'Add Tab'
            });
            tooltip.appendTo('.e-ileft.e-icon');
            (document.getElementById('btn-add') as HTMLElement).onclick = (e :
Event) => {
                let title: string = (document.getElementById('tab-title') as
any).value;
                let content: string = (document.getElementById('tab-content') as
any).value;
                let item: Object = { header: { text: title }, content:
createElement('pre', { innerHTML: content.replace(/\n/g, '<br>\n')
}).outerHTML };
                this.totalItems = document.querySelectorAll('#element .e-toolbar-
item').length;
                this.tabInstance?.addTab([item], this.totalItems-1);
            };
        }
        public tabSelected(args: SelectEventArgs): void {
            if (args.selectedIndex === document.querySelectorAll('#element .e-
toolbar-item').length -1) {
                (document.getElementById('tab-title') as any).value = '';
                (document.getElementById('tab-content') as any).value = '';
            }
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Create collapsible tabs in Angular Tab component

You can achieve collapse and expand functionality in Tab by adding/removing a custom CSS class in the click event handler for each tab.

- Define a CSS class to set the style property display as none. Here 'collapse' class is added to the content element for hiding it using [created](#) event.
- Bind the [selected](#) event for Tab to collapse the initially selected Tab item

and bind custom click handler for the Tab headers.

- In the event handler, add and remove 'collapse' class to hide and show the corresponding Tab content.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild } from '@angular/core';
import { SelectEventArgs, TabComponent } from '@syncfusion/ej2-angular-navigations';
import { enableRipple, isNullOrUndefined } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Collapsible Tab
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" class="e-background"
(created)="tabCreated()" (selected)="tabSelected($event)" #element>
    <e-tabitems>
      <e-tabitem [header]='headerText[0] '>
        <ng-template #content>
          Twitter is an online social networking service
that enables users to send and read short 140-character messages called
"tweets".
          Registered users can read and post tweets, but
those who are unregistered can only read them. Users access Twitter
through the website interface, SMS or mobile
device app Twitter Inc. is based in San Francisco and has more than 25
offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass
and launched in July 2006. The service rapidly
gained worldwide popularity, with more than 100 million users posting
340 million tweets a day in 2012. The service also
handled 1.6 billion search queries per day.
        </ng-template>
      </e-tabitem>
      <e-tabitem [header]='headerText[1] '>
        <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum,
```

```

        Dustin Moskovitz and Chris Hughes. The founders
        had initially limited the website's membership to Harvard students,
        but later expanded it to colleges in the Boston
        area, the Ivy League, and Stanford University. It gradually added support
        for students at various other universities and
        later to high-school students.
    </ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[2] '>
    <ng-template #content>
        WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones that operates under a
subscription
        business model. It uses the Internet to send text
        messages, images, video, user location and audio media messages to
        other users using standard cellular mobile
        numbers. As of February 2016, WhatsApp had a user base of up to one
        billion, [10]
        making it the most globally popular messaging
        application. WhatsApp Inc., based in Mountain View, California, was acquired
        by Facebook Inc. on February 19, 2014, for
        approximately US$19.3 billion.
    </ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>`
    })
    export class AppComponent {
        @ViewChild('element') tabInstance!: TabComponent;
        public trgIndex?: number;
        public actLine?: HTMLElement;
        public headerText: Object[] = [
            { text: 'Twitter' },
            { text: 'Facebook' },
            { text: 'WhatsApp' }
        ];
        public tabCreated(): void {
            // Custom click event binding for each tab item to make collapse/expand
            const childrenArray =
Array.from((this.tabInstance).element.children[0].children[0].children);
            const toolbarItem: Element[] = childrenArray.filter(el =>
el.classList.contains('e-toolbar-item'));
            (toolbarItem[0] as HTMLElement).onclick = (e: Event) => {
                this.updateCollapseClass(0);
            };
            (toolbarItem[1] as HTMLElement).onclick = (e: Event) => {
                this.updateCollapseClass(1);
            };
            (toolbarItem[2] as HTMLElement).onclick = (e: Event) => {
                this.updateCollapseClass(2);
            };
            // After tab created first tab content and active line are hidden by
            adding custom class to make it collapse state
            this.actLine = document.querySelector('.e-indicator') as HTMLElement;
            (this.tabInstance.element.children[1].children[0].classList.add('collapse'));
            this.actLine.classList.add('collapse');

```

```

    }
    public tabSelected(e: SelectEventArgs): void {
        // If next tab item selected custom class is removed from content and
        // active line element
        const childrenArray =
        Array.from(this.tabInstance.element.children[1].children);
        const cnttrgs = childrenArray.filter(el => el.classList.contains('e-
        item'));
        for (let i: number = 0; i < cnttrgs.length; i++) {
            cnttrgs[i].classList.remove('collapse');
        }
        if (this.actLine !== undefined) {
            this.actLine.classList.remove('collapse');
        }
        this.trgIndex = e.selectedIndex;
    }
    public updateCollapseClass(index: number): void {
        // Custom classes are added/removed from tab content and active line
        // element, when the same tab item again clicked
        const childrenArray =
        Array.from(this.tabInstance.element.children[1].children);
        const cntEle: HTMLElement = childrenArray.filter(el => el.id === `e-
        content-element_${index}`)[0] as HTMLElement;
        if (!isNullOrUndefined(cntEle) && cntEle.classList.contains('collapse'))
        {
            cntEle.classList.remove('collapse');
            (this.actLine as HTMLElement).classList.remove('collapse');
        } else if (!isNullOrUndefined(cntEle)) {
            cntEle.classList.add('collapse');
            (this.actLine as HTMLElement).classList.add('collapse');
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Render other components in tab using angular template in Angular Tab component

You can render other components inside Tab using Angular [Link to the Video](#). Through this, we can add content as other components directly with all their functionalities to our Tab. We need to use `ng-template` inside the each `e-tabitem` tag with `#content` attribute, which is mandatory to render content. And now use `ng-template` tag with select attribute of id or class name for mapping required content.

Check out this video to learn about integrating other UI components inside the Angular Tab component:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'

```

```

import { GridAllModule } from '@syncfusion/ej2-angular-grids'
import { DialogAllModule } from '@syncfusion/ej2-angular-popups'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { CalendarAllModule } from '@syncfusion/ej2-angular-calendars'
import { NumericTextBoxAllModule } from '@syncfusion/ej2-angular-inputs'
import { ComboBoxAllModule } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { enableRipple, createElement } from '@syncfusion/ej2-base';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
import { ComboBoxComponent } from '@syncfusion/ej2-angular-dropdowns';
enableRipple(true);
@Component({
  imports: [
    FormsModule,
    TabModule,

    GridAllModule,
    DialogAllModule,
    CalendarAllModule,
    ComboBoxAllModule,
    NumericTextBoxAllModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element">
    <e-tabitems>
      <e-tabitem>
        <ng-template #headerText>
          <div> Tab1 </div>
        </ng-template>
        <ng-template #content>
          <ejs-calendar></ejs-calendar>
        </ng-template>
      </e-tabitem>
      <e-tabitem>
        <ng-template #headerText>
          <div> Tab2 </div>
        </ng-template>
        <ng-template #content>
          <ejs-combobox id='comboelement' #samples [dataSource]='data'
[placeholder]='text'></ejs-combobox>
        </ng-template>
      </e-tabitem>
      <e-tabitem>
        <ng-template #headerText>
          <div> Tab3 </div>
        </ng-template>
        <ng-template #content>
          <p>text inside tab3</p>
        </ng-template>
      </e-tabitem>
    </e-tabitems>
  </ejs-tab>`
})
export class AppComponent {
  @ViewChild('element') tabInstance?: TabComponent;
  // defined the array of data

```

```

    public data: string[] = ['Badminton', 'Basketball', 'Cricket', 'Golf',
    'Hockey', 'Rugby'];
    // set placeholder text to ComboBox input element
    public text: string = 'Select a game';
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add reactive forms within a tab in Angular Tab component

As we know already, we can render other components inside the Tab using Angular **ng-template**. Likewise, we can also render the reactive forms module inside the Tab items using this **ng-template**.

For more details about Reactive Forms refer: <https://angular.io/guide/reactive-forms>.

For the reactive forms you should import a **ReactiveFormsModule** into app module as well as the **FormGroup**, **FormControl** should be imported to app component. The **FormGroup** is used to declare **formGroupName** for the form and the **FormControl** is used to declare **formControlName** for form controls. You can declare the **formControlName** to **AutoComplete** as usual. Then, you must create a value object to the **FormGroup** and each

value will be the default value of the form control.

Create the reactive form like above and then directly refer that in your **ng-template**

The following example demonstrates how to add the reactive forms within a Tab item using **ng-template**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { AutoCompleteModule } from '@syncfusion/ej2-angular-dropdowns'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, Inject } from '@angular/core';
import { FormBuilder, FormsModule, FormGroup, Validators } from
 '@angular/forms';
@Component({
  imports: [ FormsModule, ReactiveFormsModule, AutoCompleteModule,
  DropDownListModule, ButtonModule, TabModule ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-tab id="element" #tab>
    <e-tabitems>
      <e-tabitem [header]='headerText[0]'>
        <ng-template #content>
          <div id="reactive-form">
            <div class="control-section">

```

```

        <div class="col-lg-8 content-wrapper">
            <div id='content' class='box-form' style="margin:
0 auto; width:250px; padding:25px">
                <form [formGroup]="skillForm" novalidate
id="formId">
                    <div class="form-group">
                        <ejs-autocomplete
formControlName="skillname" name="skillname"
[dataSource]='skillset'
allowCustom=false [placeholder]='placeholder'
floatLabelType='Auto'>
                    </ejs-autocomplete>
                </div>
                <div class="form-group" style="padding-
top:10px;">
                    <div class="e-float-input">
                        <input type="text"
autocomplete="off" (focus)="onfocus($event)" "
(blur)="onblur($event)" "
formControlName="sname" name="sname"
maxlength="10" />
                        <span class="e-float-
line"></span>
                        <label class="e-float-text e-
label-bottom">Name:</label>
                    </div>
                </div>
                <div class="form-group" style="padding-
top:10px;">
                    <div class="e-float-input">
                        <input type="text"
autocomplete="off" (focus)="onfocus($event)" "
(blur)="onblur($event)" "
formControlName="smail" maxlength="15" />
                        <span class="e-float-
line"></span>
                        <label class="e-float-text e-
label-bottom">Email:</label>
                    </div>
                </div>
                <div class="form-group"
style="padding:10px;">
                    <div class="col-xs-6 col-sm-6 col-lg-
6 col-md-6">
                        <button ejs-button
[disabled]="!skillForm.valid">Done</button></div>
                        <div class="col-xs-6 col-sm-6 col-lg-
6 col-md-6">
                            <button ejs-button type="reset"
(click)="onreset($event)">Cancel</button>
                        </div>
                    </div>
                </div>
            </div>
            <div class="col-lg-4 property-section">
                <div class="property-panel-section">

```

```

<div class="property-panel-
header">Properties</div>
<div class="property-panel-content"
style="padding: 10px;">
    <table id="property" class="box-table"
title="Properties" style="width: 100%;">
        <tr>
            <td style="width:50%">Selected
Language: </td>
            <td class="formtext">{{
skillForm.get('skillname')?.value }}</td>
        </tr>
        <tr>
            <td style="width:50%">Buyer Name:
</td>
            <td class="formtext">{{
skillForm.get('sname')?.value }}</td>
        </tr>
        <tr>
            <td style="width:50%">Buyer Mail
ID: </td>
            <td class="formtext">{{
skillForm.get('smail')?.value }}</td>
        </tr>
    </table>
</div>
</div>
</div>
</div>
</div>
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[1] '>
    <ng-template #content>
        <div class="row" style="width:500px;">
            <div class="col-xs-7 col-sm-7 col-lg-7 col-md-7"
style="padding-top:20px;">
                <div style="webkit-box-shadow: 0 2px 2px 0
rgba(0,0,0,0.14), 0 3px 1px -2px rgba(0,0,0,0.12), 0 1px 5px 0
rgba(0,0,0,0.2);
box-shadow: 0 2px 2px 0 rgba(0,0,0,0.14), 0 3px 1px -2px
rgba(0,0,0,0.12), 0 1px 5px 0 rgba(0,0,0,0.2);
margin:0 auto;height:330px;">
                    <div style="padding:30px;">
                        <form #form="ngForm">
                            <div class="form-group">
                                <ejs-dropdownlist name="skillname"
[ (ngModel) ]='skillForm.skillname'
                                [dataSource]='skillset'
[placeholder]='placeholder' floatLabelType='Auto'>
                                    </ejs-dropdownlist>
                                </div>
                                <div class="form-group" style="padding-
top:10px;">
                                    <div class="e-float-input">
                                        <input type="text"
[ (ngModel) ]="skillForm.sname" name="sname"

```



```

                                maxlength="10" />
                                <span class="e-float-line e-
label-top"></span>
                                <label class="e-float-text e-
label-top">Name:</label>
                                </div>
                            </div>
                            <div class="form-group" style="padding-
top:10px;">
                                <div class="e-float-input">
                                    <input type="text"
                                        maxlength="15" />
                                    <span class="e-float-line e-
label-top"></span>
                                    <label class="e-float-text e-
label-top">Email:</label>
                                </div>
                            </div>
                            <div class="form-group"
style="padding:10px;">
                                <button ejjs-button>Done</button>
                                <button ejjs-button
type="reset">Cancel</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
            <div class="col-xs-5 col-sm-5 col-lg-5 col-md-5"
style="padding-top:20px;">
                <div style="webkit-box-shadow: 0 2px 2px 0
rgb(0,0,0,0.14), 0 3px 1px -2px rgb(0,0,0,0.12), 0 1px 5px 0
rgb(0,0,0,0.2);
        box-shadow: 0 2px 2px 0 rgb(0,0,0,0.14), 0 3px 1px -2px
rgb(0,0,0,0.12), 0 1px 5px 0 rgb(0,0,0,0.2);
        margin:0 auto;height:330px;">
                    <div style="padding:30px;">
                        <div class="form-group">
                            <span style="font-weight:800;">Selected
Language: </span>
                            <span style="font-
style:oblique;color:#8a2be2;">
                                {{ skillForm.skillname }}</span>
                        </div>
                        <div class="form-group" style="padding-
top:50px;">
                            <span style="font-weight:800;">Buyer
Name: </span>
                            <span style="font-
style:oblique;color:#8a2be2;">
                                {{ skillForm.sname }}</span>
                        </div>
                        <div class="form-group" style="padding-
top:50px;">
                            <span style="font-weight:800;">Buyer Mail
ID: </span>

```

```

<span style="font-
style:oblique;color:#8a2be2;">
        {{ skillForm.smail }}</span>
    </div>
</div>
</div>
</div>
</div>
</ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>
))
export class AppComponent {
    public headerText: Object | any = [{ 'text': 'ReactiveForms1' }, {
'text': 'ReactiveForms2' }];
    public skillset: string[] = [
        'ASP.NET', 'ActionScript', 'Basic',
        'C++', 'C#', 'dBase', 'Delphi',
        'ESPOL', 'F#', 'FoxPro', 'Java',
        'J#', 'Lisp', 'Logo', 'PHP'
    ];
    public placeholder: String = 'Select a language';
    public skillForm!: FormGroup;
    constructor(@Inject(FormBuilder) private builder: FormBuilder) {
        this.createForm();
    }
    createForm(): void {
        this.skillForm = this.builder.group({
            skillname: ['', Validators.required],
            sname: ['', Validators.required],
            smail: ['', Validators.required]
        });
    }
    onfocus(element: FocusEvent): void {
        let target: HTMLInputElement = element.target as HTMLInputElement;
        let parentNode: HTMLElement = target.parentNode as HTMLElement;
        if (parentNode.classList.contains('e-input-in-wrap')) {
            parentNode = (parentNode.parentNode as HTMLElement);
        }
        parentNode.classList.add('e-input-focus');
        (parentNode.children[2] as Element).classList.add('e-label-top');
        (parentNode.children[2] as Element).classList.remove('e-label-
bottom');
    }
    onblur(element: FocusEvent): void {
        let target: HTMLInputElement = element.target as HTMLInputElement;
        let parentNode: HTMLElement = target.parentNode as HTMLElement;
        if (parentNode.classList.contains('e-input-in-wrap')) {
            (parentNode.parentNode as HTMLElement).classList.remove('e-input-
focus');
        }
        parentNode.classList.remove('e-input-focus');
        if (target.value === null || target.value === '') {
            (parentNode.children[2] as Element).classList.remove('e-label-
top');

```

```

        (parentNode.children[2] as Element).classList.add('e-label-
bottom');
    } else {
        (parentNode.children[2] as Element).classList.add('e-label-top');
        (parentNode.children[2] as Element).classList.remove('e-label-
bottom');
    }
}
onreset(element: MouseEvent): void {
    let parentNode: NodeListOf<HTMLElement> =
document.getElementsByClassName('box-form')[0].querySelectorAll('.e-float-
text');
    for (let i: number = 0; i < parentNode.length; i++) {
        parentNode[i].classList.remove('e-label-top');
        parentNode[i].classList.add('e-label-bottom');
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Adding dynamic items with content reuse in Angular Tab component

You can add dynamic tabs by reusing the content using Angular **TemplateRef**. Tabs can be added dynamically by passing array of items and index value to the [addTab](#) method.

Content reuse can be achieved by using the following steps:

1. Create a TemplateRef variable in your component(app.component.ts) file.
2. Access the TemplateRef using ViewChild on the element.
3. Provide separate TemplateRef declaration for each angular component and pass content by dynamically adding tab. It will reuse the content of angular component.

Refer to the following sample.

APP.COMPONENT.HTML

```

<ng-template #DatePickertemplateRef>
    <date-picker></date-picker>
</ng-template>
<ng-template #FirstDropDowntemplateRef>
    <drop-down [data]="firstDropDownData"></drop-down>
</ng-template>
<ng-template #SecondDropDowntemplateRef>
    <drop-down [data]="secondDropDownData"></drop-down>
</ng-template>
<ng-template #ThirdDropDowntemplateRef>
    <drop-down [data]="thirdDropDownData"></drop-down>
</ng-template>

```

```

<button id='add' class='e-btn' (click)='addButtonClicked($event) '>Click to
add</button>
<button id='remove' class='e-btn' (click)='removeButtonClicked($event) '>Click
to remove</button>
<ejs-tab id="element" #element>
    <e-tabitems>
        <e-tabitem [header]='tabItemsHeaderText[0] '
[content]="DatePickertemplate"></e-tabitem>
        <e-tabitem [header]='tabItemsHeaderText[1] '
[content]="FirstDropDowntemplate"></e-tabitem>
        <e-tabitem [header]='tabItemsHeaderText[2] '
[content]="SecondDropDowntemplate"></e-tabitem>
    </e-tabitems>
</ejs-tab>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { TabAllModule } from '@syncfusion/ej2-angular-navigations'
import { Component, ViewChild, OnInit, TemplateRef } from '@angular/core';
import { createElement } from '@syncfusion/ej2-base';
import { TabComponent, SelectEventArgs } from '@syncfusion/ej2-angular-
navigations';
import { enableRipple } from '@syncfusion/ej2-base';
enableRipple(true);
/**
 * Content reuse
 */
@Component({
imports: [TabAllModule, FormsModule, DatePickerModule ],
standalone: true,
selector: 'my-app',
templateUrl: 'app/app.component.html',
styleUrls: ['app.component.css'],
})
export class AppComponent {
    @ViewChild('element') tabObj?: TabComponent;
    @ViewChild('DatePickertemplateRef') public DatePickertemplate?:
TemplateRef<any>;
    @ViewChild('FirstDropDowntemplateRef') public FirstDropDowntemplate?:
TemplateRef<any>;
    @ViewChild('SecondDropDowntemplateRef') public SecondDropDowntemplate?:
TemplateRef<any>;
    @ViewChild('ThirdDropDowntemplateRef') public ThirdDropDowntemplate?:
TemplateRef<any>;
    public firstDropDownData: string[] = [
        'Badminton',
        'Basketball',
        'Cricket',
        'Golf',
        'Hockey',
        'Rugby'
    ]
}

```

```

];
public secondDropDownData: string[] = [
    'Cricket',
    'Golf',
    'Hockey',
    'Rugby',
    'Badminton',
    'Basketball'
];
public thirdDropDownData: string[] = [
    'Rugby',
    'Badminton',
    'Basketball',
    'Cricket',
    'Golf',
    'Hockey'
];
public tabItemsHeaderText: Object = [{ 'text': 'DatePicker' }, { 'text':
'Dropdown' }, { 'text': 'Reused Dropdown' }];
public addButtonClicked(e: any): void {
    var newtabItem = [
        { header: { text: 'DynamicTabItem' }, content:
this.ThirdDropDowntemplate }
    ];
    (this.tabObj as TabComponent).addTab(newtabItem as any,1);
}
public removeButtonClicked(e: any): void {
    (this.tabObj as TabComponent).removeTab(1);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize tab content height in Angular Tab component

You can change the Tab content height by using the [heightAdjustMode](#) property. By default, the Tab content [heightAdjustMode](#) property is set to **Content** value.

- **None:** Each tab content height is set based on the Tab height. This value is used only the tab component having the [height](#) property.
- **Auto:** Each tab content height will take the maximum height of all other tabs content.
- **Content:** Each tab content height is set based on their own content.
- **Fill:** Each tab content height is set based on the full height of Tabs parent element.

APP.COMPONENT.HTML

```

<div id="wrapper" style='margin-top: 20px'>
  <div id='content' style="margin: 0px auto">
    <div id="default" style="padding-top:20px;width:250px">

```

```

        <div class='row'>
            <div>
                <label>Height Adjust Mode</label>
            </div><br />
            <div>
                <ejs-dropdownlist id='contentHeight' #contentHeight
[dataSource]='heightData'
                (change)='onChange($event)' [value]='value'
[fields]='fields' [popupHeight]='height'>
                </ejs-dropdownlist>
            </div>
        </div>
    </div>
<br />
<div>
    <ejs-tab id="element" #element height="400px">
        <e-tabitems>
            <e-tabitem [header]='headerText[0] '>
                <ng-template #content>
                    Twitter is an online social networking service
that enables users to send and read short
140-character messages called "tweets".
                    Registered users can read and post tweets, but
those who are unregistered can only read
                    them. Users access Twitter
                    through the website interface, SMS or mobile
device app Twitter Inc. is based in San
                    Francisco and has more than 25
                    offices around the world. Twitter was created in
March 2006 by Jack Dorsey, Evan Williams,
                    Biz Stone, and Noah Glass
                    and launched in July 2006. The service rapidly
gained worldwide popularity, with more than
                    100 million users posting
                    340 million tweets a day in 2012.The service also
handled 1.6 billion search queries per
                    day.
                </ng-template>
            </e-tabitem>
            <e-tabitem [header]='headerText[1] '>
                <ng-template #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 and fellow students Eduardo
                    Saverin, Andrew McCollum,
                    Dustin Moskovitz and Chris Hughes.The founders
had initially limited the website\'\'s
                    membership to Harvard students,
                    but later expanded it to colleges in the Boston
area, the Ivy League, and Stanford
                    University. It gradually added support
for students at various other universities and
                    later to high-school students.
                </ng-template>
            </e-tabitem>
        </e-tabitems>
    </ejs-tab>

```

```

        <e-tabitem [header]='headerText[2] '>
            <ng-template #content>
                WhatsApp Messenger is a proprietary cross-
platform instant messaging client for smartphones
                that operates under a subscription
                business model. It uses the Internet to send text
messages, images, video, user location and
                audio media messages to
                other users using standard cellular mobile
numbers. As of February 2016, WhatsApp had a user
                base of up to one billion, [10]
                making it the most globally popular messaging
application. WhatsApp Inc., based in Mountain
                View, California, was acquired
                by Facebook Inc. on February 19, 2014, for
approximately US$19.3 billion.
            </ng-template>
        </e-tabitem>
    </e-tabitems>
</ejs-tab>
</div>
</div>
</div>

```

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabComponent, TabItemsDirective, TabItemDirective } from
'@syncfusion/ej2-angular-navigations'
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns'
import { Component, ViewChild } from '@angular/core';
import { DropDownListComponent, ChangeEventArgs } from '@syncfusion/ej2-
angular-dropdowns';
import { HeightStyles, TabComponent } from '@syncfusion/ej2-angular-
navigations';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-container',
  // specifies the template url path
  templateUrl: 'app/app.component.html'
})
export class AppComponent {
  @ViewChild('element') tabObj?: TabComponent;
  @ViewChild('contentHeight') listObj?: DropDownListComponent;
  public heightData: Object[] = [
    { mode: 'None', text: 'None' },
    { mode: 'Content', text: 'Content' },
    { mode: 'Fill', text: 'Fill' },
    { mode: 'Auto', text: 'Auto' }
  ];
  public fields: Object = { text: 'text', value: 'mode' };

```

```

    public height: string = '220px';
    public value: string = 'Content';
    public onChange(args: ChangeEventArgs): void {
        (this.tabObj as TabComponent).heightAdjustMode = (this.listObj as
        DropDownListComponent).value as string as HeightStyles;
    }
    public headerText: Object = [{ 'text': 'Twitter' }, { 'text': 'Facebook'
    }, { 'text': 'WhatsApp' }];
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Reorder active tab in Angular Tab component

We can able to prevent the changing of the active tab item on resizing the browser when overflow mode is popup by using the [reorderActiveTab](#) property. By default, the active Tab should be reordered when we click the tab items from the popup. If we set `false` to [reorderActiveTab](#) property the active tab item from the popup will not be reordered and an active item is highlighted inside the popup. The following code example depicts to prevent the reorder active tab item inside the popup.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component, OnInit } from '@angular/core';
import { Tab, TabComponent } from '@syncfusion/ej2-angular-navigations';
/**
 * Add nested Tabs
 */
@Component({
  imports: [
    FormsModule, TabModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
    <ejs-tab id="element" #tab [items]='tabItems' overflowMode='Popup'
    heightAdjustMode='Auto' [reorderActiveTab]='reorderActiveTab'>
    </ejs-tab>`,
})
export class AppComponent implements OnInit {
  public tabItems?: Object[];
  reorderActiveTab?: boolean;
  public ngOnInit(): void {
    this.reorderActiveTab = false;
    this.tabItems = [
      {
        header: { text: 'India' },
        content:

```



```

    'India officially the Republic of India, is a country in South
    Asia. It is the seventh-largest country by area, the second-most populous
    country with over 1.2 billion people, and the most populous democracy in the
    world. Bounded by the Indian Ocean on the south, the Arabian Sea on the
    south-west, and the Bay of Bengal on the south-east, it shares land borders
    with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and
    Burma and Bangladesh to the east. In the Indian Ocean, India is in the
    vicinity of Sri Lanka and the Maldives; in addition, India Andaman and
    Nicobar Islands share a maritime border with Thailand and Indonesia.',
    },
    {
      header: { text: 'Canada' },
      content:
        'Canada is a North American country stretching from the U.S. in the
        south to the Arctic Circle in the north. Major cities include massive
        Toronto, west coast film centre Vancouver, French-speaking Montréal and
        Québec City, and capital city Ottawa. Canada vast swaths of wilderness
        include lake-filled Banff National Park in the Rocky Mountains. It also home
        to Niagara Falls, a famous group of massive waterfalls.',
    },
    {
      header: { text: 'Australia' },
      content:
        "Australia, officially the Commonwealth of Australia, is a country
        comprising the mainland of the Australian continent, the island of Tasmania
        and numerous smaller islands. It is the world sixth-largest country by total
        area. Neighboring countries include Indonesia, East Timor and Papua New
        Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the
        north-east; and New Zealand to the south-east. <br/><br/>India is a vast
        South Asian country with diverse terrain - from Himalayan peaks to Indian
        Ocean coastline - and history reaching back 5 millennia. In the north, Mughal
        Empire and marks include Delhi's Red Fort complex and massive Jama Masjid
        mosque, plus Agras iconic Taj Mahal mausoleum. Pilgrims bathe in the Ganges
        in Varanasi, and Rishikesh is a yoga centre and base for Himalayan
        trekking.",
    },
    {
      header: { text: 'USA' },
      content:
        'The United States of America (USA or U.S.A.), commonly called the
        United States (US or U.S.) and America, is a federal republic consisting of
        fifty states and a federal district. The 48 contiguous states and the federal
        district of Washington, D.C. are in central North America between Canada and
        Mexico. The state of Alaska is west of Canada and east of Russia across the
        Bering Strait, and the state of Hawaii is in the mid-North Pacific. The
        country also has five populated and nine unpopulated territories in the
        Pacific and the Caribbean.',
    },
    {
      header: { text: 'London' },
      content:
        'London, the capital of England and the United Kingdom, is a 21st-
        century city with history stretching back to Roman times. At its centre stand
        the imposing Houses of Parliament, the iconic 'Big Ben' clock tower and
        Westminster Abbey, site of British monarch coronations. Across the Thames
        River, the London Eye observation wheel provides panoramic views of the South
        Bank cultural complex, and the entire city.',
    },
  ],

```

```

    },
    {
      header: { text: 'Germany' },
      content:
        'Germany is a Western European country with a landscape of forests, rivers, mountain ranges and North Sea beaches. It has over 2 millennia of history. Berlin, its capital, is home to art and nightlife scenes, the Brandenburg Gate and many sites relating to WWII. Munich is known for its Oktoberfest and beer halls, including the 16th-century Hofbräuhaus. Frankfurt, with its skyscrapers, houses the European Central Bank.',
    },
    {
      header: { text: 'France' },
      content:
        'France, officially the French Republic is a sovereign state comprising territory in western Europe and several overseas regions and territories. The European part of France, called Metropolitan France, extends from the Mediterranean Sea to the English Channel and the North Sea, and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo metres and as of August 2015 has a population of 67 million, counting all the overseas departments and territories.',
    },
    {
      header: { text: 'Sweden' },
      content:
        'Sweden is a Scandinavian nation with thousands of coastal islands and inland lakes, along with vast boreal forests and glaciated mountains. Its principal cities, eastern capital Stockholm and southwestern Gothenburg and Malmö, are all coastal. Stockholm is built on 14 islands. It has more than 50 bridges, as well as the medieval old town, Gamla Stan, royal palaces and museums such as open-air Skansen.',
    },
    {
      header: { text: 'Africa' },
      content:
        'Africa is the world second-largest and second-most-populous continent. At about 30.3 million km2 including adjacent islands, it covers 6% of Earth total surface area and 20.4% of its total land area',
    },
    {
      header: { text: 'Japan' },
      content:
        'Japan is an island nation in the Pacific Ocean with dense cities, imperial palaces, mountainous national parks and thousands of shrines and temples. Shinkansen bullet trains connect the main islands of Kyushu, Honshu (home to Tokyo and Hiroshima's atomic-bomb memorial) and Hokkaido (famous for skiing). Tokyo, the capital, is known for skyscrapers, shopping and pop culture.',
    },
    {
      header: { text: 'Malaysia' },
      content:
        'Malaysia is a Southeast Asian country occupying parts of the Malay Peninsula and the island of Borneo. It known for its beaches, rainforests and mix of Malay, Chinese, Indian and European cultural influences. The capital, Kuala Lumpur, is home to colonial buildings, busy shopping districts such as

```

```

Bukit Bintang and skyscrapers such as the iconic, 451m-tall Petronas Twin
Towers.',
    },
    {
      header: { text: 'Singapore' },
      content:
        'Singapore, an island city-state off southern Malaysia, is a global
        financial center with a tropical climate and multicultural population. Its
        colonial core centers on the Padang, a cricket field since the 1830s and now
        flanked by grand buildings such as City Hall, with its 18 Corinthian columns.
        In Singapore circa-1820 Chinatown stands the red-and-gold Buddha Tooth Relic
        Temple, said to house one of Buddha teeth.',
    },
  ];
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Tab selection in Angular Tab component

We can able to find the tab selection whether it is selected by user interaction or programmatically way in the [selecting](#) and [selected](#) event argument with the field of `isInteracted`. When the user changes the tab through click actions it will return true otherwise, it will return false. The following code example depicts to find the tab selecting the state in selecting and selected events.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TabComponent, TabItemsDirective, TabItemDirective } from
 '@syncfusion/ej2-angular-navigations'
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns'
import { Component, OnInit, ViewChild, ElementRef } from '@angular/core';
import { DropDownListComponent, DropDownList } from '@syncfusion/ej2-angular-
dropdowns';
import { Tab, TabComponent, SelectingEventArgs, SelectEventArgs } from
 '@syncfusion/ej2-angular-navigations';
/**
 * Add nested Tabs
 */
@Component({
  imports: [
    ],
  standalone: true,
  selector: 'app-container',
  template: `<div>
    <div class="EventLog" id="EventLog" style="word-break:
normal;padding: 5px;"><b>{{eventLog}}</b></div>

```

```

    <ejs-dropdownlist id='selectDropdown' [dataSource]='dropdownData'
    (change)='onChange($event)' [value]='value' [fields]='fields'
    [placeholder]='waterMark' [popupHeight]='height'></ejs-dropdownlist>
    <ejs-tab id="element" #tabObj [items]='tabItems'
    heightAdjustMode= 'Auto' (selecting)="selecting($event)"
    (selected)="selected($event)">
    </ejs-tab></div>`,
  ))
export class AppComponent implements OnInit {
  @ViewChild('tabObj')
  public tabObj?: TabComponent;
  public eventLog: string = '';
  public tabItems?: Object[];
  public dropdownData: Object[] = [
    { text: 'India', Id: 0 },
    { text: 'Canada', Id: 1 },
    { text: 'Australia', Id: 2 },
    { text: 'USA', Id: 3 },
    { text: 'London', Id: 4 },
    { text: 'Germany', Id: 5 },
    { text: 'France', Id: 6 },
  ];
  public fields: Object = { text: 'text', value: 'Id' };
  public waterMark: string = 'Select Tab Item using dropdown';
  public value: number = 0;
  public onChange(args: any): void {
    (this.tabObj as TabComponent).select(args.value);
  }
  public ngOnInit(): void {
    this.tabItems = [
      {
        header: { text: 'India' },
        content:
          'India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.',
      },
      {
        header: { text: 'Canada' },
        content:
          'Canada is a North American country stretching from the U.S. in the south to the Arctic Circle in the north. Major cities include massive Toronto, west coast film centre Vancouver, French-speaking Montréal and Québec City, and capital city Ottawa. Canada vast swaths of wilderness include lake-filled Banff National Park in the Rocky Mountains. It also home to Niagara Falls, a famous group of massive waterfalls.',
      },
      {
        header: { text: 'Australia' },
        content:

```

```

    'Australia, officially the Commonwealth of Australia, is a country
    comprising the mainland of the Australian continent, the island of Tasmania
    and numerous smaller islands. It is the world sixth-largest country by total
    area. Neighboring countries include Indonesia, East Timor and Papua New
    Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the
    north-east; and New Zealand to the south-east. <br/><br/>India is a vast
    South Asian country with diverse terrain - from Himalayan peaks to Indian
    Ocean coastline - and history reaching back 5 millennia. In the north, Mughal
    Empire landmarks include Delhi's Red Fort complex and massive Jama Masjid
    mosque, plus Agra's iconic Taj Mahal mausoleum. Pilgrims bathe in the Ganges
in Varanasi, and Rishikesh is a yoga centre and base for Himalayan
    trekking.',
    },
    {
      header: { text: 'USA' },
      content:
        'The United States of America (USA or U.S.A.), commonly called the
        United States (US or U.S.) and America, is a federal republic consisting of
        fifty states and a federal district. The 48 contiguous states and the federal
        district of Washington, D.C. are in central North America between Canada and
        Mexico. The state of Alaska is west of Canada and east of Russia across the
        Bering Strait, and the state of Hawaii is in the mid-North Pacific. The
        country also has five populated and nine unpopulated territories in the
        Pacific and the Caribbean.',
    },
    {
      header: { text: 'London' },
      content:
        'London, the capital of England and the United Kingdom, is a 21st-
        century city with history stretching back to Roman times. At its centre stand
        the imposing Houses of Parliament, the iconic 'Big Ben' clock tower and
        Westminster Abbey, site of British monarch coronations. Across the Thames
        River, the London Eye observation wheel provides panoramic views of the South
        Bank cultural complex, and the entire city.',
    },
    {
      header: { text: 'Germany' },
      content:
        'Germany is a Western European country with a landscape of forests,
        rivers, mountain ranges and North Sea beaches. It has over 2 millennia of
        history. Berlin, its capital, is home to art and nightlife scenes, the
        Brandenburg Gate and many sites relating to WWII. Munich is known for its
        Oktoberfest and beer halls, including the 16th-century Hofbräuhaus.
        Frankfurt, with its skyscrapers, houses the European Central Bank.',
    },
    {
      header: { text: 'France' },
      content:
        'France, officially the French Republic is a sovereign state
        comprising territory in western Europe and several overseas regions and
        territories. The European part of France, called Metropolitan France, extends
from the Mediterranean Sea to the English Channel and the North Sea, and from
        the Rhine to the Atlantic Ocean; France covers 640,679 square kilo metres and
as of August 2015 has a population of 67 million, counting all the overseas
        departments and territories.',
    },
  ];

```

```

    }
    public selecting(args: SelectingEventArgs) {
        this.getInteractionDetail(args.isInteracted as boolean);
    }
    public selected(args: SelectEventArgs) {
        this.getInteractionDetail(args.isInteracted as boolean);
    }
    public getInteractionDetail(interact: boolean): void {
        let eventlog = interact
            ? 'Tab Item selected by user interaction'
            : 'Tab Item selected by programmatically';
        this.eventLog = eventlog;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show hide tab item in Angular Tab component

The [hideTab](#) method is used to show or hide the item that is in the specified index. In the following demo, specified tab item is show or hide dynamically when the Hide/Show Item button is clicked.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild } from '@angular/core';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
  imports: [
    FormsModule, TabModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-container',
  template: `
<div class="control-section e-tab-section">
  <div class="e-sample-resize-container">
    <button
      #addButton
      ej2-button
      id="addButton"
      type="button"
      content="Hide/Show Tab Item"
      (click)="onButtonClick()"
    ></button>
    <!-- Render the Tab Component -->
    <ejs-tab #tabObj id="tab_default" heightAdjustMode="Auto">

```

	<pre> <e-tabitems> <e-tabitem [header]="headerText[0]"> <ng-template #content> Twitter is an online social networking service that enables users </pre>
to	
	<pre> send and read short 140-character messages called "tweets". Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is based </pre>
in	
Twitter	<pre> San Francisco and has more than 25 offices around the world. </pre>
Stone,	<pre> was created in March 2006 by Jack Dorsey, Evan Williams, Biz </pre>
gained	<pre> and Noah Glass and launched in July 2006. The service rapidly </pre>
340	<pre> worldwide popularity, with more than 100 million users posting </pre>
	<pre> million tweets a day in 2012.The service also handled 1.6 billion search queries per day. </ng-template> </e-tabitem> <e-tabitem [header]="headerText[1]"> <ng-template #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 and fellow students Eduardo Saverin, Andrew McCollum, Dustin </pre>
Moskovitz	<pre> and Chris Hughes.The founders had initially limited the </pre>
website\'\'s	<pre> membership to Harvard students, but later expanded it to colleges </pre>
in	<pre> the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other </pre>
universities	<pre> and later to high-school students. </ng-template> </e-tabitem> <e-tabitem [header]="headerText[2]"> <ng-template #content> WhatsApp Messenger is a proprietary cross-platform instant </pre>
messaging	<pre> client for smartphones that operates under a subscription </pre>
business	<pre> model. It uses the Internet to send text messages, images, video, user location and audio media messages to other users using </pre>
standard	<pre> cellular mobile numbers. As of February 2016, WhatsApp had a user base of up to one billion,[10] making it the most globally </pre>
popular	<pre> messaging application. WhatsApp Inc., based in Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, </pre>
for	<pre> approximately US\$19.3 billion. </ng-template> </e-tabitem> </pre>

```

        </e-tabitems>
    </ejs-tab>
</div>
</div>

}))
export class AppComponent {
    @ViewChild('tabObj')
    public tabObj?: TabComponent;
    @ViewChild('addButton')
    public addButton?: ButtonComponent;
    public isBool: boolean = false;
    // Mapping Tab items Header property
    public headerText: Object = [
        { text: 'Twitter', iconCss: 'e-twitter' },
        { text: 'Facebook', iconCss: 'e-facebook' },
        { text: 'WhatsApp', iconCss: 'e-whatsapp' },
    ];
    public onClick(): void {
        this.isBool = !this.isBool;
        (this.tabObj as TabComponent).hideTab(0, this.isBool);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Enabling tab key navigation in Tabs

The [tabIndex](#) property of a Tab item is used to enable tab key navigation for that particular item. When a positive value is assigned to the [tabIndex](#) property, it allows the user to switch focus to the next or previous tab item using the Tab or Shift+Tab keys. By default, the user can only switch between tab items using the arrow keys.

If the [tabIndex](#) value is set to 0 for all tab items, the tab will switch based on the order of the elements on the page. This means that if the tab items are listed in a specific order on the page, the user will be able to navigate through them using the Tab key in that same order.

To use the [tabIndex](#) property, you can assign a positive value to the property of each tab item that you want to enable tab key navigation. For example:

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TabModule } from '@syncfusion/ej2-angular-navigations'
import { Component } from '@angular/core';
import { TabComponent } from '@syncfusion/ej2-angular-navigations';
@Component({
    imports: [
        FormsModule, TabModule
    ]
})

```



```

],
standalone: true,
  selector: 'app-container',
  template: `<ejs-tab id="element" heightAdjustMode='Auto'
overflowMode='Scrollable' width='300px'>
    <e-tabitems>
        <e-tabitem [header]='headerText[0]' tabIndex='0'>
            <ng-template #content>
                HyperText Markup Language, commonly referred
to as HTML, is the standard markup language used to create web pages. Along
                with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging web
                pages, user interfaces for web applications,
and user interfaces for many mobile applications. Web browsers can read
                HTML files and render them into visible or
audible web pages. HTML describes the structure of a website semantically
                along with cues for presentation, making it a
markup language, rather than a programming language.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[1]' tabIndex='0'>
            <ng-template #content>
                C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
                is led
                by Anders Hejlsberg. The most recent version
is C# 5.0, which was released on August 15, 2012.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[2]' tabIndex='0'>
            <ng-template #content>
                Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle
                Corporation,
                that provides a system for developing
application software and deploying it in a cross-platform computing
                environment.
                Java is used in a wide variety of computing
platforms from embedded devices and mobile phones to enterprise servers
                and supercomputers. While less common, Java
applets run in secure, sandboxed environments to provide many features
                of native applications and can be embedded in
HTML pages.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[3]' tabIndex='0'>
            <ng-template #content>
                The command-line compiler, VBC.EXE, is
installed as part of the freeware .NET Framework SDK. Mono also includes a
                command-line
                VB.NET compiler. The most recent version is
VB 2012, which was released on August 15, 2012.
            </ng-template>
        </e-tabitem>
        <e-tabitem [header]='headerText[4]' tabIndex='0'>
            <ng-template #content>

```

Xamarin **is** a San Francisco, California based software company created **in** May **2011**[3] by the engineers that created Mono, [4]

Mono **for** Android and MonoTouch that are cross-platform implementations of the Common Language Infrastructure (CLI) and Common Language **Specifications** (often called Microsoft .NET). With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms.[5]

Xamarin has over **1** million developers **in** more than **120** countries around the World **as** of May **2015**.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[5]' tabIndex='0'>
  <ng-template #content>
```

ASP.NET **is** an open-source server-side web application framework designed **for** web development to produce **dynamic** web pages.

It was developed by Microsoft to allow programmers to build **dynamic** web sites, web applications and web services.

It was first released **in** January **2002** with version **1.0** of the .NET Framework, and **is** the successor to Microsoft\'s Active Server **Pages** (ASP) technology. ASP.NET **is** built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code **using any** supported .NET language. The ASP.NET SOAP extension framework allows ASP.NET components to process SOAP messages.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[6]' tabIndex='0'>
  <ng-template #content>
```

The ASP.NET MVC **is** a web application framework developed by Microsoft, which implements the model-view-controller (MVC) pattern.

It **is** open-source software, apart **from** the ASP.NET Web Forms component which **is** proprietary. In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform **using only** Razor pages) will merge **into** a unified MVC **6**.The project **is** called ASP.NET vNext.

```
</ng-template>
</e-tabitem>
<e-tabitem [header]='headerText[7]' tabIndex='0'>
  <ng-template #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.[5] More recently, however, it has become common **in** both game development and the creation of desktop applications.

```
</ng-template>
</e-tabitem>
</e-tabitems>
</ejs-tab>
```

```

    })
    export class AppComponent {
        public headerText: Object = [{ text: 'HTML' }, { text: 'C Sharp(C#)' }, {
text: 'Java' }, { text: 'VB.Net' },
            { text: 'Xamarin' }, { text: 'ASP.NET' }, { text: 'ASP.NET MVC' }, {
text: 'JavaScript' }];
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

With this code, the user will be able to switch between the tab items using the Tab and Shift+Tab keys, in the order specified by the [tabIndex](#) values.

It's important to note that the [tabIndex](#) property only affects the ability to navigate between tab items using the Tab key. The user will still be able to use the arrow keys to switch between tab items, regardless of the value of the [tabIndex](#) property.

Ej1 api migration in Angular Tab component

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

Accessibility and Localization

<!-- markdownlint-disable MD033 -->

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

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

| Keyboard Navigation | **Property** : allowKeyboardNavigation
 <ej-tab id="Tab"
[allowKeyboardNavigation]="false" ></ej-tab> | Not Applicable |

| Localization | Not Applicable | **Property** : locale
 <ejs-tab id="Tab" locale="fr-
BE"></ejs-tab> |

| Right to left | **Property**: enableRTL
 <ej-tab id="tab" [enableRTL]="true"></ej-tab> | **Property**:
enableRTL
 <ejs-tab id='tab' [enableRTL]='true'> </ejs-tab> |

AjaxSettings

<!-- markdownlint-disable MD033 -->

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

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

| Default | **Property** : ajaxSettings
 <ej-tab id="tab" [ajaxSettings.type]="GET"></ej-tab> |
Not Applicable |

| Asynchronous | **Property** : ajaxSettings.async
 <ej-tab id="tab" [ajaxSettings.async]="true"
></ej-tab> | Not Applicable |

| Browser Cache | **Property:** ajaxSettings.cache
 <ej-tab id="tab" [ajaxSettings.cache]="false"></ej-tab> | Not Applicable |

| Request type | **Property:** ajaxSettings.contentType
 <ej-tab id="tab" [ajaxSettings.contentType]="html"></ej-tab> | Not Applicable |

| Data | **Property:** ajaxSettings.data
 <ej-tab id="tab" [ajaxSettings.data]={ }></ej-tab> | Not Applicable |

| Response type | **Property:** ajaxSettings.dataType
 <ej-tab id="tab" [ajaxSettings.dataType]="html"></ej-tab> | Not Applicable |

| HTTP request type | **Property:** ajaxSettings.type
 <ej-tab id="tab" [ajaxSettings.type]="GET"></ej-tab> | Not Applicable |

| AjaxBeforeLoad | **Event:** ajaxBeforeLoad
 <ej-tab id='tab' (ajaxBeforeLoad)='onajaxBeforeLoad(\$event)'></ej-tab>
 TS:
 onajaxBeforeLoad (event){ } | Not Applicable |

| AjaxError | **Event:** AjaxError
 <ej-tab id='tab' (ajaxError)='onajaxError(\$event)'></ej-tab>
 TS:
 onajaxError (event){ } | Not Applicable |

| AjaxLoad | **Event:** ajaxLoad
 <ej-tab id='tab' (ajaxLoad)='onajaxLoad(\$event)'></ej-tab>
 TS:
 onajaxLoad (event){ } | Not Applicable |

| AjaxSuccess | **Event:** ajaxSuccess
 <ej-tab id='tab' (ajaxSuccess)='onajaxSuccess(\$event)'></ej-tab>
 TS:
 onajaxSuccess (event){ } | Not Applicable |

Animation

<!-- markdownlint-disable MD033 -->

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

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

| Default | Not Applicable | **Property:** animation
 <ejs-tab id="tab" [animation]="animation"></ejs-tab>
 TS:
 public animation: Object[] = [{ prev: { }, next: { } }] |

| EnableAnimation | **Property:** animation
 <ej-tab id="tab" [enableAnimation]="false"></ej-tab>
 | Not Applicable |

| Previous animation | Not Applicable | **Property:** animation.prev
 <ejs-tab id='tab' [animation]="animation"></ejs-tab>
 TS:
 public animation: Object[] = [{prev: { duration: 400,easing: 'ease-in', effect: 'SlideLeft'}}] |

| Next animation | Not Applicable | **Property:** animation.next
 <ejs-tab id='tab' [animation]="animation"></ejs-tab>
 TS:
 public animation: Object[] = [{next: {duration: 400, easing: 'ease-in', effect: 'SlideLeft' } }] |

| Duration [prev / next] | Not Applicable | **Property:** animation.collapse.duration
 <ejs-tab id='tab' [animation]="animation"></ejs-tab>
 TS:
 public animation: Object[] = [{prev: { duration: 400 } }] |

| Easing [expand / collapse] | **Not Applicable** | **Property** : animation.next.easing
 **<ejs-tab id='tab' [animation]="animation"></ejs-tab>
TS:
public animation: Object[] = [{prev: { easing: 'ease-in' } }]** |

| Effect [expand / collapse] | **Not Applicable** | **Property** : animation.next.effect
 **<ejs-tab id='tab' [animation]="animation"></ejs-tab>
TS:
public animation: Object[] = [{prev: { effect: 'SlideLeft' } }]** |

Header

<!-- markdownlint-disable MD033 -->

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

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

| Header position | **Property** : headerPosition
 <ej-tab id="Tab" [headerPosition]="Bottom"></ej-tab> | **Property** : headerPlacement
 <ej-tab id="Tab" [headerPlacement]="Bottom"></ej-tab> |

| Header size | **Property** : headerSize
 <ej-tab id="Tab" [headerSize]="100px"></ej-tab> | **Not Applicable** |

| OverflowModes | **Not Applicable** | **Property**: overflowMode
 <ej-tab id='tab' [overflowMode]='Popup'></ej-tab> |

| TabScroll | **Property**: enableTabScroll
 <ej-tab id="Tab" [enableTabScroll]="false"></ej-tab> | **Not Applicable** |

Items

<!-- markdownlint-disable MD033 -->

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

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

| Default | **Not Applicable** | **Property** : items
 <ej-tab id="tab" [items]="items"></ej-tab> |

| Content | **Not Applicable** | **Property** : items[0].content
 **<ej-tab id="tab" [items]="items"></ej-tab>
TS:
public items: Object[] = [{ content: 'Contents'}];|**

| Custom class | **Not Applicable** | **Property** : items[0].cssClass
 **<ej-tab id="tab" [items]="items"></ej-tab>
TS:
public items: Object[] = [{ cssClass: 'customClass'}];|**

| Header | **Not Applicable** | **Property** : items[0].Header
 **<ej-tab id="tab" [items]="items"></ej-tab>
TS:
public items: Object[] = [{ header: ' ' }];|**

| Icon class | **Not Applicable** | **Property** : items[0].header.iconCss
 **<ej-tab id="tab" [items]="items"></ej-tab>
TS:
public items: Object[] = [{ header: { iconCss: 'e-icon' } }];|**

| Icon position | **Not Applicable** | **Property** : items[0].header.iconPosition
 **<ej-tab id="tab" [items]="items"></ej-tab>
TS:
public items: Object[] = [{ header: { iconPosition: 'Left' } }];|**

| Header text | **Not Applicable** | **Property** : items[0].header.text **
** **<ej-tab id="tab" [items]="items"></ej-tab>
** **TS:****
** public items: Object[] = [{header: { text: 'Tab1' }}];|

| Get items length | **Method** : getItemCount() **
** **<ej-tab id="Tab" #Tab></ej-tab>
** **TS:****
@ViewChild('Tab') public TabObj: TabComponent;
TabObj.getItemCount();| **Not Applicable|

| Add Items | **Method** : addItem(url, displayLabel, index, cssClass, id) **
** **<ej-tab id="Tab" #Tab></ej-tab>
** **TS:****
@ViewChild('Tab') public TabObj: TabComponent;
TabObj.addItem("#new", "New Item", 3, "myClass", "newItem");| **Method :addTab(items, index) **
** **<ej-tab id="tab" #Tab> </ej-tab>
** **TS:****
** @ViewChild('Tab') public TabObj: TabComponent;**
**TabObj.addTab([{ header: { text: 'Tab1' },content: 'contents' }], 1);|

| BeforeAdd | **Not Applicable** | **Event:** adding**
** **<ej-tab id="tab" #Tab (adding)='onadding(\$event)'> </ej-tab>
** **TS:****
** onadding(event){ } |

| AfterAdd | **Event:** itemAdd**
** **<ej-tab id='tab' (itemAdd)='onitemAdd(\$event)'></ej-tab>
** **TS:****
** onitemAdd(event){ } | **Event:** added **
** **<ej-tab id="tab" (added)='onadded(\$event)'></ej-tab>
** **TS:****
** onadded(event){ } |

| Remove Item | **Method** : removeItem(index) **
** **<ej-tab id="Tab" #Tab></ej-tab>
** **TS:****
@ViewChild('Tab') public TabObj: TabComponent;
TabObj.removeItem(0);| **Method :addItem(items, index) **
** **<ej-tab id="tab" #Tab> </ej-tab>
** **TS:****
** @ViewChild('Tab') public TabObj: TabComponent;**
**TabObj.addItem([{ header: 'App', content: 'text' }], 0);|

| BeforeRemove | **Event:** beforeItemRemove **
** **<ej-tab id='tab' (beforeItemRemove)='onbeforeItemRemove(\$event)'></ej-tab>
** **TS:****
** onbeforeItemRemove(event){ } | **Event:** removing **
** **<ej-tab id="tab" #Tab (removing)='onremoving(\$event)'></ej-tab>
** **TS:****
** onremoving(event){ } |

| AfterRemove | **Event:** afterRemove **
** **<ej-tab id='tab' (itemRemove)='onitemRemove(\$event)'></ej-tab>
** **TS:****
** onitemRemove(event){ } | **Event:** removed **
** **<ej-tab id="tab" #Tab (removed)='onremoved(\$event)'></ej-tab>
** **TS:****
** onremoved(event){ } |

| Select item | **Not Applicable** | **Method** :select(index)**
** **<ej-tab id="tab" #Tab> </ej-tab>
** **TS:****
** @ViewChild('Tab') public TabObj: TabComponent;**
**TabObj.select(1);|

| SelectedItemIndex | **Property** : selectedItemIndex **
** **<ej-tab id="tab" selectedItemIndex="1" ></ej-tab>** | **Property** : selectedItem **
** **<ej-tab id="tab" selectedItem="1" > </ej-tab>** |

| BeforeActive | **Event:** beforeActive**
** **<ej-tab id="tab" #Tab (beforeActive)='onbeforeActive(\$event)'></ej-tab>
** **TS:****
** onbeforeActive(event){ } | **Event:** selecting**
** **<ej-tab id="tab" #Tab (selecting)='onselecting(\$event)'></ej-tab>
** **TS:****
** onselecting(event){ } |

| AfterActive | **Event:** itemActive**
** **<ej-tab id="tab" #Tab (itemActive)='onitemActive(\$event)'></ej-tab>
** **TS:****
** onitemActive(event){ } | **Event:** selected**
** **<ej-tab id="tab" #Tab (selected)='onselected(\$event)'></ej-tab>
** **TS:****
** onselected(event){ } |

| Disable items | **Property** : disabledItemIndex
 <ej-tab id="tab" disabledItemIndex="[1,2]" >
</ej-tab> | Not Applicable |

| Enable items | **Property** : enabledItemIndex
 <ej-tab id="tab" enabledItemIndex="[1,2]" >
</ej-tab> | Not Applicable |

| Enable/Disable item | Not Applicable | **Property** : items[0].disabled
 <ejs-tab id="tab"
[items]="items" > </ejs-tab>
 TS:
public items: Object[] = [{ disabled: true }]; |

| Hide items | **Property** : hiddenItemIndex
 <ej-tab id="tab" hiddenItemIndex="[1,2]" > </ej-
tab> | Not Applicable |

| Hide item | **Method** : hideItem(index)
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
TabObj.hideItem(1); | **Method**
:hideTab(index, true)
 <ejs-tab id="tab" #Tab> </ejs-tab>
 TS:
 @ViewChild('Tab')
public TabObj: TabComponent;
TabObj.hideTab(1, true); |

| Show item | **Method** : showItem(index)
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
TabObj.showItem(1); | **Method** :
hideTab(index, false)
 <ejs-tab id="tab" #Tab> </ejs-tab>
 TS:
 @ViewChild('Tab')
public TabObj: TabComponent;
 TabObj.hideTab(1, false); |

Common

<!-- markdownlint-disable MD033 -->

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

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

| Collapse active item | **Property** : collapsible
 <ej-tab id="tab" [collapsible]="true"> </ej-tab> |
Not Applicable |

| Custom class | **Property** : cssClass
 <ej-tab id="tab" cssClass="customClass" > </ej-tab> |
Property : cssClass
 <ejs-tab id="tab" cssClass="customClass" > </ejs-tab> |

| Enabled | **Property** : enabled
 <ej-tab id="tab" enabled="false"> </ej-tab> | **Method** :
disable(false)
 <ej-tab id="Tab" #Tab></ej-tab>
 TS:
@ViewChild('Tab') public TabObj:
TabComponent;
TabObj.disable(false); |

| Persistence | **Property** : enablePersistence
 <ej-tab id="tab" [enablePersistence]="false" >
</ej-tab> | **Property** : enablePersistence
 <ejs-tab id="tab" [enablePersistence]="false" >
</ejs-tab> |

| Events | **Property** : events
 <ej-tab id="tab" events="click" > </ej-tab> | Not
Applicable |

| Height | **Property** : height
 <ej-tab id="Tab" height="100%" > </ej-tab> | **Property** : height

 <ejs-tab id="Tab" height="100%" > </ejs-tab> |

| HeightAdjustMode | **Property** : heightAdjustMode
 <ej-tab id="Tab"
heightAdjustMode="Content" > </ej-tab> | **Property** : heightAdjustMode
 <ejs-tab id="Tab"
heightAdjustMode="Content" > </ejs-tab> |

| HtmlAttributes | **Property** : htmlAttributes
 <ej-tab id="Tab" [htmlAttributes]="attributes" >
</ej-tab>
 TS:
 public attributes: Object = {class: "my-class"}; | Not Applicable |

| ID prefix | **Property** : idPrefix
 <ej-tab id="Tab" [idPrefix]="ej-tab-" > </ej-tab> | Not Applicable |

| ShowCloseButton | **Property** : showCloseButton
 <ej-tab id="Tab" [showCloseButton]="true" >
</ej-tab> | **Property** : showCloseButton
 <ejs-tab id="Tab" [showCloseButton]="true" >
</ejs-tab> |

| showReloadIcon | **Property** : showReloadIcon
 <ej-tab id="Tab" [showReloadIcon]="true" >
</ej-tab> | Not Applicable |

| ShowRoundedCorner | **Property** : showRoundedCorner
 <ej-tab id="Tab"
[showRoundedCorner]="true" > </ej-tab> | Not Applicable |

| Destroy | **Method** : destroy()
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
 TabObj.destroy(); | **Method** : destroy()

 <ejs-tab id="tab" #Tab></ejs-tab>
 TS:
@ViewChild('Tab') public TabObj:
TabComponent;
 TabObj.destroy(); |

| Disable Tab | **Method** : disable()
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
 TabObj.disable(); | **Method** :
disable(true)
 <ejs-tab id="Tab" #Tab></ejs-tab>
 TS:
@ViewChild('Tab') public
TabObj: TabComponent;
 TabObj.disable(true); |

| Enable Tab | **Method** : enable()
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
 TabObj.enable(); | **Method** :
disable(false)
 <ejs-tab id="Tab" #Tab></ejs-tab>
 TS:
@ViewChild('Tab') public
TabObj: TabComponent;
 TabObj.disable(false); |

| Show Tab | **Method** : show()
 <ej-tab id="Tab" #Tab></ej-tab>

TS:
@ViewChild('Tab') public TabObj: TabComponent;
 TabObj.show(); | Not
Applicable |

| Hide Tab | **Method** : hide()
 <ej-tab id="Tab" #Tab></ej-tab>
 TS:
@ViewChild('Tab')
public TabObj: TabComponent;
 TabObj.hide(); | Not Applicable |

| Refresh | Not Applicable | **Method** : refresh()
 <ejs-tab id="tab" #Tab></ejs-
tab>
 TS:
@ViewChild('Tab') public TabObj: TabComponent;
 TabObj.refresh(); |

| Created | **Event**: create
 <ej-tab id="tab" #Tab (create)='oncreate(\$event)'></ej-tab>

TS:
 oncreate(event) { } | **Event**: created
 <ejs-tab id="tab" #Tab
(created)='oncreated(\$event)'></ejs-tab>
 TS:
 oncreated(event) { } |

| Destroyed | **Event**: destroy
 <ej-tab id="tab" #Tab (destroy)='ondestroy(\$event)'></ej-
tab>
 TS:
 ondestroy(event) { } | **Event**: destroyed
 <ejs-tab id="tab" #Tab
(destroyed)='ondestroyed(\$event)'></ejs-tab>
 TS:
 ondestroyed(event) { } |

TextArea

Getting started with Angular TextArea Component

This section briefly explains about how to create a simple TextArea component using Angular seed repository.

Dependencies

The following list of dependencies are required to use the TextArea component in your application.

```
`js
|-- @syncfusion/ej2-angular-inputs
|-- @syncfusion/ej2-angular-base
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-base
`
```

Setup angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash
npm install -g @angular/cli
`
```

Create a new application

```
`bash
ng new syncfusion-angular-textarea
`
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash
ng new syncfusion-angular-textarea --style=scss
`
```

Navigate to the created project folder.

```
`bash
cd syncfusion-angular-textarea
`
```

Installing syncfusion TextArea package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-inputs](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-inputs@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-inputs:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding TextArea to the application

- Modify the template in `app.component.ts` file to render the `TextArea` component.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: <textarea id="default"></textarea>
})
export class AppComponent { }
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-inputs/styles/material.css';
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Running the application

After completing the configuration required to render a basic TextArea, run the following command to display the output in your default browser.

```
ng serve
```

The following example illustrates the output in your browser.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    TextAreaModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea id="default"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Getting and setting values

To set the initial value of the TextArea component, you can utilize the [value](#) property. Here's how you can achieve it:

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class='wrap'>
<ejs-textarea id="default" value="comments"></ejs-textarea>
</div>`
})
export class AppComponent { }
```

- Alternatively, you can set the value of the TextArea using ng-model.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `
<div class='wrap'>
<ejs-textarea id="default" [(ngModel)]="value"></ejs-textarea>
</div>
`
})
export class AppComponent {
  value: string = 'Comments'; // Set the initial value here
}
```

- You can dynamically retrieve the value of the TextArea component using the value property bound to ng-model of the TextArea component.

```
`typescript
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `
<div class='wrap'>
<ejs-textarea id="default" [(ngModel)]="value"></ejs-textarea>
<button id="valuebtn" (click)="onButtonClick()">Get Value</button>
</div>
`
})
export class AppComponent {
  value: string = 'Comments'; // Set the initial value here
  onButtonClick() {
    // Get the value of the TextArea
    let textareaValue = this.value;
  }
}
```

- You can retrieve the value of the TextArea by accessing it as an argument from the [change](#) event.

```
`typescript
import { Component } from '@angular/core';
import { ChangedEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<input id='default' (Change)="onChange($event)"><br />
</div>
`
})
export class AppComponent {
  public onChange(args: ChangedEventArgs) {
```

```
// get the value of the TextArea component
```

```
let textareaValue = args.value;
```

```
};
```

```
}
```

```
`
```

Floating Label in Angular TextArea Component

The floating label functionality in the TextArea component allows the placeholder text to float above the TextArea while the user interacts with it, providing a more intuitive user experience. This feature can be achieved using the [floatLabelType](#) API, which offers various options for defining the floating behavior:

Type	Description
Auto	The label floats above the TextArea when it receives focus or input, returning to its initial position when the TextArea loses focus and contains no value.
Always	The label always remains floating above the TextArea, regardless of user interaction.
Never	The label never floats; it remains in its default position within the TextArea.

Type	Description
Auto	The label floats above the TextArea when it receives focus or input, returning to its initial position when the TextArea loses focus and contains no value.
Always	The label always remains floating above the TextArea, regardless of user interaction.
Never	The label never floats; it remains in its default position within the TextArea.

Type	Description
Auto	The label floats above the TextArea when it receives focus or input, returning to its initial position when the TextArea loses focus and contains no value.
Always	The label always remains floating above the TextArea, regardless of user interaction.
Never	The label never floats; it remains in its default position within the TextArea.

Type	Description
Auto	The label floats above the TextArea when it receives focus or input, returning to its initial position when the TextArea loses focus and contains no value.
Always	The label always remains floating above the TextArea, regardless of user interaction.
Never	The label never floats; it remains in its default position within the TextArea.

Type	Description
Auto	The label floats above the TextArea when it receives focus or input, returning to its initial position when the TextArea loses focus and contains no value.
Always	The label always remains floating above the TextArea, regardless of user interaction.
Never	The label never floats; it remains in its default position within the TextArea.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <ejs-textarea id="default" placeholder="Enter your comments"
floatLabelType="Auto"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Placeholder with localization

Localization library allows to localize the placeholder text of the TextArea to different cultures using the `locale` property.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea id='default' floatLabelType="Auto"
placeholder="veuillez inscrire vos commentaires" locale="fr-BE"></ejs-
textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
```

```
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

To load translation object in an application use `load` function of `L10n` class.

In the below sample, `German` culture is loaded to the TextArea placeholder text.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <ejs-textarea id='default' floatLabelType="Auto"
placeholder="veuillez inscrire vos commentaires" locale="fr-BE"></ejs-
textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Rows and Columns in Angular TextArea Component

Two essential attributes, `rows` and `columns`, play a pivotal role in customizing the TextArea's appearance and layout.

The `rows` attribute determines the initial visible number of lines within the TextArea, controlling its vertical size. Conversely, the `columns` attribute specifies the visible width of the TextArea in characters per line, determining its initial width.

- You can customize the TextArea control by setting the number of rows using the [rowCount](#) property and the number of columns using the [columnsCount](#) property. These properties allow precise control over the dimensions of the TextArea, ensuring it fits seamlessly within the layout of the application.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <div>
        <ejs-textarea id="default1" placeholder="Enter your
comments" floatLabelType="Auto" rowCount="3" columnsCount="35"></ejs-
textarea>
      </div>
      <div>
        <ejs-textarea id="default2" placeholder="Enter your
comments" floatLabelType="Auto" rowCount="5" columnsCount="40"></ejs-
textarea>
      </div>
    </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
```

```
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Resize in Angular TextArea Component

The TextArea allows users to input and edit large amounts of text. Resizing this control effectively can enhance the user experience and accommodate varying content needs. This resizing behavior can be enabled and configured using the [resizeMode](#) API, which offers several options for resizing the TextArea:

| Type | Description |

| -- | -- |

| Vertical | Allows users to adjust the height of the TextArea vertically. It is suitable when users want to resize the TextArea only along the vertical axis, accommodating varying amounts of text input. |

| Horizontal | Users can adjust the width of the TextArea horizontally. This option is helpful for accommodating longer lines of text without altering the height of the control. |

| Both | Allows users to adjust both the height and width of the TextArea, offering maximum flexibility in resizing. It is ideal for situations where users need precise control over the dimensions of the TextArea. |

| None | Disables the resizing in the TextArea. This option is ideal for situations where maintaining a consistent layout is critical, and resizing by users is unnecessary. |

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <ejs-textarea id="default" placeholder="Enter your comments"
resizeMode="Both"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
```

```
bootstrap: [AppComponent]
}))
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Width of angular TextArea component

You can easily customize the width of the TextArea using the [width](#) property. This property allows precise adjustment of the TextArea's width according to the specific layout requirements of the application.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <ejs-textarea id="default" placeholder="Enter your comments"
resizeMode="Both" width="500"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
```

```
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Maximum Length in Angular TextArea Component

You can enforce a maximum length limit for the text input in the TextArea using the [maxLength](#) property. This property allows to define the maximum number of characters that users can input into the TextArea.

- By setting the `maxLength` property, you can control the length of text input, preventing users from exceeding a specified character limit.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea id="default" placeholder="Enter your comments"
maxLength="20"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
```

```
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

When the user reaches the specified limit, the TextArea prevents further input, ensuring compliance with the defined character limit. This feature helps maintain data integrity and provides users with clear feedback on the allowed input length.

Form Support in Angular TextArea Component

The TextArea component seamlessly integrates with HTML forms, enabling efficient submission of longer text data. By including TextArea inputs within HTML forms, users can conveniently input multiline text content and submit it as part of form submissions.

This integration enhances the usability of forms, allowing users to provide detailed feedback, enter lengthy descriptions, or input other multiline text data seamlessly.

APP.COMPONENT.TS

```
import { Component, ViewEncapsulation, OnInit, ViewChild } from
'@angular/core';
import { TextAreaComponent } from '@syncfusion/ej2-angular-inputs';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <div className="control_wrapper" id="control_wrapper">
        <h3 className="form-title">Feedback</h3>
        <div className="control_wrapper textarea-form">
          <form id="form-element" method="post">
            <div className="form-group">
              <div className="e-float-input">
                <label>Email</label>
                <input type="email" id="email"
name="email" data-email-message="Please enter valid email address."
data-required-message="Required field"
required data-msg-containerid="emailError"/>
                <span className="e-float-line"></span>
              </div>
              <div id="emailError"></div>
            </div>
            <div className="form-group">
              <div>
                <label>Comments</label>
                <br/>
                <ejs-textarea id="default"
name="comments" data-msg-containerid="commentError" placeholder="Enter your
comments" floatLabelType="Auto" required=""></ejs-textarea>
              </div>
              <div id="commentError"></div>
            </div>
            <div className="row">
              <div style="float: left;">
                <button className="btn"
type="submit">Submit</button>
              </div>
              <div style="float: left;">
```

```

                                <button className="btn"
type="reset">Reset</button>
                                </div>
                            </div>
                        </form>
                    </div>
                <br />
                <br />
            </div>
        </div>
    </div>`
})
export class AppComponent {
    @ViewChild('form-element') element?: any;
    @ViewChild('default') mask?: TextAreaComponent;
    public formObject?: FormValidator;
    ngAfterViewInit() {
        // sets required property in the FormValidator rules collection
        let options: FormValidatorModel = {
            rules: {
                'email': { required: [true, "* Please enter valid email"] },
                'comments': {required: [true, "* Please enter your comments"]}
            },
            //to place the error message in custom position
            customPlacement: (inputElement: HTMLElement, errorElement:
HTMLElement) => {
                (inputElement as HTMLElement |
any).parentNode.parentNode.parentNode.appendChild(errorElement);
            }
        };
        this.formObject = new FormValidator(this.element.nativeElement,
options);
        var proxy = this;
    }
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
    imports: [
        BrowserModule,
        TextAreaModule
    ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Integration of angular TextArea component with FormValidator component

TextArea component seamlessly integrates with the **FormValidator** component, allowing users to incorporate textarea inputs into form validation processes efficiently.

By integrating TextArea components with the **FormValidator** component, users can enforce validation rules specific to text inputs, such as required fields, minimum and maximum length constraints, pattern matching, and more. This ensures that user-submitted text data meets specified criteria and maintains data integrity.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <form id="myForm">
        <span>Please leave your comments</span>
        <br />
        <div id="input-container">
          <ejs-textarea id="default" name="myTextarea"
placeholder='Enter your comments' floatLabelType='Auto' required=""></ejs-
textarea>
        </div>
        <input id="submit" type="submit" value="Submit">
        <input id="reset" type="reset" value="Reset">
      </form>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
```

```

    declarations: [AppComponent],
    bootstrap: [AppComponent]
  })
  export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

Sizing in Angular TextArea Component

you can adjust the size of the TextArea by applying specific classes:

Property	Description
-- --	
Small	Add the <code>e-small</code> class to the input element or its container to render a smaller-sized TextArea.
Bigger	Add the <code>e-bigger</code> class to the input element or its container to render a larger-sized TextArea.

By applying these classes, users can easily customize the appearance of the TextArea to better fit their application's design requirements.

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="textarea">
      <h4> Small Size </h4>
      <div className="e-input-group e-small">
        <ejs-textarea id="default1" className="e-input"
placeholder="Enter your comments" (focus)= "onfocus($event) "
(blur)="onBlur($event) "></ejs-textarea>
      </div>
      <h4> Bigger Size </h4>
      <div className="e-input-group e-bigger">
        <ejs-textarea id="default2" className="e-input"
placeholder="Enter your comments" (focus)= "onfocus($event) "
(blur)="onBlur($event) "></ejs-textarea>
      </div>
    </div>`
})
export class AppComponent {
  public onfocus(args: any): void {
    args.target.parentElement.classList.add("e-input-focus");
  }
}

```



```
public onBlur(args: any): void {
    args.target.parentElement.classList.remove("e-input-focus");
}
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
    imports: [
        BrowserModule,
        TextAreaModule
    ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Filled and outline mode

The Filled and Outline modes can be enabled in the TextArea component by adding the `e-outline` or `e-filled` class to the `cssClass` API.

By adding these classes, users can choose between a filled or outline appearance for the TextArea component, aligning with the design aesthetics of their application.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
    selector: 'app-root',
    template: `<div class="wrap">
        <div class='textarea'>
            <label className="custom-input-label"> Filled TextArea
        </label>
        <div id='container'>
            <ejs-textarea id='filled' placeholder="Filled"
            floatLabelType="Auto" cssClass="e-filled"></ejs-textarea>
        </div>
    `
})
```

```

        <label className="custom-input-label"> Outlined TextArea
    </label>

    <div id='container1'>
        <ejs-textarea id='outlined' placeholder="Outlined"
floatLabelType="Auto" cssClass="e-outline"></ejs-textarea>
    </div>
    </div>
</div>`
))
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Note: Filled and Outline theme customization are available only with Material themes.

Custom styling with cssClass API in TextArea

The `cssClass` Api provides a powerful way to apply custom styling to the `TextArea` component, allowing users to customize its appearance and layout according to their design requirements.

By utilizing the `cssClass` API, users can apply custom CSS classes to the `TextArea` component's container, enabling control over its styling properties such as color, padding, margins, borders, and more.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
```

```

        <ejs-textarea id='default' placeholder="Enter your comments"
floatLabelType="Auto" cssClass="custom-textarea"></ejs-textarea>
        </div>
    </div>`
    })
    export class AppComponent { }

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

Setting the disabled state in TextArea

To disable the TextArea, you can utilize the [enabled](#) property. When set to `false`, the TextArea becomes disabled, preventing user interaction.

```

`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id='default' enabled="false"></ejs-textarea>
</div>`
})

```

```
export class AppComponent { }
`
```

Set the readonly TextArea

To make the TextArea read-only , you can use the [readonly](#) property. When set to `true`, it prevents users from editing the content of the TextArea.

```
`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id='default' readonly="true" value="Readonly"></ejs-textarea>
</div>`
})
export class AppComponent { }
`
```

Set the rounded corner in TextArea

Render the TextArea with `rounded corner` by adding the `e-corner` class to the input parent element.

This rounded corner is visible only in box model input component

```
`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<div class="e-input-group e-corner">
<textarea class="e-input" placeholder="Enter your comments"></textarea>
</div>
</div>`
})
export class AppComponent { }
`
```

Static clear button in TextArea

To display a static clear button in the TextArea component, you can add the `e-static-clear` class to the `cssClass` property. This class ensures that the clear button remains visible at all times, providing users with the ability to easily clear the TextArea content without needing to focus on the control.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea id='default' placeholder="Enter your comments"
cssClass="e-static-clear" showClearButton="true"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Customize the TextArea background color and text color

You can customize the TextArea styles such as background-color, text-color and border-color by overriding its default styles to achieve the desired appearance for the TextArea.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
```

```

    selector: 'app-root',
    template: `<div class="wrap">
        <div class='textarea'>
            <ejs-textarea id='default' placeholder="Enter your comments"
floatLabelType="Auto"></ejs-textarea>
        </div>
    </div>`
  })
  export class AppComponent { }

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

Change the floating label color of the TextArea

You can change the floating label color of the TextArea for both **success** and **warning** validation states by applying the following CSS styles.

```

`css
/ For Success state /
.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
color: #22b24b;
}

```

/ For Warning state /

```
.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
color: #ffca1c;
}
,
```

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <div>
        <ejs-textarea id='default1' placeholder="Success"
floatLabelType="Auto" cssClass="e-success"></ejs-textarea>
      </div>
      <div>
        <ejs-textarea id='default2' placeholder="Warning"
floatLabelType="Auto" cssClass="e-warning"></ejs-textarea>
      </div>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
```

```
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Adding mandatory asterisk to placeholder

To add a mandatory asterisk (*) to the placeholder in the TextArea component, you can utilize CSS to append the asterisk after the placeholder text.

```
`css
/ To add asterick to float label in textarea /
.e-float-input.e-control-wrapper .e-float-text::after {
content: '*/ Add asterisk after the placeholder */
color: red; / Customize asterisk color /
}
```

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea id='default' placeholder="Enter your comments"
floatLabelType="Auto"></ejs-textarea>
    </div>
  </div>`
})
export class AppComponent { }
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS


```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

Events in Angular TextArea Component

This section describes the TextArea events that will be triggered when appropriate actions are performed. The following events are available in the TextArea component.

Created event

The TextArea component triggers the [created](#) event when the TextArea component is created. This event provides users with an opportunity to perform actions immediately after the TextArea has been created and initialized.

`typescript

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (created)="created()"></ejs-textarea>
</div>`
})

export class AppComponent {
  public created(){
    //Your required action here
  };
}
```

Input event

The TextArea component triggers the [input](#) each time when the value of TextArea has changed. This event provides users with an opportunity to perform actions in response to real-time changes in the TextArea's content.

The [InputEventArgs](#) passed as an event argument provides the details about the input event in the TextArea.

`typescript

```
import { Component } from '@angular/core';
import { InputEventArgs } from '@syncfusion/ej2-angular-inputs';
```

```
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (input)="inputHandler($event)"/>
</div>`
})
export class AppComponent {
  public inputHandler(args: InputEventArgs){
    //Your required action here
  };
}
```

Change event

The TextArea component triggers the [change](#) event when the content of TextArea has changed and gets focus-out. This event provides users with an opportunity to execute specific actions in response to changes made by the user.

The [ChangedEventArgs](#) passed as an event argument provides the details about the changes in the TextArea's value.

```
`typescript
import { Component } from '@angular/core';
import { ChangedEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (change)="changeHandler($event)"/>
</div>`
})
export class AppComponent {
  public changeHandler(args: ChangedEventArgs){
    //Your required action here
  };
}
```

`

Focus event

The TextArea component triggers the [focus](#) when the TextArea gains focus. This event allows developers to execute specific actions when the user interacts with the TextArea by focusing on it.

The [FocusInEventArgs](#) passed as an argument provides details about the focus event in the TextArea.

```
`typescript
import { Component } from '@angular/core';
import { FocusInEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (focus)="focusHandler($event)"/>
</div>`
})
export class AppComponent {
  public focusHandler(args: FocusInEventArgs){
    //Your required action here
  };
}
```

`

Blur event

The TextArea component triggers the [blur](#) when the TextArea loses focus. This event allows users to execute specific actions when the user interacts with the TextArea by moving focus away from it.

The [FocusOutEventArgs](#) passed as an argument provides details about the blur event in the TextArea.

```
`typescript
import { Component } from '@angular/core';
import { FocusOutEventArgs } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (blur)="blurHandler($event)"/>
</div>`
})
```

```

})
export class AppComponent {
  public blurHandler(args: FocusOutEventArgs){
    //Your required action here
  };
}
`

```

Destroyed event

The TextArea component triggers the [destroyed](#) when the TextArea component is destroyed.

`typescript

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<!-- To Render TextArea component. -->
<div class="wrap">
<ejs-textarea id="default" (destroyed)="destroyed()"></ejs-textarea>
</div>`
})
export class AppComponent {
  public destroyed(){
    //Your required action here
  };
}
`

```

Methods in Angular TextArea Component

This section outlines the methods available for interacting with the TextArea component.

FocusIn method

The [focusIn](#) method in the TextArea, is used to set focus to the textarea element, enabling user interaction.

By calling the [focusIn](#) method, you can programmatically set focus to the TextArea component, allowing users to interact with it via keyboard input or other means.

APP.COMPONENT.TS

```

import { Component, ViewChild } from '@angular/core';
import { TextAreaComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',

```

```

    template: `<div class="wrap">
        <div class='textarea'>
            <ejs-textarea #default id="default"></ejs-textarea>
            <br/>
            <button id="button">Focus-in</button>
        </div>
    `
  })
  export class AppComponent {
    @ViewChild('default')
    private textareaObj?: TextAreaComponent;
    ngAfterViewInit(): void {
      (document.getElementById('button') as HTMLElement).onclick = () => {
        this.textareaObj?.focusIn();
      }
    }
  }
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

FocusOut method

The [focusOut](#) method in the TextArea component is used to remove focus from the textarea element, ending user interaction.

This method is beneficial for scenarios where user need to programmatically remove focus from the TextArea component, such as after completing a specific task or when navigating to another element in the application.

APP.COMPONENT.TS

```
import { Component, ViewChild } from '@angular/core';
import { TextAreaComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea #default id="default"></ejs-textarea>
      <br/>
      <button id="button">Focus-Out</button>
    </div>
  </div>`
})
export class AppComponent {
  @ViewChild('default')
  private textareaObj?: TextAreaComponent;
  ngAfterViewInit(): void {
    (document.getElementById('button') as HTMLElement).onclick = () => {
      this.textareaObj?.focusOut();
    }
  }
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

GetPersistData method

The [getPersistDataLink to the Video](#) method in the TextArea component retrieves the properties that need to be maintained in the persisted state.

This method returns an object containing the properties to be persisted, which can include various configuration options and state information of the TextArea component.

APP.COMPONENT.TS

```
import { Component, ViewChild } from '@angular/core';
import { TextAreaComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
    <div class='textarea'>
      <ejs-textarea #default id="default"></ejs-textarea>
      <br/>
      <button id="button">Get Persist-data</button>
    </div>`
})
export class AppComponent {
  @ViewChild('default')
  private textareaObj?: TextAreaComponent;
  ngAfterViewInit(): void {
    (document.getElementById('button') as HTMLElement).onclick = () => {
      this.textareaObj?.getPersistData();
    }
  }
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { TextAreaModule } from '@syncfusion/ej2-angular-inputs';
/**
 * Module
 */
@NgModule({
  imports: [
    BrowserModule,
    TextAreaModule
  ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
```

```
enableProdMode();  
platformBrowserDynamic().bootstrapModule(AppModule);
```

TextBox

Getting started with Angular Textbox component

This section briefly explains about how to create a simple TextBox through CSS classes using Angular seed repository.

To get started quickly with Angular TextBox component, you can check out this video:

Dependencies

The following list of dependencies are required to use the TextBox component in your application.

```
`js  
|-- @syncfusion/ej2-angular-inputs  
|-- @syncfusion/ej2-angular-base  
|-- @syncfusion/ej2-inputs  
|-- @syncfusion/ej2-base  
`
```

Setup angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash  
npm install -g @angular/cli  
`
```

Create a new application

```
`bash  
ng new syncfusion-angular-textbox  
`
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash  
ng new syncfusion-angular-textbox --style=scss  
`
```

Navigate to the created project folder.

```
`bash  
cd syncfusion-angular-textbox
```


Installing Syncfusion TextBox Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages($\geq 20.2.36$) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-inputs](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs --save
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-inputs@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-inputs@ngcc --save
```

To mention the `ngcc` package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-inputs:"20.2.38-ngcc"
```

Note: If the `ngcc` tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding TextBox to the application

- Modify the template as HTML input element with `e-input` class in `app.component.ts` file to render the `TextBox` component.

```
`javascript
import { Component } from '@angular/core';
```

```
@Component({
  selector: 'app-root',
  template: <input class="e-input" type="text" placeholder="Enter Name" />
})
export class AppComponent { }
,
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-inputs/styles/material.css';
,
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Adding icons to the TextBox

You can create a TextBox with icon as a group by creating the parent div element with the class `e-input-group` and add the icon element as span with the class `e-input-group-icon`. For detailed information, refer to the [Groups](#) section.

```
`javascript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<div class="wrap">
<div class="e-input-group">
<input class="e-input" name='input' type="text" (focus)="focusIn($event.target)"
placeholder = "Enter Date" (blur)="focusOut($event.target)"/>
<span class="e-input-group-icon e-input-popup-date" (mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
</div>
</div>`
})
export class AppComponent {
```

```

public focusIn(target: HTMLElement): void {
target.parentElement.classList.add('e-input-focus');
}

public focusOut(target: HTMLElement): void {
target.parentElement.classList.remove('e-input-focus');
}

public onMouseDown(target: HTMLElement): void {
target.classList.add('e-input-btn-ripple');
}

public onMouseUp(target: HTMLElement): void {
let ele: HTMLElement = target;
setTimeout(
() => {ele.classList.remove('e-input-btn-ripple'); },
500);
}
}
`

```

Running the application

After completing the configuration required to render a basic TextBox, run the following command to display the output in your default browser.

```

ng serve
`

```

The following example illustrates the output in your browser.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component } from '@angular/core';
@Component({
imports: [

],
standalone: true,
selector: 'app-root',
template: `<div class="wrap">
<div class="e-input-group">
<input class="e-input" name='input' type="text"
(focus)="focusIn($event.target)"
placeholder = "Enter Date" (blur)="focusOut($event.target)"/>
`

```

```

        <span class="e-input-group-icon e-input-popup-date"
(mouseup)="onMouseUp($event.target) "
(mousedown)="onMouseDown($event.target) "></span>
        </div>
    </div>`
    })
    export class AppComponent {
        public focusIn(target: HTMLElement | any): void {
            target.parentElement.classList.add('e-input-focus');
        }
        public focusOut(target: HTMLElement | any): void {
            target.parentElement.classList.remove('e-input-focus');
        }
        public onMouseDown(target: HTMLElement | any): void {
            target.classList.add('e-input-btn-ripple');
        }
        public onMouseUp(target: HTMLElement | any): void {
            let ele: HTMLElement = target;
            setTimeout(
                () => {ele.classList.remove('e-input-btn-ripple'); },
                500);
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Floating label

The floating label TextBox floats the label above the TextBox after focusing, or filled with value in the TextBox.

You can create the floating label TextBox by using the [floatLabelType](#) API.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
    imports: [
        TextBoxModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="wrap">
    <h4>Floating label as auto</h4>
    <div class="textboxes">
        <ejs-textbox placeholder="First Name" floatLabelType="Auto"></ejs-
textbox>
    </div>
`
})

```

```
<h4>Floating label as always</h4>
<div class='textboxes'>
  <ejs-textbox placeholder="First Name" floatLabelType="Always"></ejs-
textbox>
</div>
</div>`
}))
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [How to render TextBox programmatically](#)
- [How to add floating label to TextBox programmatically](#)

Groups in Angular Textbox component

The following section explains you the steps required to create TextBox with **icon** and **floating label**.

TextBox:

- Create a parent div element with the class **e-input-group**
- Place input element with the class **e-input** inside the parent div element.

`typescript

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  template: `<div class="wrap">
<div class="e-float-input e-input-group">
<input class="e-input" type="text" placeholder="Enter Name" />
</div>
</div>`
})
export class AppComponent { }
```

Floating label:

- Add the `e-float-input` class to the parent div element.
- Remove the `e-input` class and add `required` attribute to the input element.
- Place the span element with class `e-float-line` after the input element.
- Place the label element with class `e-float-text` after the above created span element.

When you focus or filled with value in the TextBox, the label floats above the TextBox.

Creating the Floating label TextBox, you have to set the `required` attribute to the Input element to achieve the floating label functionality which is used for validating the value existence in TextBox.

If you want to render the Floating label TextBox without `required` attribute then refer to the [Floating Label without required attribute](#) section.

```
`typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<div class="wrap">
<div class="e-float-input e-input-group">
<input type="text" required/>
<span class="e-float-line"></span>
<label class="e-float-text">Enter Name </label>
</div>
</div>`
})
export class AppComponent { }
`
```

And refer to the following sections to add the icons to the TextBox.

With icon and floating label

Create an icon element as a span with the class `e-input-group-icon`, and the user can place the icon in either side of TextBox by adding the created icon element before/after the input.

For the floating label enabled TextBox add the icon element as first or last element inside the TextBox wrapper, and based on the element position it will act as prefix or suffix icon.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule,
```

```

    ],
    standalone: true,
    selector: 'app-root',
    template: `<div class="wrap">
        <h4> TextBox with icons </h4>
        <div class="e-input-group">
            <input class="e-input" type="text"
(focus)="focusIn($event.target)"
            placeholder = "Enter Date"
(blur)="focusOut($event.target)" />
            <span class="e-input-group-icon e-input-popup-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
        </div>
        <div class="e-input-group e-float-icon-left">
            <span class="e-input-group-icon e-input-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
            <div class="e-input-in-wrap">
                <input class="e-input" type="text"
(focus)="focusIn($event.target)"
                placeholder = "Enter Date"
(blur)="focusOut($event.target)" />
            </div>
        </div>
        <div class="e-input-group e-float-icon-left">
            <span class="e-input-group-icon e-input-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
            <div class="e-input-in-wrap">
                <input class="e-input" type="text"
(focus)="focusIn($event.target)"
                placeholder = "Enter Date"
(blur)="focusOut($event.target)" />
            <span class="e-input-group-icon e-input-down"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
        </div>
        <h4> Floating label with icons </h4>
        <div class="e-float-input e-input-group">
            <input type="text" required
(focus)="focusIn($event.target)" (blur)="focusOut($event.target)">
            <span class="e-float-line"></span>
            <label class="e-float-text" >Enter Date</label>
            <span class="e-input-group-icon e-input-popup-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
        </div>
        <div class="e-float-input e-input-group e-float-icon-left">
            <span class="e-input-group-icon e-input-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
            <div class="e-input-in-wrap">
                <input type="text" required
(focus)="focusIn($event.target)" (blur)="focusOut($event.target)">
            <span class="e-float-line"></span>

```

```

        <label class="e-float-text" >Enter Date</label>
      </div>
    </div>
    <div class="e-float-input e-input-group e-float-icon-left">
      <span class="e-input-group-icon e-input-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
      <div class="e-input-in-wrap">
        <input type="text" required
(focus)="focusIn($event.target)" (blur)="focusOut($event.target)">
        <span class="e-float-line"></span>
        <label class="e-float-text" >Enter Date</label>
        <span class="e-input-group-icon e-input-down"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
      </div>
    </div>
  </div>`
  })
  export class AppComponent {
    public focusIn(target: any): void {
      let parent: HTMLElement = target.parentElement as HTMLElement;
      if (parent.classList.contains('e-input-in-wrap')) {
        (parent.parentElement as HTMLElement).classList.add('e-input-
focus');
      } else {
        parent.classList.add('e-input-focus');
      }
    }
    public focusOut(target: any): void {
      let parent: HTMLElement = target.parentElement as HTMLElement;
      if (parent.classList.contains('e-input-in-wrap')) {
        (parent.parentElement as HTMLElement).classList.remove('e-input-
focus');
      } else {
        parent.classList.remove('e-input-focus');
      }
    }
    public onMouseDown(target: any): void {
      target.classList.add('e-input-btn-ripple');
    }
    public onMouseUp(target: any): void {
      let ele: HTMLElement = target;
      setTimeout(
        () => {ele.classList.remove('e-input-btn-ripple');
        }, 500);
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```


To place the icon on left side of the TextBox, create a div element with the class `e-input-in-wrap` as wrapper to the input element and place the `floating line`, `floating text`, and right side icon element within it.

Add the `e-float-icon-left` class to the TextBox container element.

With clear button and floating label

The clear button is added to the input for clearing the value given in the TextBox.

It is shown only when the input field has a value, otherwise not shown.

You can add the clear button to the TextBox by `showClearButton` API in textbox

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    TextBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
<h4>Textbox with clear icon</h4>
<div class='textboxes'>
  <ejs-textbox placeholder="First Name" showClearButton='true'
floatLabelType="Never"></ejs-textbox>
</div>
<h4>Floating Textbox with clear icon</h4>
<div class='textboxes'>
  <ejs-textbox placeholder="Last Name" showClearButton='true'
floatLabelType="Auto"></ejs-textbox>
</div>
</div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Floating Label without required attribute

You can render the `Floating label TextBox` without `required` attribute by manually float the label above of the TextBox using `ngClass`.

You can manually float the label above of the TextBox by adding the below list of classes to the floating label element. The classes are:

Class Name | Description

e-label-top | Floats the label above of the TextBox.

e-label-bottom | Label to be placed as placeholder for the TextBox.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <h4> Floating Label without required attribute </h4>
    <div class="e-float-input e-input-group" [ngClass]="{'e-
input-focus': tb1Focused}>
      <input type="text" id="textBox1"
[ (ngModel) ]="textBoxValue"
      #testTextBox1 (focus)="tb1Focused = true"
(blur)="tb1Focused = false"/>
      <span class="e-float-line"></span>
      <label class="e-float-text" [ngClass]="{'e-label-top':
textBoxValue !== '' , 'e-label-bottom': textBoxValue == ''}" >Enter
Name</label>
    </div>
  </div>`
})
export class AppComponent {
  public textBoxValue?: string;
  tb1Focused?: any;
  ngOnInit() {
    this.textBoxValue = '';
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Multi-line Input with Floating Label

Add the HTML textarea element with the `e-input` class to create default multi-line input.

Add the floating label support to the `multi-line input` by creating the floating label structure as defined in the initial section.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [

    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <textarea class="e-input" placeholder="Address"></textarea>
    <div class="e-float-input">
      <textarea required></textarea>
      <span class="e-float-line"></span>
      <label class="e-float-text"> Address </label>
    </div>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [How to add floating label to TextBox programmatically](#)
- [Change the floating label color of the TextBox](#)
- [Change the color of the TextBox based on its value](#)

Sizing in Angular Textbox component

You can render the TextBox in two different sizes.

Property | Description

Normal | By default, the TextBox is rendered with normal size.

Small | You need to add `e-small` class to the input element, or else add to the input container.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [

    FormsModule
  ],
```

```

standalone: true,
selector: 'app-root',
template: `<div class="wrap">
  <h4> Normal Size </h4>
  <input class="e-input" type="text" placeholder="Enter Name"/>
  <div class="e-float-input">
    <input type='text' required />
    <span class="e-float-line"></span>
    <label class="e-float-text">Enter Name</label>
  </div>
  <div class="e-input-group">
    <input class="e-input" type="text" (focus)="focusIn($event.target)"
      placeholder = "Enter Date" (blur)="focusOut($event.target)" />
    <span class="e-input-group-icon e-input-popup-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
  </div>
  <h4> Small Size </h4>
  <input class="e-input e-small" type="text" placeholder="Enter Name"/>
  <div class="e-float-input e-small">
    <input type='text' required />
    <span class="e-float-line"></span>
    <label class="e-float-text">Enter Name</label>
  </div>
  <div class="e-input-group e-small">
    <input class="e-input" type="text" (focus)="focusIn($event.target)"
      placeholder = "Enter Date" (blur)="focusOut($event.target)" />
    <span class="e-input-group-icon e-input-popup-date"
(mouseup)="onMouseUp($event.target)"
(mousedown)="onMouseDown($event.target)"></span>
  </div>
</div>`
))
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
  public onMouseDown(target: any): void {
    target.classList.add('e-input-btn-ripple');
  }
  public onMouseUp(target: any): void {
    let ele: HTMLElement = target;
    setTimeout(
      () => {ele.classList.remove('e-input-btn-ripple');
    }, 500);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';

```

```
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Validation in Angular Textbox component

The TextBox supports three types of validation styles namely **error**, **warning**, and **success**. These states are enabled by adding corresponding classes **.e-error**, **.e-warning**, or **.e-success** to the input parent element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <input class="e-input e-warning" type="text" placeholder="Input with
warning"/>
    <div class="e-input-group e-error">
      <input class="e-input" type="text"
(focus)="focusIn($event.target)"
placeholder = "Input with error"
(blur)="focusOut($event.target)"/>
    </div>
    <div class="e-input-group e-success">
      <input class="e-input" type="text"
(focus)="focusIn($event.target)"
placeholder = "Input with success"
(blur)="focusOut($event.target)"/>
    </div>
  </div>`
})
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-float-input.e-control-wrapper .e-float-text::after` class.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component } from '@angular/core';
@Component({
  imports: [
    TextBoxModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <h4>Floating label as auto</h4>
    <div class="textboxes">
      <ejs-textbox placeholder="First Name" floatLabelType="Auto"></ejs-
textbox>
    </div>
    <h4>Floating label as always</h4>
    <div class="textboxes">
      <ejs-textbox placeholder="First Name" floatLabelType="Always"></ejs-
textbox>
    </div>
  </div>`
})
export class AppComponent { }
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Multiline in Angular Textbox component

This feature allows the textbox to accept one or more lines of text like address, description, comments, and more.

Create multiline textbox

You can convert the default textbox into the multiline textbox by setting the [multiline](#) API value as true or pass HTML5 textarea as element to the textbox.

The multiline textbox allows you to resize it in vertical direction alone.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
```

```
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    FormsModule, TextBoxModule
  ],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./index.css'],
  template: `<div class="multiline">
    <ejs-textbox [multiline]='true' value= 'Mr. Dodsworth
Dodsworth, System Analyst, Studio 103, The Business Center' ></ejs-textbox>
  </div>
`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Implementing floating label

You can achieve the floating label behavior in the multiline textbox by setting [floatLabelType](#) as 'Auto'. The placeholder text act as floating label to the multiline textbox. You can provide the placeholder text to the multiline textbox either by using the [placeholder](#) property or placeholder attribute.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    FormsModule, TextBoxModule
  ],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./index.css'],
  template: `<label class="label">float label type auto</label>
    <div class="multiline">
      <ejs-textbox [multiline]='true' floatLabelType='Auto'
placeholder='Enter your address' ></ejs-textbox>
    </div>
    <label class="label">float label type always</label>
    <div class="multiline">
      <ejs-textbox [multiline]='true' floatLabelType='Always'
placeholder='Enter your address' ></ejs-textbox>
    </div>
`
})
```

```

        <label class="label">float label type never</label>
        <div class="multiline">
            <ejs-textbox [multiline]='true' floatLabelType='Never'
placeholder='Enter your address' ></ejs-textbox>
        </div>
    })
    export class AppComponent {
        constructor() {
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Auto resizing

By default, you can manually resize the multiline textbox. But you can also create an auto resizing multiline textbox with both the initial and dynamic value change. It can be done by calculating the height of the textarea in the created event for initial value update and in the input event for dynamic value update of the auto resize multiline textbox, as explained in the following code sample.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
    imports: [
        FormsModule, TextBoxModule
    ],
    standalone: true,
    selector: 'app-container',
    styleUrls: ['./index.css'],
    template: `<div class="multiline">
        <ejs-textbox #default [multiline]='true' value= 'Mr. Dodsworth
Dodsworth, System Analyst, Studio 103, The Business Center'
        floatLabelType='Auto' placeholder='Enter your address'
        (created)="createHandler($event)" (input)="inputHandler($event)" ></ejs-
textbox>
    </div>
    `
})
export class AppComponent {
    @ViewChild('default')
    public textareaObj?: TextBoxComponent;
    public createHandler(args: any): void {
        (this.textareaObj as TextBoxComponent).addAttributes({rows: 1} as
any);
    }
}

```



```
(this.textareaObj as TextBoxComponent).element.style.height = "auto";
(this.textareaObj as TextBoxComponent).element.style.height =
((this.textareaObj as TextBoxComponent).element.scrollHeight)+"px";
}
public inputHandler(args: any): void {
(this.textareaObj as TextBoxComponent).element.style.height = "auto";
(this.textareaObj as TextBoxComponent).element.style.height =
((this.textareaObj as TextBoxComponent).element.scrollHeight)+"px";
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Disable resizing

By default, the multiline textbox is rendered with resizable. You can disable the resize of the multiline textbox by applying the following CSS styles.

`CSS

```
textarea.e-input,
.e-float-input textarea,
.e-float-input.e-control-wrapper textarea,
.e-input-group textarea,
.e-input-group.e-control-wrapper textarea {
resize: none;
}
`
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
@Component({
imports: [
FormsModule, TextBoxModule
],
standalone: true,
selector: 'app-container',
styleUrls: ['./index.css'],
template: `<div class="multiline">
<ejs-textbox id='default' [multiline]='true'
floatLabelType='Auto' placeholder='Enter your address' ></ejs-textbox>
</div>`
})
```

```

    })
    export class AppComponent {
        constructor() {
        }
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Limit the text length

By default, the text length of the multiline textbox is unlimited. You can limit the text length by setting the `maxLength` attribute using the [addAttributes](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
    imports: [
        FormsModule, TextBoxModule, ButtonModule
    ],
    standalone: true,
    selector: 'app-container',
    styleUrls: ['./index.css'],
    template: `<label class="label">Add maxlength attribute through
inline</label>
        <div class="multiline">
            <ejs-textbox [multiline]='true' maxlength='15'
floatLabelType='Auto' placeholder='Enter your address' ></ejs-textbox>
        </div>
        <label class="label">Add maxlength attribute through
addAttributes method</label>
        <div class="multiline">
            <ejs-textbox #default [multiline]='true'
floatLabelType='Auto' placeholder='Enter your address' ></ejs-textbox>
        </div>
        <button ej-button id=length
(click)='clickHandler($event)'>Add max length</button>
    `
})
export class AppComponent {
    @ViewChild('default')
    public textAreaObj?: TextBoxComponent;
    public clickHandler(args: any) {

```

```
(this.textareaObj as TextBoxComponent).addAttributes({maxlength: 15}
as any);
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Count characters

You can show the number of characters entered inside the textarea by calculating the character count in the input event of multiline textbox. The character count is updated while entering or deleting any character inside the textarea. The character count shows how many characters can be entered or left to be entered.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { Component, ViewChild } from '@angular/core';
import { TextBoxComponent } from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [
    FormsModule, TextBoxModule
  ],
  standalone: true,
  selector: 'app-container',
  styleUrls: ['./index.css'],
  template: `<div class="multiline">
    <ejs-textbox #default [multiline]='true' maxlength='25'
floatLabelType='Auto' placeholder='Enter your address'
(input)='inputHandler($event)' ></ejs-textbox>
    <span id='numbercount'></span>
  </div>`
})
export class AppComponent {
  @ViewChild('default')
  public textareaObj?: TextBoxComponent;
  public inputHandler(args: any): void {
    let word: any, addresscountRem, addressCount: string;
    word = this.textareaObj?.element.value;
    addressCount = word.length;
    addresscountRem = document.getElementById('numbercount');
    (addresscountRem as HTMLElement).textContent = addressCount+"/25";
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

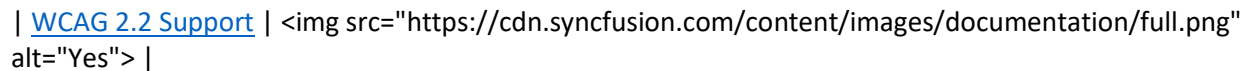
Accessibility in Angular Textbox component

The Textbox 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 Textbox component is outlined below.

| Accessibility Criteria | Compatibility |

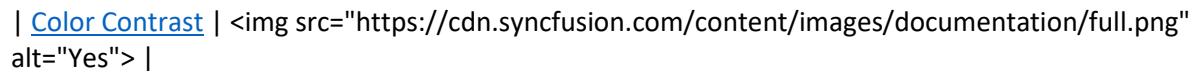
| -- | -- |

| [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 Textbox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The Textbox uses the `textbox` role and following ARIA properties for its element based on its state.

| **Property** | **Functionality** |

| --- | --- |

| `aria-placeholder` | The `aria-placeholder` is a short hint to help the users with data entry when the Textbox has no value. |

| `aria-labelledby` | The `aria-labelledby` property indicates the floating label element of the Textbox. |

Ensuring accessibility

The Textbox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Textbox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Textbox component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style appearance in Angular Textbox component

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 TextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
```

```
/ To specify height and font size /
```

```
.e-input:not(:valid), .e-input:valid, .e-float-input.e-control-wrapper input:not(:valid), .e-float-input.e-control-wrapper input:valid, .e-float-input input:not(:valid), .e-float-input input:valid, .e-input-group input:not(:valid), .e-input-group input:valid, .e-input-group.e-control-wrapper input:not(:valid), .e-input-group.e-control-wrapper input:valid, .e-float-input.e-control-wrapper textarea:not(:valid), .e-float-input.e-control-wrapper textarea:valid, .e-float-input textarea:not(:valid), .e-float-input textarea:valid, .e-input-group.e-control-wrapper textarea:not(:valid), .e-input-group.e-control-wrapper textarea:valid, .e-input-group textarea:not(:valid), .e-input-group textarea:valid {
```

```
font-size: 30px;
```

```
height: 40px;
```

```
}
```

```
,
```

Customizing the TextBox placeholder

Use the following CSS to customize the TextBox placeholder

```
`css
```

```
/ To specify font size and color /
```

```
.e-float-input.e-control-wrapper:not(.e-error) input:valid ~ label.e-float-text, .e-float-input.e-control-wrapper:not(.e-error) input ~ label.e-label-top.e-float-text {
```


```
color: pink;
```

```
font-size: 15px;
```

```
}
```

```
,
```

Toggle password visibility using eye icon

You can show text or hide text by showing  character instead of actual text in angular textbox by following below steps.

- Add eye icon using [addIcon](#) method.
- In the click event of icon added above, toggle the text visibility by changing the **type** of element.

APP.COMPONENT.TS

```
import { TextBoxModule } from '@syncfusion/ej2-angular-inputs'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core'
import { Component, Inject, ViewChild } from '@angular/core';
import {
  TextBoxComponent,
  NumericTextBoxComponent,
} from '@syncfusion/ej2-angular-inputs';
@Component({
  imports: [ FormsModule, TextBoxModule],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./app.component.css'],
  templateUrl: './app.component.html',
})
export class AppComponent {
  @ViewChild('default', { static: true })
  public textbox?: TextBoxComponent;
  ngAfterViewInit() {
    (this.textbox as TextBoxComponent).addIcon('append', 'e-icons e-eye');
    document
      .getElementsByClassName('e-input-eye')[0]
      .addEventListener('click', function (e) {
        let textObj: any = (document.getElementById('password') as any)
          .ej2_instances[0];
        if (textObj.element.type === 'password') {
          textObj.element.type = 'text';
        } else {
          textObj.element.type = 'password';
        }
      });
  }
}
```

```
    }
  });
}
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

How To

Set the rounded corner in Angular Textbox component

Render the TextBox with rounded corner by adding the e-corner class to the input parent element.

This rounded corner visible only in box model input component

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { Component } from '@angular/core';
@Component({
  imports: [
    ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="e-input-group e-corner">
      <input class="e-input" type="text"
(focus)="focusIn($event.target)"
placeholder = "Enter Date"
(blur)="focusOut($event.target)"/>
      <span class="e-input-group-icon e-input-popup-date"
></span>
    </div>
    <div class="e-float-input e-input-group e-corner">
      <input type="text" required />
      <span class="e-float-line"></span>
      <label class="e-float-text">Enter Date</label>
      <span class="e-input-group-icon e-input-popup-date"
></span>
    </div>
  </div>`,
  styleUrls: ['./index.css']
})
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
}
```

```
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set the disabled state in Angular Textbox component

Disable the TextBox by adding the `e-disabled` to the input parent element and set `disabled` attribute to the input element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <input class="e-input" type="text" placeholder="Enter Name"
disabled />
    <div class="e-float-input e-disabled">
      <input type='text' required disabled/>
      <span class="e-float-line"></span>
      <label class="e-float-text">Enter Name</label>
    </div>
  </div>`
})
export class AppComponent {}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Set the read only textbox in Angular Textbox component

You can make the TextBox as `read-only` by setting the `readonly` attribute to the input element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
```



```
@Component({
  imports: [

    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <input class="e-input" type="text" value="John"
placeholder="Enter Name" readonly />
    <div class="e-float-input">
      <input type="text" value="John" required readonly/>
      <span class="e-float-line"></span>
      <label class="e-float-text e-label-top">Enter
Name</label>
    </div>
  </div>`
})
export class AppComponent {}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add textbox programmatically in Angular Textbox component

- Import the **Input** modules from **ej2-inputs** library as shown below.

`typescript

```
import {Input} from '@syncfusion/ej2-inputs';
```

,

- Pass the **HTML Input** element as parameter to the **createInput** method.
- You can also add the icons on the input by passing **buttons** property value with the class name **e-input-group-icon** as parameter to the **createInput** method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
import { Input } from '@syncfusion/ej2-inputs';
@Component({
  imports: [

    FormsModule
  ],
  standalone: true,
```

```

    selector: 'app-root',
    template: `<div class="wrap">
        <input id="textbox" type="text" placeholder="Enter Name" />
        <input id="textbox-icon" type="text" />
    </div>`
  })
  export class AppComponent {
    ngOnInit() {
      let element: HTMLInputElement = document.getElementById('textbox') as
HTMLInputElement;
      Input.createInput ({
        element: element
      });
      let element1: HTMLInputElement = document.getElementById('textbox-
icon') as HTMLInputElement;
      Input.createInput ({
        element: element1,
        buttons: ['e-input-group-icon e-input-down'],
        properties: {
          placeholder: 'Enter Value'
        }
      });
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add floating label to textbox programmatically in Angular Textbox component

The **Floating Label TextBox** floats label above the TextBox after focusing, or entering a value in the TextBox.

Floating label supports the types of actions as given below.

Type	Description
Auto	The floating label will float above the input after focusing, or entering a value in the input.
Always	The floating label will always float above the input.
Never	By default, never float the label in the input when the placeholder is available.

- Import the **Input** modules from **ej2-inputs** library as shown in below.

`typescript

```
import {Input} from '@syncfusion/ej2-inputs';
```

,

- Pass the **HTML Input** element and **floatLabelType** property as **Auto** to the **createInput** method.
- Set the **placeholder** value to the input element via **element attribute** or pass the parameter to the **createInput** method.

The **watermark label** will be updated based on the specified **placeholder** value of the **Floating Label TextBox**.

- You can add the **icons** on the input by passing **buttons** property value with the class name **e-input-group-icon** as parameter to the **createInput** method.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
import { Input } from '@syncfusion/ej2-inputs';
@Component({
  imports: [

    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <h4> FloatLabelType as Auto </h4>
    <input type="text" id="textbox" required/>
    <h4> FloatLabelType as Always </h4>
    <input type="text" id="textbox-01" required/>
    <h4> FloatLabelType as Never </h4>
    <input type="text" id="textbox-02" required/>
    <h4> Float label input with icons </h4>
    <input id="textbox-icon" type="text" />
  </div>`
})
export class AppComponent {
  ngOnInit() {
    let element: HTMLInputElement = document.getElementById('textbox') as
HTMLInputElement;
    Input.createInput ({
      element: element,
      floatLabelType: "Auto",
      properties: {
        placeholder: 'Enter Name'
      }
    });
    let element1: HTMLInputElement = document.getElementById('textbox-
01') as HTMLInputElement;
    Input.createInput ({
      element: element1,
      floatLabelType: "Always",
      properties: {
        placeholder: 'Enter Name'
      }
    });
  }
}
```

```

    });
    let element2: HTMLInputElement = document.getElementById('textbox-
02') as HTMLInputElement;
    Input.createInput ({
        element: element2,
        floatLabelType: "Never",
        properties: {
            placeholder: 'Enter Name '
        }
    });
    let element3: HTMLInputElement = document.getElementById('textbox-
icon') as HTMLInputElement;
    Input.createInput ({
        element: element3,
        floatLabelType: "Auto",
        buttons: ['e-input-group-icon e-input-down'],
        properties: {
            placeholder: 'Enter Value '
        }
    });
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Change the floating label color of the textbox in Angular Textbox component

You can change the floating label color of the TextBox for both **success** and **warning** validation states by applying the following CSS styles.

`CSS

/ For Success state /

```

.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
color: #22b24b;
}

```

/ For Warning state /

```

.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
color: #ffc107;
}

```

```
}
,
```

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="e-float-input e-input-group e-success">
      <input type="text" (focus)="focusIn($event.target)"
(blur)="focusOut($event.target)" required/>
      <span class="e-float-line"></span>
      <label class="e-float-text">Success</label>
    </div>
    <div class="e-float-input e-input-group e-warning">
      <input type="text" (focus)="focusIn($event.target)"
(blur)="focusOut($event.target)" required/>
      <span class="e-float-line"></span>
      <label class="e-float-text">Warning</label>
    </div></div>`
})
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Change the color of the textbox based on its value in Angular Textbox component

You can change the color of the TextBox by validating its value using regular expression in the **keyup** event for predicting the numeric values as demonstrated in the following code sample.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule } from '@angular/forms'
```

```
import { Component } from '@angular/core';
@Component({
  imports: [
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <label> Normal Input </label>
    <div class="e-input-group">
      <input class="e-input" type="text"
        (focus)="focusIn($event.target)"
        placeholder = "Enter numeric values"
        (blur)="focusOut($event.target)" (keyup)="onKeyup($event)" />
    </div>
    <label> Floating Input </label>
    <div class="e-float-input">
      <input type="text" (focus)="focusIn($event.target)"
        (blur)="focusOut($event.target)" (keyup)="onKeyup($event)" required/>
      <span class="e-float-line"></span>
      <label class="e-float-text">Enter numeric values</label>
    </div>
  </div>`
})
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
  public onKeyup(event: any): void {
    let str = event.target.value.match(/^[0-9]+$/);
    if (!((str && str.length > 0)) && event.target.value.length) {
      event.target.parentElement.classList.add('e-error');
    } else {
      event.target.parentElement.classList.remove('e-error');
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Add floating label to read only textbox in Angular TextBox component

You can achieve floating label for read-only textboxes by adding/removing `e-label-top` and `e-label-bottom` classes to the label element

In this sample, click the update value button to fill the read-only textbox with value and float a label.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component } from '@angular/core';
@Component({
  imports: [

    ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <div class="row">
      <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
        <div class="e-float-input">
          <input class="e-input myField" id="myText"
name="readonlyAttr" type="text" readOnly>
          <span class="e-float-line"></span>
          <label class="e-float-text">Enter value</label>
        </div>
      </div>
    </div>
    <div class="row">
      <div class="col-xs-10 col-sm-10 col-lg-10 col-md-10">
        <button ejs-button class="e-btn update_value"
id='valuebtn' >Set value</button>
        <button ejs-button class="e-btn remove_value"
id='removebtn' >Remove value</button>
      </div>
    </div>
  </div>`
})
export class AppComponent {
  ngAfterViewInit(): void {
    (document.getElementById('valuebtn') as HTMLElement).onclick = () => {
      ((document.getElementsByClassName('myField')[0] as any).value = '10';
      checkFloatingLabel('myText'))
    }
    (document.getElementById('removebtn') as
HTMLElement).addEventListener('click', function() {
      ((document.getElementsByClassName('myField')[0] as any).value = '';
      checkFloatingLabel('myText'))
    })
  }
}
function checkFloatingLabel(id: any): void {
  let inputElement: HTMLElement | any = document.getElementById(id) as
HTMLElement;
  let labelElement: Element = (inputElement.parentElement as
HTMLElement).querySelector('.e-float-text') as Element;
  if (inputElement.value !== '') {
    labelElement.classList.remove('e-label-bottom');
    labelElement.classList.add('e-label-top');
  } else {
    labelElement.classList.remove('e-label-top');
    labelElement.classList.add('e-label-bottom');
  }
}
```

```
let inputField = document.getElementById('myText');
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Customize the textbox background color and text color in Angular Textbox component

You can customize the textbox styles such as background-color, text-color and border-color by overriding its default styles.

To change the styles of the **floating label**, you must override the style to the input element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { Component } from '@angular/core';
@Component({
  imports: [
    ],
  standalone: true,
  selector: 'app-root',
  template: `<div class="wrap">
    <label> Normal Input </label>
    <div class="e-input-group">
      <input (focus)="focusIn($event.target)"
(blur)="focusOut($event.target)" class="e-input" type="text"
placeholder="First Name">
    </div>
    <label> Floating Input </label>
    <div class="e-float-input">
      <input (focus)="focusIn($event.target)"
(blur)="focusOut($event.target)" type="text" required>
      <span class="e-float-line"></span>
      <label class="e-float-text">Last Name</label>
    </div>
  </div>`
})
export class AppComponent {
  public focusIn(target: any): void {
    target.parentElement.classList.add('e-input-focus');
  }
  public focusOut(target: any): void {
    target.parentElement.classList.remove('e-input-focus');
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
```



```
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Migration from css textbox to angular textbox in Angular Textbox component

From v16.3.21 version, the textbox is provided as Angular component to achieve the floating label textbox with minimal code. You can find the available textbox properties, methods, and events in the [API reference](#).

The following table describes the migration from CSS textbox to Angular textbox component.

Normal textbox

<!-- markdownlint-disable MD038 -->

Rendering mode	CSS textbox	Angular textbox component
Default rendering	<code><div class='e-input-group'>
<input class='e-input' type='text' placeholder='Enter Value' />
</div></code>	<code><ejs-textbox id="default" placeholder="Enter Value" floatLabelType="Never"></ejs-textbox></code>
Textbox with clear button	<code><div class='e-input-group'>
<input class='e-input' placeholder='Enter Value' required='true'>

</div>

</code> Note: You have to write action for clear button.	<code><ejs-textbox id="clear-input" placeholder="Enter Value" floatLabelType="Never" [showClearButton]="true"></ejs-textbox></code>
Validation states	<code><div class='e-input-group e-error'>
<input class='e-input' type='text' placeholder='Enter Value' />
</div>

</code> Note: Textbox component consists of three types of validation rules such as success, warning, and error.	<code><ejs-textbox id="validation" placeholder="Enter Value" floatLabelType="Never" cssClass="e-error"></ejs-textbox></code>

Floating label textbox

<!-- markdownlint-disable MD038 -->

Rendering mode	CSS textbox	Angular textbox component
Default rendering	<code><div class='e-float-input'>
<input type='text' required />

<label class='e-float-text'>Enter Value</label>
</div></code>	<code><ejs-textbox id="default" placeholder="Enter Value" floatLabelType="Auto"></ejs-textbox></code>
Textbox with clear button	<code><div class='e-float-input e-input-group'>
<input type='text' required value='Clear Input' />

<label class='e-float-text'>Enter Value</label>

</div>

</code> Note: You have to write action for clear button.	<code><ejs-textbox id="clear-input" placeholder="Enter Value" floatLabelType="Auto" [showClearButton]="true"></ejs-textbox></code>

```
| Validation states | <div class='e-float-input e-error'><br/><input type='text' required
/><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter
Value</label><br/></div><br/><br/> Note: Textbox component consists of three types of validation
rules such as success, warning, and error. | <ejs-textbox id="validation" placeholder="Enter Value"
floatLabelType="Auto" cssClass="e-error"></ejs-textbox> |
```

Timeline

Getting started with Angular Timeline component

This section explains how to create a simple Timeline, and demonstrate the basic usage of the Timeline module in an Angular environment.

Dependencies

The list of dependencies required to use the Timeline module in your application is given below:

```
`javascript
|-- @syncfusion/ej2-angular-layouts
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-layouts
|-- @syncfusion/ej2-angular-base
`,`
```

Setup Angular environment

You can use [Angular CLI](#) to setup your Angular applications. To install Angular CLI use the following command.

```
`
npm install -g @angular/cli
`,`
```

Create an Angular application

Start a new Angular application using below Angular CLI command.

```
`
ng new my-app
cd my-app
`,`
```

Installing Syncfusion Timeline Package

Syncfusion packages are distributed in npm as [@syncfusion](#) scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-layouts](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the [ngcc](#) package use the below.

Add [@syncfusion/ej2-angular-layouts@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-layouts@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-layouts:"21.1.35-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Timeline module

Import Timeline module into Angular application(app.module.ts) from the package [@syncfusion/ej2-angular-layouts](#).

```
`javascript
import { NgModule } from "@angular/core";
import { BrowserModule } from "@angular/platform-browser";
// Import Syncfusion Timeline module from Timeline package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-layouts';
import { AppComponent } from './app.component';
@NgModule({
  imports: [BrowserModule, TimelineAllModule, TimelineModule], // Registering EJ2 Timeline Module.
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
```

```
export class AppModule {}  
`
```

Adding CSS reference

Add Timeline component's styles as given below in `style.css`.

```
`css  
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";  
@import "../node_modules/@syncfusion/ej2-layouts/styles/material.css";  
`
```

Adding Syncfusion Timeline component

Modify the template in `app.component.ts` file with `ejs-timeline` to render the Timeline component.

```
`javascript  
import { Component } from "@angular/core";  
import { StepModel, Timeline } from '@syncfusion/ej2-angular-layouts';  
@Component({  
  selector: "app-root",  
  template: `<!-- To Render Timeline. -->  
  <div>  
    <ejs-timeline id="timeline"></ejs-timeline>  
  </div>`,  
})  
export class AppComponent {  
}  
`
```

Adding Items

You can define Timeline items by using `<e-item>` tag directive.

```
`javascript  
import { Component } from "@angular/core";  
@Component({  
  selector: "app-root",  
  template: `<!-- To Render Timeline. -->  
  <div>  
    <ejs-timeline id="timeline">  
      <e-items>  
        <e-item></e-item>  
      </e-items>  
    </div>  
  `
```

```

</e-item></e-item>
<e-item></e-item>
<e-item></e-item>
</e-items>
</ejs-timeline>
</div>`,
  })
  export class AppComponent {
  }
  ,

```

Running the application

Run the application in the browser using the following command:

```
ng serve
```

The following example shows a default Timeline component.

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})

```

```
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline id="timeline">
    <e-items>
      <e-item></e-item>
      <e-item></e-item>
      <e-item></e-item>
      <e-item></e-item>
    </e-items>
  </ejs-timeline>
</div>
```

Items in Angular Timeline component

The Timeline items can be added by using the `<e-item>` tag directive. Each item can be configured with options such as `content`, `oppositeContent`, `dotCss`, `disabled` and `cssClass`.

Adding content

You can define the item content using the `content` property.

String content

You can define string content for the Timeline items.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public orderStatus: TimelineItemModel[] = [
    { content: 'Shipped' },
    { content: 'Departed' },
    { content: 'Arrived' },
    { content: 'Out for Delivery' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline>
    <e-items>
      <e-item *ngFor="let item of orderStatus"
[content]="item.content"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS*Templated content*

You can specify the template content for the items, by using the selector for an element in HTML.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent { }
```

```

    public templateContents = [
        { title: 'Shipped', description: 'Package details received', info: '-
Awaiting dispatch' },
        { title: 'Departed', description: 'In-transit', info: '(International
warehouse)' },
        { title: 'Arrived', description: 'Package arrived at nearest hub',
info: '(New york - US)' }
    ];
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
    <ejs-timeline>
        <e-items>
            <e-item [content]="contentTemplate"> </e-item>
            <e-item [content]="contentTemplate"> </e-item>
            <e-item [content]="contentTemplate"> </e-item>
            <e-item content="Out for Delivery"> </e-item>
        </e-items>
        <ng-template #contentTemplate let-data="">
            <div class="content-container">
                <div class="title">
                    {{ templateContents[data.itemIndex].title }}
                </div>
                <span class="description">
                    {{ templateContents[data.itemIndex].description }}
                </span>
            </div>
        </ng-template>
    </ejs-timeline>
</div>

```



```

        </span>
        <div class="info">
            {{ templateContents[data.itemIndex].info }}
        </div>
    </div>
</ng-template>
</ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.content-container {
    position: relative;
    width: 180px;
    padding: 10px;
    margin-left: 5px;
    box-shadow: rgba(9, 30, 66, 0.25) 0px 4px 8px -2px, rgba(9, 30, 66, 0.08)
0px 0px 0px 1px;
    background-color: ghostwhite;
}
.content-container::before {
    content: '';
    position: absolute;
    left: -8px;
    transform: translateY(-50%);
    width: 0;
    height: 0;
    border-top: 5px solid transparent;
    border-bottom: 5px solid transparent;
    border-right: 8px solid silver;
}
.content-container .title {
    font-size: 16px;
}
.content-container .description {
    color: #999999;
    font-size: 12px;
}
.content-container .info {
    color: #999999;
    font-size: 10px;
}

```

Adding opposite content

You can add additional information to each Timeline item, by using the [oppositeContent](#) property which is positioned opposite to the item content. Similar to the `content` property you can define `string` or `function` as contents to the `oppositeContent`.

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',

```

```

        styleUrls: ['./app.component.css'],
    })
    export class AppComponent {
        public mealTimes: TimelineItemModel[] = [
            { content: 'Breakfast', oppositeContent: '8:00 AM'},
            { content: 'Lunch', oppositeContent: '1:00 PM'},
            { content: 'Dinner', oppositeContent: '8:00 PM'},
        ];
    }

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
    <ejs-timeline>
        <e-items>
            <e-item *ngFor="let item of mealTimes" [content]="item.content"
[oppositeContent]="item.oppositeContent"> </e-item>
        </e-items>
    </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```


```

Dot item

You can define CSS class to set icons, background colors, or images to personalize the appearance of dots associated with each Timeline item by using the [dotCss](#) property.

Adding icons

You can define the CSS class to show the icon for each item using the `dotCss` property.

Adding images

You can include images for the Timeline items using the `dotCss` property, by setting the CSS `background-image` property.

Adding text

You can display text for the Timeline items using the `dotCss` property, by adding text to the CSS `content` property.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  public dotItems: TimelineItemModel[] = [
    { content: 'Default' },
    { content: 'Icon', dotCss: 'e-icons e-check' },
    { content: 'Image', dotCss: 'custom-image' },
    { content: 'Text', dotCss: 'custom-text' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
```

```
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline>
    <e-items>
      <e-item *ngFor="let item of dotItems" [content]="item.content"
[dotCss]="item.dotCss"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

INDEX.CSS

```
.e-dot.custom-image {
  background-image: url('./dot-image.png');
}
.e-dot.custom-text::before {
  content: 'A';
}
```

Disabling items

You can use the [disabled](#) property to disable an item when set to `true`. By default, the value is `false`.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public dailyRoutine: TimelineItemModel[] = [
    { content: 'Eat' },
    { content: 'Code' },
    { content: 'Repeat', disabled: true },
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-
layouts';
import { AppComponent } from './app.component';
```

```
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline>
    <e-items>
      <e-item *ngFor="let item of dailyRoutine"
[content]="item.content" [disabled]="item.disabled"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

cssClass

You can customize the appearance of the Timeline item by specifying a custom CSS class using the [cssClass](#) property.

Orientations in Angular Timeline component

The Timeline component supports the display of items in both horizontal and vertical direction by using the [orientation](#) property.

Vertical

You can display the items one below the other vertically by setting the [orientation](#) property to **Vertical**. By default, the items are displayed in vertical orientation.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
```

```

        templateUrl: './app.component.html',
        styleUrls: ['./app.component.css'],
    })
    export class AppComponent {
        public tripItinerary: TimelineItemModel[] = [
            { content: 'Day 1, 4:00 PM', oppositeContent: 'Check-in and campsite visit' },
            { content: 'Day 1, 7:00 PM', oppositeContent: 'Dinner with music' },
            { content: 'Day 2, 5:30 AM', oppositeContent: 'Sunrise between mountains' },
            { content: 'Day 2, 8:00 AM', oppositeContent: 'Breakfast and check-out' }
        ],
    };
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
    <ejs-timeline orientation="Vertical">
        <e-items>
            <e-item *ngFor="let item of tripItinerary"
            [content]="item.content" [oppositeContent]="item.oppositeContent"> </e-item>
        </e-items>
    </ejs-timeline>
</div>

```

APP.COMPONENT.CSS**Horizontal**

In horizontal orientation, the items are displayed in a side-by-side manner by setting the [orientation](#) property to **Horizontal**.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public tripItinerary: TimelineItemModel[] = [
    { content: 'Day 1, 4:00 PM', oppositeContent: 'Check-in and campsite visit' },
    { content: 'Day 1, 7:00 PM', oppositeContent: 'Dinner with music' },
    { content: 'Day 2, 5:30 AM', oppositeContent: 'Sunrise between mountains' },
    { content: 'Day 2, 8:00 AM', oppositeContent: 'Breakfast and check-out' }
  ],
};
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
```

```
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="margin-top: 50px;">
  <ejs-timeline orientation="Horizontal">
    <e-items>
      <e-item *ngFor="let item of tripItenerary"
[content]="item.content" [oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

Alignment in Angular Timeline component

You can display the Timeline content **Before**, **After**, **Alternate** and **AlternateReverse** by using the [align](#) property. The oppositeContent will be displayed parallel to the content when configured.

Before

In **Before** alignment, for **horizontal** orientation the item content is placed at the top and oppositeContent at the bottom whereas in **vertical**, the content to the left and oppositeContent to the right.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public frameworks: TimelineItemModel[] = [
    { content: 'ReactJs', oppositeContent: 'Owned by Facebook' },
    { content: 'Angular', oppositeContent: 'Owned by Google' },
    { content: 'VueJs', oppositeContent: 'Owned by Evan you' },
    { content: 'Svelte', oppositeContent: 'Owned by Rich Harris' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-
layouts';
```



```
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline align="Before">
    <e-items>
      <e-item *ngFor="let item of frameworks" [content]="item.content"
[oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS**After**

In [After](#) alignment, for **horizontal** orientation the item content is placed at the bottom and oppositeContent at the top whereas in **vertical**, the content to the right and oppositeContent to the left.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public frameworks: TimelineItemModel[] = [
    { content: 'ReactJs', oppositeContent: 'Owned by Facebook' },
```

```

    { content: 'Angular', oppositeContent: 'Owned by Google' },
    { content: 'VueJs', oppositeContent: 'Owned by Evan you' },
    { content: 'Svelte', oppositeContent: 'Owned by Rich Harris' }
  ];
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline align="After">
    <e-items>
      <e-item *ngFor="let item of frameworks" [content]="item.content"
[oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
</div>

```

APP.COMPONENT.CSS**Alternate**

In [Alternate](#) alignment, the item content are arranged alternatively regardless of the Timeline orientation.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public frameworks: TimelineItemModel[] = [
    { content: 'ReactJs', oppositeContent: 'Owned by Facebook' },
    { content: 'Angular', oppositeContent: 'Owned by Google' },
    { content: 'VueJs', oppositeContent: 'Owned by Evan you' },
    { content: 'Svelte', oppositeContent: 'Owned by Rich Harris' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline align="Alternate">
    <e-items>
      <e-item *ngFor="let item of frameworks" [content]="item.content"
[oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
```

```
</div>
```

APP.COMPONENT.CSS

Alternate reverse

In [AlternateReverse](#) alignment, the item content are arranged in reverse alternate regardless of the Timeline orientation.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public frameworks: TimelineItemModel[] = [
    { content: 'ReactJs', oppositeContent: 'Owned by Facebook' },
    { content: 'Angular', oppositeContent: 'Owned by Google' },
    { content: 'VueJs', oppositeContent: 'Owned by Evan you' },
    { content: 'Svelte', oppositeContent: 'Owned by Rich Harris' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-layouts';
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
```

```
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline align="AlternateReverse">
    <e-items>
      <e-item *ngFor="let item of frameworks" [content]="item.content"
[oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

Reverse in Angular Timeline component

You can display the Timeline items in reverse order, for different alignments by using the [reverse](#) property which provides adaptability and improves user interaction.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public careerProgress: TimelineItemModel[] = [
    { content: 'June 2022', oppositeContent: 'Graduated \n Bachelors in
Computer Engineering' },
    { content: 'Aug 2022', oppositeContent: 'Software Engineering
Internship \n ABC Software and Technology' },
    { content: 'Feb 2023', oppositeContent: 'Associate Software Engineer \n
ABC Software and Technology' },
    { content: 'Mar 2024', oppositeContent: 'Software Level 1 Engineer \n
XYZ Solutions' },
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
```

```

*/
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline reverse="true" align="Before">
    <e-items>
      <e-item *ngFor="let item of careerProgress"
[content]="item.content" [oppositeContent]="item.oppositeContent"> </e-item>
    </e-items>
  </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

Template in Angular Timeline component

The Timeline component allows you to customize the appearance for each item by using the [template](#) to modify the dot items, templated contents, progress bar styling and more.

The **template** context receives the following information:

| Type | Purpose |

| --- | --- |

| **item** | Indicates the current data of the Timeline item. |

| **itemIndex** | Indicates the current index of the Timeline item. |

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],

```

```

    })
    export class AppComponent {
        public projectMilestones: TimelineItemModel[] = [
            { content: 'Kickoff meeting'},
            { content: 'Content approved'},
            { content: 'Design approved'},
            { content: 'Product delivered'}
        ];
    }
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="margin-top: 50px;">
    <ejs-timeline orientation="Horizontal" cssClass="custom-timeline"
    [template]="timelineTemplate">
        <e-items>
            <e-item *ngFor="let item of projectMilestones"
            [content]="item.content"> </e-item>
        </e-items>
        <ng-template #timelineTemplate let-data="">
            <div class='template-container item-{{data.itemIndex}}'>
                <div class="content-container">
                    <div class="timeline-content"> {{data.item.content}}
                </div>
            </div>
            <div class="content-connector"></div>
        </ng-template>
    </e-items>
</div>

```

```

        <div class="progress-line">
            <span class="indicator"></span>
        </div>
    </div>
</ng-template>
</ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.container {
    height: 150px;
    width: 600px;
    margin: 50px auto;
}
.custom-timeline .e-timeline-item.e-item-template {
    align-items: flex-end;
}
.custom-timeline .e-timeline-items {
    justify-content: center;
}
.template-container .content-connector {
    position: absolute;
    left: 88%;
    width: 3px;
    height: 28px;
}
.template-container .content-container {
    padding: 8px;
    border-width: 1px;
    border-style: solid;
}
.content-container .timeline-content {
    font-size: 14px;
}
/* Color customizations - Progress line, connector line, dot border */
.item-0 .progress-line, .item-0 .content-connector { background-color:
rgb(233, 93, 93); }
.item-1 .progress-line, .item-1 .content-connector { background-color:
rgba(247, 179, 22, 0.907); }
.item-2 .progress-line, .item-2 .content-connector { background-color:
rgb(60, 184, 60); }
.item-3 .progress-line, .item-3 .content-connector { background-color:
rgb(153, 29, 230); }
.item-0 .progress-line .indicator, .item-0 .content-container { border-color:
rgb(233, 93, 93); }
.item-1 .progress-line .indicator, .item-1 .content-container { border-color:
rgba(247, 179, 22, 0.907); }
.item-2 .progress-line .indicator, .item-2 .content-container { border-color:
rgb(60, 184, 60); }
.item-3 .progress-line .indicator, .item-3 .content-container { border-color:
rgb(153, 29, 230); }
.item-0 .content-container { box-shadow: 2px 2px 8px rgb(233, 93, 93); }
.item-1 .content-container { box-shadow: 2px 2px 8px rgba(247, 179, 22,
0.907); }
.item-2 .content-container { box-shadow: 2px 2px 8px rgb(60, 184, 60); }

```



```
.item-3 .content-container { box-shadow: 2px 2px 8px rgb(153, 29, 230); }
/* START --- Customizing Dot and progress line */
.custom-timeline .template-container .indicator {
  position: absolute;
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background-color: #fff;
  border-width: 6px;
  border-style: solid;
  left: 88%;
  transform: translate(-50%, -40%);
  cursor: pointer;
}
.progress-line {
  position: absolute;
  height: 10px;
  width: 100%;
  left: 0;
  top: 50%;
}
/* END --- Customizing Icon and progress line */
```

Customization in Angular Timeline component

You can customize the Timeline items dot size, connectors, dot borders, dot outer space and more to personalize its appearance. This section explains the different ways for styling the items.

Connector styling

Common styling

You can define the styles applicable to the all the Timeline item connectors.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public dailyRoutine: TimelineItemModel[] = [
    { content: 'Eat' },
    { content: 'Code' },
    { content: 'Repeat' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-layouts';
```

```
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 250px;margin-top: 30px;">
  <ejs-timeline cssClass="custom-connector">
    <e-items>
      <e-item *ngFor="let item of dailyRoutine"
[content]="item.content"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

```
.custom-connector .e-timeline-item.e-connector::after {
  border-color: #f7c867;
  border-width: 1.4px;
}
```

Individual styling

You can also apply unique styles to individual connectors, to differentiate specific items within the Timeline.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public dailyRoutine: TimelineItemModel[] = [
```

```

        { content: 'Eat', cssClass: 'state-initial' },
        { content: 'Code', cssClass: 'state-intermediate' },
        { content: 'Repeat' }
    ];
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 250px;margin-top: 30px;">
    <ejs-timeline cssClass="custom-connector">
        <e-items>
            <e-item *ngFor="let item of dailyRoutine"
[content]="item.content" [cssClass]="item.cssClass"> </e-item>
        </e-items>
    </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.custom-connector .state-initial.e-connector::after {
    border: 1.5px #f8c050 dashed;
}
.custom-connector .state-intermediate.e-connector::after {
    border: 1.5px #4d85f5 dashed;
}

```

Dot styling

Dot color

You can modify the color of the dots to highlight the specific Timeline items.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {

  public orderStatus: TimelineItemModel[] = [
    { content: 'Ordered', cssClass: 'state-completed' },
    { content: 'Shipped', cssClass: 'state-progress' },
    { content: 'Delivered' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 250px;margin-top: 30px;">
  <ejs-timeline cssClass="dot-color">
    <e-items>
```

```

        <e-item *ngFor="let item of orderStatus" [content]="item.content"
        [cssClass]="item.cssClass"> </e-item>
    </e-items>
</ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.dot-color .state-completed .e-dot {
    background: #ff9900 ;
    outline: 1px dashed #ff9900;
    border-color: #ff9900;
}
.dot-color .state-progress .e-dot {
    background: #33cc33;
    outline: 1px dashed #33cc33;
    border-color: #33cc33;
}

```

Dot size

You can adjust the size of the dot to make it larger or smaller by using the `--dot-size` variable.

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css'],
})
export class AppComponent {

    public dotSizes: TimelineItemModel[] = [
        { content: 'Extra Small', cssClass: 'x-small' },
        { content: 'Small', cssClass: 'small' },
        { content: 'Medium', cssClass: 'medium' },
        { content: 'Large', cssClass: 'large' }
    ];
}

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from '@syncfusion/ej2-angular-
layouts';
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],

```

```

    declarations: [AppComponent],
    bootstrap: [AppComponent]
  })
  export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 250px;margin-top: 30px;">
  <ejs-timeline cssClass="dot-size">
    <e-items>
      <e-item *ngFor="let item of dotSizes" [content]="item.content"
[cssClass]="item.cssClass"> </e-item>
    </e-items>
  </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.dot-size .e-dot {
  background: #33cc33;
}
.dot-size .x-small .e-dot {
  --dot-size: 12px;
}
.dot-size .small .e-dot {
  --dot-size: 18px;
}
.dot-size .medium .e-dot {
  --dot-size: 24px;
}
.dot-size .large .e-dot {
  --dot-size: 30px;
}

```

Dot shadow

You can add shadow effects to the Timeline dots to make it feel visually engaging by using the `--dot-outer-space` & `--dot-border` variables.

APP.COMPONENT.TS

```

import { Component } from '@angular/core';
import { TimelineItemModel } from '@syncfusion/ej2-angular-layouts';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',

```

```

        styleUrls: ['./app.component.css'],
    })
    export class AppComponent {

        public orderStatus: TimelineItemModel[] = [
            { content: 'Ordered' },
            { content: 'Shipped' },
            { content: 'Delivered' }
        ];
    }

```

APP.MODULE.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
    //declaration of ej2-angular-layouts module into NgModule
    imports: [BrowserModule, TimelineModule, TimelineAllModule ],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 250px;margin-top: 30px;">
    <ejs-timeline cssClass="dot-shadow">
        <e-items>
            <e-item *ngFor="let item of orderStatus"
[content]="item.content"> </e-item>
        </e-items>
    </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

```

.dot-shadow .e-dot {
    --dot-outer-space: 3px;
    --dot-border: 3px;
}

```

```
--dot-size: 20px;
outline-color: #dee2e6;
border-color: #fff;
box-shadow: 3px 3px 10px rgba(0, 0, 0, 0.5) ,
2px -2px 4px rgba(255, 255, 255, 0.5) inset;
}
```

Dot variant

You can achieve the desired dot variant by customizing the border, outline and background colors of the Timeline dots.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {

  public dotVariants: TimelineItemModel[] = [
    { content: 'Filled', cssClass: 'dot-filled' },
    { content: 'Flat', cssClass: 'dot-flat' },
    { content: 'Outlined', cssClass: 'dot-outlined' }
  ];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
```



```
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 250px;margin-top: 30px;">
  <ejs-timeline cssClass="dot-variant">
    <e-items>
      <e-item *ngFor="let item of dotVariants" [content]="item.content"
[cssClass]="item.cssClass"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

```
.dot-variant .dot-filled .e-dot::before { content: 'A'; color: #fff;}
.dot-variant .dot-flat .e-dot::before { content: 'B'; color: #fff;}
.dot-variant .dot-outlined .e-dot::before { content: 'C';}
.dot-variant .dot-filled .e-dot {
  background: #33cc33;
  --dot-outer-space: 3px;
  outline-color: #81ff05;
  --dot-size: 25px;
}
.dot-variant .dot-flat .e-dot {
  background: #33cc33;
  --dot-size: 25px;
  --dot-radius: 10%;
}
.dot-variant .dot-outlined .e-dot {
  outline-color: #33cc33;
  --dot-outer-space: 3px;
  background-color: unset;
  --dot-size: 25px;
}
```

Dot outline

By adding the `e-outline` class to the Timeline `cssClass` property it enables the dots to have an outline state.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {

  public orderStatus: TimelineItemModel[] = [
    { content: 'Shipped' },
    { content: 'Departed' },
    { content: 'Arrived' },
  ]
```

```
{ content: 'Out for Delivery'}
];
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 250px;margin-top: 30px;">
  <ejs-timeline cssClass="e-outline">
    <e-items>
      <e-item *ngFor="let item of orderStatus"
[content]="item.content"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

Accessibility in Angular Timeline component

The Timeline 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 Timeline component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device 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 following ARIA attributes are used in the Timeline component:

| Attributes | Purpose |

| --- | --- |

| `role=navigation` | Specified its purpose as a navigational element. |

| `aria-label` | Provides an accessible name for an element when a visible label is not present |

Ensuring accessibility

The Timeline component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

See also

- [Accessibility in Syncfusion Angular components](#)

Events in Angular Timeline component

This section describes the Timeline events that will be triggered when an appropriate actions are performed. The following events are available in the Timeline component.

created

The Timeline component triggers the [created](#) event when the component rendering is completed.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel } from "@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public productLifecycle: TimelineItemModel[] = [
    { content: 'Planning' },
    { content: 'Developing' },
    { content: 'Testing' },
    { content: 'Launch' },
  ];
  handleTimelineCreated = () => {
    //your required action here
  }
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-layouts";
import { AppComponent } from './app.component';
/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MAIN.TS

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
```

```
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);
```

APP.COMPONENT.HTML

```
<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline (created)="handleTimelineCreated()">
    <e-items>
      <e-item *ngFor="let item of productLifecycle"
[content]="item.content"> </e-item>
    </e-items>
  </ejs-timeline>
</div>
```

APP.COMPONENT.CSS

beforeItemRender

The Timeline component triggers the [beforeItemRender](#) event before rendering each item.

APP.COMPONENT.TS

```
import { Component } from '@angular/core';
import { TimelineItemModel, TimelineRenderingEventArgs } from
"@syncfusion/ej2-angular-layouts";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class AppComponent {
  public productLifecycle: TimelineItemModel[] = [
    { content: 'Planning' },
    { content: 'Developing' },
    { content: 'Testing' },
    { content: 'Launch' },
  ];

  handleBeforeItemRender = (args: TimelineRenderingEventArgs) => {
    //your required action here
  }
}
```

APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// Import Syncfusion Timeline module from layouts package.
import { TimelineModule, TimelineAllModule } from "@syncfusion/ej2-angular-
layouts";
import { AppComponent } from './app.component';
```

```

/**
 * Module
 */
@NgModule({
  //declaration of ej2-angular-layouts module into NgModule
  imports: [BrowserModule, TimelineModule, TimelineAllModule ],
  declarations: [AppComponent],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MAIN.TS

```

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { enableProdMode } from '@angular/core';
import { AppModule } from './app.module';
import 'zone.js';
enableProdMode();
platformBrowserDynamic().bootstrapModule(AppModule);

```

APP.COMPONENT.HTML

```

<div class="container" style="height: 330px;margin-top: 30px;">
  <ejs-timeline (beforeItemRender)="handleBeforeItemRender($event)">
    <e-items>
      <e-item *ngFor="let item of productLifecycle"
[content]="item.content"> </e-item>
    </e-items>
  </ejs-timeline>
</div>

```

APP.COMPONENT.CSS

TimePicker

Getting started with Angular Timepicker component

This section briefly explains how to create a simple [Link to the Video](#) component, and configure its available functionalities in Angular by using [quickstart](#) seed repository.

To get start quickly with Angular TimePicker component, refer to the video below.

Dependencies

Install the below required dependency package in order to use the TimePicker component in your application.

```
`javascript
```

```
|-- @syncfusion/ej2-angular-calendars
```

```
|-- @syncfusion/ej2-angular-base
```

```
|-- @syncfusion/ej2-base
```

```
|-- @syncfusion/ej2-calendars  
|-- @syncfusion/ej2-inputs  
|-- @syncfusion/ej2-splitbuttons  
|-- @syncfusion/ej2-lists  
|-- @syncfusion/ej2-popups  
|-- @syncfusion/ej2-buttons  
,
```

Setup Angular environment

Angular provides the easiest way to set angular CLI projects using [Angular CLI](#) tool.

Install the CLI application globally to your machine.

```
`bash  
npm install -g @angular/cli  
,
```

Create a new application

```
`bash  
ng new syncfusion-angular-timepicker  
,
```

By default, it install the CSS style base application. To setup with SCSS, pass `--style=scss` argument on create project.

Example code snippet.

```
`bash  
ng new syncfusion-angular-timepicker --style=scss  
,
```

Navigate to the created project folder.

```
`bash  
cd syncfusion-angular-timepicker  
,
```

Installing Syncfusion TimePicker package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(>=20.2.36) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-calendars](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars --save
`
```

Angular compatibility compiled package(ngcc)

For Angular version below 12, you can use the legacy (ngcc) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-calendars@ngcc](#) package to the application.

```
`bash
npm install @syncfusion/ej2-angular-calendars@ngcc --save
`
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
@syncfusion/ej2-angular-calendars:"20.2.38-ngcc"
`
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Registering TimePicker module

Import TimePicker module into Angular application(src/app/app.module.ts) from the package [@syncfusion/ej2-angular-calendars](#).

```
`javascript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
// import the TimePickerModule for the TimePicker component
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars';
import { AppComponent } from './app.component';

@NgModule({
  //declaration of TimePickerModule into NgModule
  imports: [ BrowserModule, TimePickerModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
```



```
})  
export class AppModule { }  
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder.

This can be referenced in `[src/styles.css]` using following code.

```
`css  
  
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-inputs/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-lists/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-calendars/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-angular-calendars/styles/material.css';  
`
```

If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Adding TimePicker component

Modify the template in `[src/app/app.component.ts]` file to render the TimePicker component.

Add the Angular TimePicker by using `<ejs-timepicker>` selector in `template` section of the `[app.component.ts]` file.

```
`javascript  
  
import { Component } from '@angular/core';  
import { enableRipple } from '@syncfusion/ej2-base';  
  
//enable ripple style  
enableRipple(true);  
  
@Component({  
  selector: 'app-root',  
  template: `<!-- To Render TimePicker -->  
<ejs-timepicker></ejs-timepicker>`  
})  
  
export class AppComponent { }  
`
```

Running the application

After completing the configuration required to render a basic TimePicker, run the following command to display the output in your default browser.

`

ng serve

`

The following example illustrates the output in your browser

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<ejs-timepicker placeholder='Select a time' ></ejs-timepicker>`
})
export class AppComponent {
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Setting the selected, min, and max time

The following example demonstrates how to set the value, min and max time on initializing the TimePicker. The TimePicker allows you to select the time value within a range from 7:00 AM to 4:00 PM.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
```

```
enableRipple(true);
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [value]='dateValue' [min]='minValue'
    [max]='maxValue'></ejs-timepicker>
  `
})
export class AppComponent {
  public month: number = new Date().getMonth();
  public fullYear: number = new Date().getFullYear();
  public date: number = new Date().getDate();
  public dateValue: Date = new Date(this.fullYear, this.month, this.date,
10, 0, 0);
  public minValue: Date = new Date(this.fullYear, this.month, this.date,
7, 0, 0);
  public maxValue: Date = new Date(this.fullYear, this.month, this.date,
16, 0, 0);
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Setting the time format

Time formats is a way of representing the time value in different string format in textbox and popup list. By default, the TimePicker's format is based on the culture. You can also customize the format by using the [format](#) property. To know more about the time format standards, refer to the [Date and Time Format](#) section.

The following example demonstrates the TimePicker component in 24 hours format with 60 minutes interval. The time interval is set to 60 minutes by using the [step](#) property.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [
```

```

        TimePickerModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
        <ejs-timepicker [placeholder]='watermark' [format]='formatString'
        [step]='interval' ></ejs-timepicker>
    `
  })
  export class AppComponent {
    public watermark: string = 'Select a time';
    // sets the format property to display the time value in 24 hours format.
    public formatString: string = 'HH:mm';
    public interval: number = 60;
    constructor() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Once the format property is defined, it will be applicable to all the cultures.

See Also

- [Render TimePicker with min and max time](#)
- [How to achieve validation with TimePicker](#)
- [Render TimePicker with specific culture](#)
- [How to achieve two-way binding with TimePicker](#)
- [Reactive forms with TimePicker](#)

Time range in Angular Timepicker component

TimePicker provides an option to select a time value within a specified range by using the [min](#) and [max](#) properties. The min value should always be lesser than the max value.

When the min and max properties are configured and the selected time value is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The following example allows you to select a time value within a range of **9:00 AM** to **11:30 AM**.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';

```

```
//enable ripple style
enableRipple(true);
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [value]='dateValue' [min]='minDate'
    [max]='maxDate'></ejs-timepicker>
  `
})
export class AppComponent {
  public minDate: Date = new Date('8/3/2017 9:00 AM');
  public maxDate: Date = new Date('8/3/2017 11:30 AM');
  public dateValue: Date = new Date('8/3/2017 10:00 AM');
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the value of `min` or `max` property is changed through code behind you have to update the `value` property to set within the range.

Time masking in Angular Timepicker component

TimePicker has `enableMask` property that provides the option to enable the built-in time masking support. Also, you must inject the `MaskedDateTimeService` module to enable the masking support.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [enableMask]="enableMaskSupport"></ejs-timepicker>
  `
  ,
  providers: [MaskedDateTimeService],
})
export class AppComponent {
```

```

    public format: string = "HH:mm";
    public enableMaskSupport: boolean = true;
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

The mask pattern is defined based on the provided time 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 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 TimePicker component with mask.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
    imports: [
        TimePickerModule
    ],
    standalone: true,
    selector: 'app-root',
    templateUrl: './format.html',
    providers: [MaskedDateTimeService],
})
export class AppComponent {
    constructor() {
        public format: string = "hh:mm:ss";
        public enableMaskSupport: boolean = true;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of time co-ordinates such as `hour`, `minute` and `second`.

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.

```
`typescript
//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': {
    timepicker: { hour: 'Stunde', minute: 'Minute', second: 'Sekunde' }
  }
});
`
```

The following example demonstrates default and customized mask placeholder value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars';
import { Component } from '@angular/core';
import { MaskedDateTimeService } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './maskplaceholder.html',
  providers: [MaskedDateTimeService],
})
export class AppComponent {
  constructor() {
    }
    public enableMaskSupport: boolean = true;
    public maskPlaceholderValue: Object = {hour: 'h', minute: 'm', second:
's'}
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Globalization in Angular Timepicker component

Globalization is the combination of internationalization and localization. You can adapt the component to various languages by means of parsing and formatting the date or number [internationalization](#) and also add culture specific customization and translation to the text [localization](#).

By default, the 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 go with the different culture other than English, follow the steps below.

- Install the `CLDR-Data` package by using the following command (it installs all 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.component.ts` file.
- Now use the `loadCldr` method to load the culture specific CLDR JSON data from the installed location to `app.component.ts` file.
- TimePicker displayed Sunday as the first day of week based on default culture ("en-US"). If you want to display the TimePicker 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.

```
`typescript
```

```
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/supplemental/weekdata.json') // To load the culture based first day of week
);
```

- Before changing to a culture other than English, ensure that the locale text for the concerned culture is loaded through `L10n.load` method.


```
`typescript
//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': {
    'timepicker': { placeholder: 'Wählen Sie Zeit' }
  }
});
`
```

- Set the culture by using the [locale](#) property. In the following code example, the TimePicker component is initialized in **German** culture with corresponding localized text.

```
`typescript
import { Component } from '@angular/core';
import { L10n, loadCldr } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
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')
);
@Component({
  selector: 'app-root',
  template: `
    <ejs-timepicker [locale]='culture'></ejs-timepicker>
  `
})
export class AppComponent {
  public culture: string = 'de';
  constructor() {}
}
```

```

ngOnInit(): void {
  L10n.load({
    'de': {
      'timepicker': { placeholder: 'Wählen Sie Zeit' }
    }
  });
}
}
,

```

The following sample demonstrate the TimePicker component in **German** culture.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
loadCldr(numberingSystems, gregorian, numbers);
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [locale]='locale' [value]='dateValue'></ejs-
    timepicker>
  `
})
export class AppComponent {
  public locale: string = 'de';
  public dateValue: Date = new Date();
  constructor() {}
  ngOnInit(): void {
    L10n.load({
      'de': {
        'timepicker': { placeholder: 'Wählen Sie Zeit' }
      }
    });
  }
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Right-To-Left

The TimePicker 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 demonstrates the TimePicker component in **Arabic** culture. It also explains how to set localized text to the placeholder using **L10n.load** method.

`typescript

```
import { Component } from '@angular/core';
import { L10n, loadCldr } from '@syncfusion/ej2-base';
import { enableRipple } from '@syncfusion/ej2-base';

//enable ripple style
enableRipple(true);

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')
);

@Component({
  selector: 'app-root',
  template: `
    <ejs-timepicker [locale]='culture' [enableRtl]='isRTL' [value]='dateValue'></ejs-timepicker>
  `
})

export class AppComponent {
  public culture: string = 'ar';
  public isRTL: boolean = true;
  public dateValue: Date = new Date();
  constructor() {
  }
}
```

```

ngOnInit(): void {
  L10n.load({
    'ar': {
      'timepicker': { placeholder: 'حدد الوقت' }
    }
  });
}

```

The following example demonstrates TimePicker in **Arabic** culture with right-to-left direction.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { loadCldr, L10n } from '@syncfusion/ej2-base';
// Here we have referred local json files for preview purpose
import * as numberingSystems from './numberingSystems.json';
import * as gregorian from './ca-gregorian.json';
import * as numbers from './numbers.json';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
loadCldr(numberingSystems, gregorian, numbers);
@Component({
  imports: [
    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [locale]='locale' [enableRtl]='isRTL'
    [value]='dateValue'></ejs-timepicker>
  `
})
export class AppComponent {
  public locale: string = 'ar';
  public isRTL: boolean = true;
  public dateValue: Date = new Date();
  constructor() {
  }
  ngOnInit(): void {
    L10n.load({
      'ar': {
        'timepicker': { placeholder: 'حدد الوقت' }
      }
    });
  }
}

```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Strict mode in Angular Timepicker component

The [strictMode](#) is an act that allows you to enter only valid time value within the specified min/max range in the textbox. If the time value is invalid, the component value sets to the previous value.

If the time value is out of range, the component sets the time value to min/max value.

The following example demonstrates the TimePicker in `strictMode` with min/max range of `10:00 AM` to `4:00 PM`. It allows you to enter only valid time within the specified range. If you enter the out-of-range value like `8:00 PM`, the value sets to the max time `4:00 PM` as the value `8:00 PM` is greater than max value of `4:00 PM`. If you enter invalid time value like `9:00 tt`, the value sets to the previous value.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [

    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [value]='dateValue' [min]='minValue' [max]='maxValue'
    [strictMode]='isStrictMode'></ejs-timepicker>
  `
})
export class AppComponent {
  public dateValue: Date = new Date('10/8/2017 3:00 PM');
  public minValue: Date = new Date('10/8/2017 10:00 AM');
  public maxValue: Date = new Date('10/8/2017 4:00 PM');
  public isStrictMode: boolean = true;
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

By default, the TimePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range time in textbox.

If the time is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time 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 time value, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [

    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker [value]='dateValue' [min]='minValue' [max]='maxValue'
    [strictMode]='isStrictMode'></ejs-timepicker>
  `
})
export class AppComponent {
  public dateValue: Date = new Date('10/8/2017 4:00 PM');
  public minValue: Date = new Date('10/8/2017 5:00 AM');
  public maxValue: Date = new Date('10/8/2017 10:00 PM');
  public isStrictMode: boolean = false;
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

If the value of `min` or `max` property is changed through code behind, update the `value` property to set within the range.

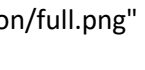
Accessibility in Angular Timepicker component

The TimePicker 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 TimePicker component is outlined below.

| Accessibility Criteria | Compatibility |

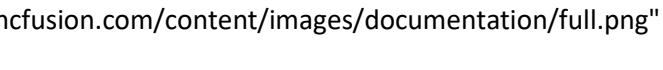
| -- | -- |

| [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 applications and its content more accessible to people with disabilities without any barriers. Especially it tracks the dynamic value changes and DOM changes.

TimePicker component has covered the [WAI-ARIA](#) specifications with the following list of WAI-ARIA attributes `aria-haspopup`, `aria-selected`, `aria-disabled`, `aria-activedescendant`, `aria-expanded`, `aria-owns`, and `aria-autocomplete`.

Here, the `combobox` as a role for input element and `listbox` as a role for popup element in the TimePicker.

- **Aria-haspopup** : Provides the information about whether this element display a pop-up window or not.
- **Aria-selected** : Indicates the current selected value of the TimePicker component.
- **Aria-disabled** : Indicates disabled state of the TimePicker component.
- **Aria-expanded** : Indicates expanded state of the popup.
- **Aria-autocomplete** : Indicates whether user input completion suggestions are provided or not.
- **Aria-owns** : Attribute that creates a parent/child relationship between two DOM element in the accessibility layer.
- **Aria-activedescendent** : Attribute that helps in managing the current active child of the TimePicker

component.

- **Role** : Attribute that gives assistive technology information for handling each element in a widget.

Keyboard Interaction

Keyboard accessibility is one of the most important aspects of web accessibility. It will be more useful to all the computer users, as they often allow to interact keyboard more than a mouse.

Among people with disabilities like blind or who have motor disabilities are also can make frequent use of keyboard shortcuts.

The TimePicker component has built-in keyboard accessibility support by following the [WAI-ARIA practices](#).

It supports the following list of shortcut keys to interact the TimePicker control.

| 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 |

The following sample use the **alt+t** keys to focus the TimePicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, HostListener, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [

    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <ejs-timepicker #keyboard [value]='dateValue'></ejs-timepicker>
  `
})
export class AppComponent {
  @ViewChild('keyboard') timeObj: any;
  public dateValue: Date = new Date();
  @HostListener('document:keyup', ['$event'])
  handleKeyboardEvent(event: KeyboardEvent) {
    if (event.altKey && event.keyCode === 84 /* t */) {
      // press alt+t to focus the control by calling public method.
      this.timeObj.focusIn();
    }
  }
  constructor() {
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Ensuring accessibility

The TimePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TimePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TimePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

Style appearance in Angular Timepicker component

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 TimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height and font size /
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
font-size: 20px;
height: 40px;
}
```

Customizing the TimePicker icon element

Use the following CSS to customize the TimePicker icon element

```
`css
/ To specify background color and font size /
.e-time-wrapper .e-time-icon.e-icons, *.e-control-wrapper.e-time-wrapper .e-time-icon.e-icons {
font-size: 20px;
background-color: beige;
}
```

Customizing the TimePicker popup

Use the following CSS to customize the TimePicker popup

```
`css
/ To specify height /
.e-timepicker.e-popup {
height: 100px;
}
```

`

Customizing the TimePicker popup content

Use the following CSS to customize the TimePicker popup content

`css

/ To specify height /

```
.e-timepicker.e-popup .e-list-parent.e-ul li.e-list-item {
```

```
background-color: beige;
```

```
font-size: 20px;
```

```
}
```

`

Full screen mode support in mobiles and tablets

The TimePicker 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 TimePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the popup element to occupy the entire screen on mobile devices.

`javascript

```
import { Component } from '@angular/core';
```

```
@Component({
```

```
selector: 'app-root',
```

```
template: `<!-- To Render timepicker -->
```

```
<ejs-timepicker [fullScreenMode]="true"></ejs-timepicker >`
```

```
})
```

```
export class AppComponent { }
```

`

Default Sample 

How To

Json data binding with timepicker in Angular Timepicker component

In most of the real cases, the model data will be available with JSON format only. Here we have showcased TimePicker component by setting JSON string to value property.

In this JSON, we have used ISO formatted date string which is frequently used date format to get proper date and time value without any misreading.

Also our TimePicker component supports the ISO formatted date value, so parsed JSON value can be directly set to TimePicker model.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component } from '@angular/core';
export interface User {
    selectedDate: Date;
}
export interface JSONUser {
    selectedDate: string;
}
@Component({
    imports: [

        TimePickerModule //declaration of ej2-angular-calendars module into
NgModule
    ],
    standalone: true,
    selector: 'app-root',
    template: `
<!-- To Render TimePicker -->
<ejs-timepicker id="timepicker" width="240px"
[ (value) ]='user.selectedDate' (change)='onChange($event)'></ejs-timepicker>
<div class="valuestring">
<b>User model</b>: <br/>{{user | json }}
<br/><br/>
<b>JSON Data</b>: <br/>{{ model_result }}
<br/><br/>
</div>`
    })
export class AppComponent {
    public user?: User | any;
    public JSONData: JSONUser = JSON.parse('{ "selectedDate": "2018-12-18T08:56:00+00:00"}');
    public model_result: string = JSON.stringify(this.JSONData);
    public ngOnInit() {
        this.user = this.JSONData;
    }
    onChange(args: any) {
        this.JSONData.selectedDate = args.value;
        this.model_result = JSON.stringify(this.JSONData);
    }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Two way binding in Angular Timepicker component

The following example demonstrates how to achieve **two-way binding** by binding the **value** to the first TimePicker component by using property binding and binding the model data using **ngModel** to the second TimePicker component. The **value** of the TimePicker will get change, when their is any change in the property value or model value.

The two-way binding can also be achieved only by using **property binding** or **model binding** in the TimePicker component.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { TimePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
  imports: [

    TimePickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <!-- two-way binding using the value binding and model binding in the
    TimePicker -->
    <ejs-timepicker id="firsttime" #ejTimePicker [(value)]= 'value'
    width="230px"></ejs-timepicker>
    <ejs-timepicker id="secondtime" #ejTimePickers [(ngModel)]= 'value'
    width="230px"></ejs-timepicker>
  `
})
export class AppComponent {
  value: Date;
  constructor() {
    this.value = new Date("1/1/2019 1:30 PM");
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Css customization in Angular Timepicker component

TimePicker allows you to customize the textbox and popup list appearance to suit for your application by using [cssClass](#) property.

The below sample demonstrates customization of text appearance in a textbox, popup button, and popup list along with hover and active state by using `e-custom-style` class. Following is the list of available classes used to customize the entire TimePicker component.

Class Name	Description
---	---
e-time-wrapper	Applied to TimePicker wrapper element.
e-timepicker	Applied to input element and TimePicker popup element.
e-time-wrapper.e-timepicker	Applied to input element only.
e-input-group-icon.e-time-icon	Applied to popup button.
e-float-text	Applied to floating label text element.
e-popup	Applied to popup element.
e-timepicker.e-popup	Applied to TimePicker popup element only.
e-list-parent	Applied to popup UL element.
e-timepicker.e-list-parent	Applied to TimePicker popup UL element only.
e-list-item	Applied to LI elements.
e-hover	Applied to LI element hovering time.
e-active	Applied to active LI element.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { FormsModule } from '@angular/forms'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { enableRipple } from '@syncfusion/ej2-base';
//enable ripple style
enableRipple(true);
@Component({
  imports: [
    TimePickerModule,
    FormsModule
  ],
  standalone: true,
  selector: 'app-root',
  styleUrls: ['./style.css'],
  template: `
    <ejs-timepicker [placeholder]='watermark'
    [cssClass]='customClass'></ejs-timepicker>`
})
```

```

    })
    export class AppComponent {
        public watermark: string = 'Select a time';
        public customClass: string = 'e-custom-style';
    }

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom event emitter in Angular Timepicker component

The **two-way binding** in TimePicker can also be achieved using the custom event binding and property binding in the controls present in two different components. To create custom event, we need to create an instance of **event emitter**.

In the following example, **property binding** is used to share the data from the parent component to the child component using **@input** directive and **custom event binding** is used to share the data from the child component to the parent component by using **@output** directive.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { TimePickerComponent } from '@syncfusion/ej2-angular-calendars';
@Component({
    imports: [ TimePickerModule ],
    standalone: true,
    selector: 'app-root',
    template: `
        <div class="parentelement">
            <div class="datevalue">
                <ejs-timepicker id="timepicker" #time (change)="deposit()"
                placeholder="Parent component" floatLabelType="Always" [value]="value"
                width="200px"></ejs-timepicker>
            </div>
        </div>
        <child [xvalue]="value" (valueChange)="valuecheck($event)"> </child>
    `
})
export class ParentComponent {
    @ViewChild('time')
    public Time?: TimePickerComponent;
    value: Date | any;
    constructor() {
        this.value = new Date("2/1/2020");
    }
    deposit() {
        this.value = this.Time?.value;
    }
    valuecheck(args: any) {

```



```

        this.value = args;
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Custom validation using form validator in Angular Timepicker component

The client side validation takes place in the browser to avoid the waiting time to receive the response from sever. It validates the form elements to provide the better feedback messages to correct the every fields before the form submission.

To achieve the client side validation in a TimePicker component by using **FormValidator** function. It provides an option to customize the feedback error messages to the corresponding fields to take action to resolve the issue.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, OnInit } from '@angular/core';
import { TimePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
@Component({
  imports: [

    TimePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<form id="form-element" class="form-vertical">
    <ejs-timepicker #ejTime id='timepicker' placeholder='Select a time'
    width="275px" (blur)="onFocusOut()" (change)="onChange($event)"></ejs-
    timepicker>
  </form>`
})
export class AppComponent implements OnInit {
  @ViewChild('formElement') element: any;
  @ViewChild('ejTime') ejTime: any;
  public formObject?: FormValidator;
  ngOnInit() {
    // custom validator function.
    let customFn: (args: {
      [key: string]: string
    }) => boolean = (args: {
      [key: string]: string
    }) => {
      return ((this.ejTime.value).getHours() < 10);
    };
    let options: FormValidatorModel = {

```

```

        rules: {
            'timepicker': {
                required: [true, "Value is required"]
            }
        },
        customPlacement: (inputElement: HTMLElement, errorElement:
HTMLElement) => {
            inputElement.parentElement?.appendChild(errorElement);
        }
    };
    this.formObject = new FormValidator('#form-element', options);
    this.formObject.addRules('timepicker', {
        range: [customFn, "Please select a time between 12:00 AM to 10:00
AM"]
    });
    this.formObject = new FormValidator('#form-element', options);
}
// Form validation takes place when focus() event of timepicker is
triggered.
public onFocusOut(): void {
    this.formObject?.validate("timepicker");
}
// Custom validation takes place when value is changed.
public onChange(args: any) {
    if (this.ejTime.value != null)
        this.formObject?.validate("timepicker");
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Reactive form in Angular Timepicker component

TimePicker is a form component and validation is playing vital role in forms to get the valid data. Here to showcase the TimePicker with form validations we have used the reactive form.

- The reactive forms uses the reactive model-driven technique to handle form data between component and view, due to that we also call it as the model-driven forms. It's listen the form data changes between App component and view also returns the valid states and values of form elements.

For more details about Reactive Forms refer: <https://angular.io/guide/reactive-forms>.

- For the reactive forms, import a **ReactiveFormsModule** into app module as well as the **FormGroup**,

FormControl should be imported to app component. The **FormGroup** is used to declare **formGroupName** for the form and the **FormControl** is used to declare **formControlName** for form

controls. You can declare the `formControlName` to TimePicker component as usual. Then, create a value object to the `FormGroup` and each value will be the default value of the form control.

The following example demonstrates how to use the reactive forms.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { ButtonModule } from '@syncfusion/ej2-angular-buttons'
import { Component, ViewChild, OnInit, ElementRef, Inject } from '@angular/core';
import { TimePickerComponent } from '@syncfusion/ej2-angular-calendars';
import { FormValidator, FormValidatorModel } from '@syncfusion/ej2-inputs';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ButtonComponent } from '@syncfusion/ej2-angular-buttons';
import { FormsModule } from '@angular/forms';
@Component({
  imports: [
    FormsModule, ReactiveFormsModule, TimePickerModule, ButtonModule
  ],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class AppComponent implements OnInit {
  @ViewChild('ejTimePicker') ejTimePicker?: TimePickerComponent;
  public targetElement?: HTMLElement;
  public placeholder: String = 'Select a time';
  skillForm?: FormGroup | any;
  build: FormBuilder;
  constructor(@Inject(FormBuilder) private builder: FormBuilder) {
    this.build = builder;
    this.createForm();
  }
  ngOnInit(): void {
  }
  createForm() {
    this.skillForm = this.build.group({
      timepicker: ['', Validators.required],
      username: ['', Validators.required],
      usermail: ['', Validators.email],
    });
  }
  get username() {
    return this.skillForm?.get('username');
  }
  get timepicker() {
    return this.skillForm?.get('timepicker');
  }
  get usermail() {
    return this.skillForm?.get('usermail');
  }
  onSubmit() {
```

```

        alert("Form Submitted!");
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Template driven forms in Angular Timepicker component

The form can be build with Angular template syntax easily along with form specific directives. This template-driven forms uses the `ng` directives in view to handle the forms controls.

- To enable the template-driven, import the FormsModule into corresponding app component.

For more details about template-driven Forms refer to: <https://angular.io/guide/forms#template-driven-forms>.

- In angular forms mentioning the name is must to process as form elements.
- Mention the `name` attribute to TimePicker element which will be used to identify the form element. To register an TimePicker element to ngForm, give the ngModel to it so the FormsModule will automatically detect the TimePicker as a form element. After that, the TimePicker value will be selected based on the ngModel value.

The following example demonstrates template driven forms with TimePicker component.

APP.COMPONENT.TS

```

import { BrowserModule } from '@angular/platform-browser'
import { NgModule } from '@angular/core'
import { FormsModule, ReactiveFormsModule } from '@angular/forms'
import { TimePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild, ViewEncapsulation, Inject } from '@angular/core';
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import { TimePickerComponent } from '@syncfusion/ej2-angular-calendars';

class User {
    constructor() {

    }
}

@Component({
  imports: [ FormsModule, ReactiveFormsModule, TimePickerModule],
  standalone: true,
  selector: 'app-root',
  templateUrl: './template.html'
})
export class DefaultTimePickerComponent {
  public time?: Date
  user?: User | any;
  ngOnInit() {

```

```

        this.user = new User();
    }
    onSubmit(userForm:any) {
        (userForm.valid) ? alert("submitted"): alert("form is invalid");
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

TEMPLATE.HTML

```

<div>
  <form #userForm="ngForm" (ngSubmit)="onSubmit(userForm)">
    <div class="form-group">
      <ejs-timepicker id="timepicker" name="time" [(ngModel)]="user.time"
#time="ngModel" width="300px" required></ejs-timepicker>
      <div *ngIf="userForm.invalid && (userForm.dirty ||
userForm.touched)" class="alert alert-danger formerror">
        Valid time is required
      </div>
      <div *ngIf="userForm.valid && (userForm.dirty || userForm.touched)"
class="alert alert-success formerror">
        <table>
          <tr>
            <td style="width:50%">Selected value: </td>
            <td class="formtext ">{{ user.time | date:"hh:mm a"}}</td>
          </tr>
        </table>
      </div>
    </div>
    <div class="buttons">
      <button type="submit" class="e-btn e-success">Submit</button>
      <button type="reset" class="e-btn e-warning">Reset</button>
    </div>
  </form>
  <div class="dataclass">userForm.value: {{userForm.value | json}}</div>
  <div class="dataclass">userForm.valid: {{userForm.valid}}</div>
</div>
<style>
  form {
    margin-top: 50px;
  }
</style>

```

[Ej1 api migration in Angular Timepicker component](#)

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

Time Selection

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	Property: value	Property: value

Time Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display time format	Property: timeFormat	Property: format

Time Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum time	Property: minTime	Property: min
Maximum time	Property: maxTime	Property: Max
Set current time	Method: setCurrentTime() public onCreate(e:any) { var timeObj = \$('#timePicker').data("ejTimePicker"); timeObj.setCurrentTime();}	Can be achieved by public value: Date = new Date();

Disabled Time Ranges

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disable time ranges	Property: disableTimeRanges disableTime: Object = [{ startTime: '3:00 AM', endTime: '6:00 AM' }, { startTime: '1:00 PM', endTime: '3:00 PM' }, { startTime: '8:00 PM', endTime: '10:00 PM' }];	Can be achieved by public itemRenderHandler(args: ItemEventArgs): void { if (args.value.getHours() === 4) { args.isDisabled = true; }}

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	Property: CssClass	Property: CssClass
Popup list customization	Not Applicable	Event: itemRenderpublic itemRenderHandler(args: ItemEventArgs): void { / code block */ }
Show/Hide the popup button	Property: showPopupButton	Can be achieved by@ViewChild("timeObj") timeObj: TimePickerComponent;public onFocus(args:any):void { this.timeObj.show();}css.e- control-wrapper .e-input- group-icon.e-time-icon { display: none;}
Enable/Disable the rounded corner	Property: showRoundedCorner	Can be achieved bycss.e-control- wrapper.e-custom-style.e-time- wrapper.e-input-group { border-radius: 4px;}
Enable/Disable the animation	Property: enableAnimation	Not Applicable
Interval	Property: interval	Property: step
FocusIn event	Event: focusInpublic onFocus(e:any) { / code block */ }	Event: focus@ViewChild("timeObj") timeObj: TimePickerComponent;public onFocus(args:any):void { / code block */ }
FocusOut event	Event: focusOutpublic onFocus(e:any) { / code block */ }	Event: blur@ViewChild("timeObj") timeObj: TimePickerComponent;public onBlur(args:any):void { / code block */ }
FocusIn method	Not Applicable	Method: focusIn()@ViewChild("timeObj") timeObj: TimePickerComponent;public create(args:any):void { this.timeObj.focusIn();}

FocusOut method	Not Applicable	Event: focusOut@ViewChild("timeObj") timeObj: TimePickerComponent;public create(args:any):void { this.timeObj.focusOut();}
Prevent popup close	Not Applicable	Event: close@ViewChild("timeObj") timeObj: TimePickerComponent;public onClose(args:any): void { args.cancel = true;}
Prevent popup open	Not Applicable	Event: open@ViewChild("timeObj") timeObj: TimePickerComponent;public onOpen(args:any): void { args.cancel = true;}

Accessibility

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the RTL	Property: enableRTL	Property: enableRtl

Persistence

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Enable/Disable the persistence	Property: enablePersistence	Property: enablePersistence

Validation

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Validation rules	Property: validationRules:validationRules:Object;constructor(){ this.validationRules = {required:true};}	Can be achieved by let options: FormValidatorModel = { rules: { 'timepicker': { required: [true, "Value is required"] } } }; this.formObject = new FormValidator('#form-element', options);
Validation message	Property: validationMessages:validationRules:Object;validationMessages:Object;constructor(){ this.validationMessages = {required: "Required Time value"}; this.validationRules = {required:true};}	Can be achieved by let options: FormValidatorModel = { rules: { 'timepicker': { required: [true, "Value is required"] } }, customPlacement: (inputElement: HTMLElement, errorElement: HTMLElement) => { inputElement.parentElement.parentElement.appendChild(errorElement); } }; this.formObject = new FormValidator('#form-element', options);

Common

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Width	Property: width	Property: width
Readonly	Property: readOnly	Property: Readonly
Show/Hide the clear button	Not Applicable	Property: showClearButton
Height	Property: Height	Can be achieved bycss.e-control-wrapper.e-custom-style.e-time-wrapper.e-input-group { height: 35px;}
Html Attributes	Property: HtmlAttributeshtmlAttributes: Object = {required:"required"}	Not Applicable
Watermark text	Property: watermarkText	Property: placeholder
Enable the TimePicker	Property: enabled	Property: enabled
Disable the TimePicker	Method: disable() public onCreate(e:any) { var timeObject = \$("#timePicker").data("ejTimePicker"); timeObject.disable();}	Property: enabled
Enable/Disable the textBox editing	Not Applicable	Property: allowEdit
zIndex	Not Applicable	Property: zIndex
Specify the placeholder text behavior	Not Applicable	Property: floatLabelType

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	Property: locale	Property: locale

Strict Mode

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Strict mode	Property: enableStrictMode	Property: strictMode

Open and Close

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Close	Event: close <code>public onClose(e:any) {/ code block */ }</code>	Event: close@ViewChild("timeObj") timeObj: TimePickerComponent;public onClose(args:any): void {/ code block */ }
Open	Event: open <code>public onOpen(e:any) {/ code block */ }</code>	Event: open@ViewChild("timeObj") timeObj: TimePickerComponent;public onOpen(args:any): void {/ code block */ }
Hide	Method: hide() <code>public onCreate(e:any) { var timeObject = \$(("#timePicker").data("ejTimePicker")); timeObject.show(); timeObject.hide(); }</code>	Method: hide()@ViewChild("timeObj") timeObj: TimePickerComponent;public create(args:any): void { this.timeObj.show(); this.timeObj.hide(); }
Show	Method: show() <code>public onCreate(e:any) { var timeObject = \$(("#timePicker").data("ejTimePicker")); timeObject.show(); }</code>	Method: show()@ViewChild("timeObj") timeObj: TimePickerComponent;public create(args:any): void { this.timeObj.show(); }

Toast

Getting started with Angular Toast component

This section briefly explains about how to create a simple **Toast** and configure its available functionalities in Angular using Angular quickstart.

Getting Started with Angular CLI

The following section explains the steps required to create a simple `angular-cli` application and how to configure **Toast** component.

Prerequisites

To get started with Syncfusion Angular UI Components, make sure that you have compatible versions of Angular and TypeScript.

- Angular : 6+
- TypeScript : 2.6+

Setting up an Angular project

Angular provides an easiest way to setup project using Angular CLI [Angular CLI](#) tool.

Install the CLI application globally in your machine.

```
`javascript
npm install -g @angular/cli
`
```

Create a new application

```
`javascript
```

```
ng new syncfusion-angular-app
```

```
,
```

Once you have executed the above command you may ask for following options,

- Would you like to add Angular routing?
- Which stylesheet format would you like to use?

By default it install the CSS style base application. To setup with SCSS, pass `--style=SCSS` argument on create project.

Example code snippet.

```
`javascript
```

```
ng new syncfusion-angular-app --style=SCSS
```

```
,
```

Navigate to the created project folder.

```
`javascript
```

```
cd syncfusion-angular-app
```

```
,
```

Installing Syncfusion notifications Package

Syncfusion packages are distributed in npm as `@syncfusion` scoped packages. You can get all the Angular Syncfusion package from npm [link](#).

Currently, Syncfusion provides two types of package structures for Angular components,

1. Ivy library distribution package [format](#)
2. Angular compatibility compiler(Angular's legacy compilation and rendering pipeline) package.

Ivy library distribution package

Syncfusion Angular packages(`>=20.2.36`) has been moved to the Ivy distribution to support the Angular [Ivy](#) rendering engine and the package are compatible with Angular version 12 and above. To download the package use the below command.

Add [@syncfusion/ej2-angular-notifications](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-notifications --save
```

```
,
```

Angular compatibility compiled package(`ngcc`)

For Angular version below 12, you can use the legacy (`ngcc`) package of the Syncfusion Angular components. To download the `ngcc` package use the below.

Add [@syncfusion/ej2-angular-notifications@ngcc](#) package to the application.

```
`bash
```

```
npm install @syncfusion/ej2-angular-notifications@ngcc --save
```

```
,
```

To mention the ngcc package in the `package.json` file, add the suffix `-ngcc` with the package version as below.

```
`bash
```

```
@syncfusion/ej2-angular-notifications:"20.2.38-ngcc"
```

```
,
```

Note: If the ngcc tag is not specified while installing the package, the Ivy Library Package will be installed and this package will throw a warning.

Adding Toast module

After installing the package, the component modules are ready to configure in your application from installed syncfusion package. Syncfusion Angular package provides two different types of ngModules.

Refer to [Ng-Module](#) to learn about ngModules.

Refer the following snippet to import the `ToastModule` in `app.module.ts` from the `@syncfusion/ej2-angular-notifications`.

```
`javascript
```

```
import { BrowserModule } from '@angular/platform-browser';
```

```
import { NgModule } from '@angular/core';
```

```
import { AppRoutingModule } from './app-routing.module';
```

```
import { AppComponent } from './app.component';
```

```
// Imported syncfusion ToastModule from ej2-angular-notifications package
```

```
import { ToastModule } from '@syncfusion/ej2-angular-notifications';
```

```
@NgModule({
```

```
  declarations: [
```

```
    AppComponent
```

```
  ],
```

```
  imports: [
```

```
    BrowserModule,
```

```
    AppRoutingModule,
```

```
    // Registering EJ2 Toast Module
```

```
    ToastModule
```

```
  ],
```

```
  providers: [],
```

```
  bootstrap: [AppComponent]
```

```
})  
export class AppModule { }  
`
```

Adding Toast component

Add the Toast component snippet in `app.component.ts` as follows.

```
`typescript  
import { Component, ViewChild } from '@angular/core';  
import { ToastComponent } from '@syncfusion/ej2-angular-notifications';  
@Component({  
  selector: 'app-root',  
  template: `<ejs-toast #element (created)="onCreate($event)">  
    <ng-template #title>  
      <div>Sample Toast Title</div>  
    </ng-template>  
    <ng-template #content>  
      <div>Sample Toast Content</div>  
    </ng-template>  
  </ejs-toast>`  
})  
export class AppComponent {  
  @ViewChild('element') element: ToastComponent;  
  onCreate() {  
    this.element.show();  
  }  
  constructor() {  
  }  
}  
`
```

Adding CSS reference

The following CSS files are available in `../node_modules/@syncfusion` package folder. This can be referenced in `[src/styles.css]` using following code.

```
`css  
@import '../node_modules/@syncfusion/ej2-base/styles/material.css';  
@import '../node_modules/@syncfusion/ej2-buttons/styles/material.css';
```

```
@import '../node_modules/@syncfusion/ej2-popups/styles/material.css';
@import '../node_modules/@syncfusion/ej2-angular-notifications/styles/material.css';
`
```

The [Custom Resource Generator \(CRG\)](#) is an online web tool, which can be used to generate the custom script and styles for a set of specific components.

This web tool is useful to combine the required component scripts and styles in a single file.

Initialize the Toast with message

The Toast message can be rendered by defining an `title` or `content`.

Running the application

Run the `ng serve` command in command window, it will serve your application and you can open the browser window. Output will appear as follows.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)"
    [position]='position'>
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
        hoverboards?</div>
      </ng-template>
    </ejs-toast>`
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right' };
  onCreate(args: any) {
    this.element.show();
  }
  btnClick(args: any) {
    this.element.show();
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Render different types of toast](#)

Config in Angular Toast component

This section explains on customizing the Toast appearance using built-in APIs.

Title and content template

Toast can be created with the notification message. The message contains [title](#) and [content](#) of the Toasts. Title and contents are adaptable in any resolution.

Title or Content property can be given as HTML element/element ID as a string that can be displayed as a Toast.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ejs-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)" [position] =
    'position' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
        hoverboards?</div>
      </ng-template>
    </ejs-toast>`
```



```

    })
    export class AppComponent {
      @ViewChild('element') public element: any;
      public position = { X: 'Right' };
      onCreate(args: any) {
        this.toastShow();
      }
      btnClick(args: any) {
        this.toastShow();
      }
      toastShow() {
        setTimeout(
          () => {
            this.element.show();
          }, 700);
      }
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Specifying custom target

By default toast can be rendered in the document body, we can change the target position for toast rendering using [target](#) property. Based on the target [position](#) will update.

Close Button

In default [showCloseButton](#) is not enabled. We can enable it by setting true value. Before expiring toast we can use to close or destroy toasts manually.

Progress bar

In default [showProgressBar](#) is not enabled. If we enabled it can visually indicate when will the toast gets expired. Based on the `timeOut` property Progress bar will appear.

Progress bar direction

By default, the [progressDirection](#) is set to "Rtl" and it will appear from right to left direction. You can change the progressDirection to "Ltr" to make it appear from left to right direction.

Newest on top

In default, newly created toasts will append next with existing toast. We can change the Sequence like inserting before the toast, by enabling the [newestOnTop](#).

Here below sample demonstrates the combination of `target`, `showCloseButton`, `showProgressBar` and `newestOnTop` properties in toast.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ToastModule } from '@syncfusion/ej2-angular-notifications';

```

```

import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)" showCloseButton=true
    newestOnTop=true showProgressBar=true progressDirection="Ltr"
    target='#toast_target' [position] = 'position' >
      <ng-template #title>
        <div>File Downloading</div>
      </ng-template>
      <ng-template #content>
        <div class="progress"><span style="width: 80%"></span></div>
      </ng-template>
    </ejs-toast>`
  })
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Center' };
  onCreate(args: any) {
    this.element.show();
  }
  btnClick(args: any) {
    this.element.show();
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Width and height

we can set toast dimensions through [width](#) and [height](#) property. This will individually set all toasts, we can create different custom dimension toasts.

In default toast can be rendered with '300px' width with 'auto' height

In mobile device toast default width gets '100%' width of the page.

When we sets toast width as '100%' toast will occupies full width and displayed top or bottom based on position **Y** property.

Both width and height property allows setting pixels/numbers/percentage. The number value is considered as pixels.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id=sample_container>
      <div id='container'>
        <div class='row' style="padding-top: 20px;margin:0" id=
"toast_pos_target">
          <table>
            <tr>
              <td>
                <div style='padding:25px 0 0 0;'>
                  <ejs-radiobutton (change)="topChange($event)"
label="Top" name="position" value="Top"></ejs-radiobutton>
                </div>
              </td>
              <td>
                <div style='padding:25px 0 0 0;'>
                  <ejs-radiobutton (change)="bottomChange($event)"
label="Bottom" name="position" value="Bottom" checked="true"></ejs-
radiobutton>
                </div>
              </td>
            </tr>
            <tr>
              <td>
                <div style='padding:25px 0 0 0;'>
                  <ejs-checkbox #checkbox label="100% Width"
(change)="checkBoxChange($event)"></ejs-checkbox>
                </div>
              </td>
            </tr>
          </table>
        </div></div>
        <button ej-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>
      </div>
      <ejs-toast #element (created)="onCreate($event)" [position] =
'position' >
```

```

    <ng-template #title>
      <div>Matt sent you a friend request</div>
    </ng-template>
    <ng-template #content>
      <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
    </ng-template>
  </ejs-toast>
  ))
export class AppComponent {
  @ViewChild('element') public toast: any;
  public position = { X: 'Center', Y: 'Bottom' };
  onCreate(args: any) {
    this.toastShow();
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.toast.show();
      }, 700);
  }
  bottomChange(e: any) {
    let toast = this.toast;
    if (e.event.target.checked) {
      toast.position.Y = "Bottom";
      toast.hide('All');
      this.toastShow();
    }
  }
  topChange(e: any) {
    let toast = this.toast;
    if (e.event.target.checked) {
      toast.position.Y = "Top";
      toast.hide('All');
      this.toastShow();
    }
  }
  checkBoxChange(e: any) {
    let toast = this.toast;
    if (e.checked) {
      toast.hide('All');
      toast.width = "100%";
      toast.title = "";
      toast.content = "<div class='e-custom'>Take a look at our next
generation <b>Javascript</b> <a
href='https://ej2.syncfusion.com/home/index.html' target='_blank'>LEARN
MORE</a></div>";
      this.toastShow();
    } else {
      toast.hide('All');
      toast.width = 300;
      toast.title = 'Matt sent you a friend request',

```

```

        toast.content = 'Hey, wanna dress up as wizards and ride our
        hoverboards?',
        this.toastShow();
    }
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Prevent duplicate toasts](#)
- [Customize the progress bar](#)

Position in Angular Toast component

Toast position can be updated based on predefined positions or user customizable positions. Predefined position combinations are updated in [X](#) and [Y](#) position properties.

Predefined

X Positions

- Left
- Center
- Right

Y Positions

- Top
- Bottom

In the case of multiple Toast display, new Toast position will not update on dynamic change of property values, until the old Toast messages removed.

Toast occupies full width when we set width as '100%', so X positions won't affect changes when '100%' width.

Custom

Custom [X](#) and [Y](#) Position can be given as pixels/numbers/percentage. The number value is considered as pixels. based value top and left value updated in the toast.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'

```

```

import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { DropDownList } from '@syncfusion/ej2-angular-dropdowns';
import { ToastComponent } from '@syncfusion/ej2-angular-notifications';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id='sample_container'>
      <div id='container'>
        <div class='row' style="padding-top: 20px" id= "toast_pos_target">
          <div id="toast_pos"> </div>
          <div id="toast_pos_property">
            <table style="width: 100%">
              <tr>
                <td>
                  <div style='padding:25px 0 0 0;'>
                    <ejs-radiobutton
                      (change)="positionChange($event)" label="Position" name="position"
                      value="position" checked="true"></ejs-radiobutton>
                  </div>
                </td>
                <td>
                  <div style='padding:25px 0 0 0;'>
                    <ejs-radiobutton
                      (change)="customePosition($event)" label="Custom" name="position"
                      value="position"></ejs-radiobutton>
                  </div>
                </td>
              </tr>
              <tr>
                <td id="dropdownChoose" colspan="2">
                  <div id="dropdown" style='padding-top:25px;'>
                    <ejs-dropdownlist #dropDown (change)="dropDownChange($event)"
                      [dataSource]='dropdownDB' index='4'>
                      </ejs-dropdownlist>
                    </div>
                  </td>
                <tr>
                  <td colspan="2" id="customChoose" style="display:
none">
                    <form id="formId" class="form-horizontal">
                      <div class='e-row'>
                        <div class="e-float-input">
                          <input class="e-input" id="xPos"
name="Digits" value=50 digits="true" data-digits-message="Please enter digits
only." required="">
                        <span class="e-float-line"></span>

```

```

        <label class="e-float-text"
for="Digits">X Position</label>
        </div>
    </div>
    <div class='e-row'>
        <div class="e-float-input">
            <input class="e-input" id="yPos"
name="Digits" value=50 digits="true" data-digits-message="Please enter digits
only." required="">
            <span class="e-float-line"></span>
            <label class="e-float-text"
for="Digits">Y Position</label>
        </div>
    </div>
</form>
</td>
</tr>
<tr>
    <td>
        <div style='padding:25px 0 0 0;'>
            <ejs-radiobutton
(change)="targetChange($event)" label="Target" name="target"
value="Bottom"></ejs-radiobutton>
        </div>
    </td>
    <td>
        <div style='padding:25px 0 0 0;'>
            <ejs-radiobutton
(change)="globalTargetChange($event)" label="Global" name="target"
value="Bottom" checked="true"></ejs-radiobutton>
        </div>
    </td>
</tr>
</table>
<div id="toast_btn" style="padding-top: 25px">
    <button ejs-button [isPrimary]="true"
(click)="btnClick($event)">Show Toast</button>
</div>
</div>
</div>
</div></div>
<ejs-toast #element (created)="onCreate($event)" [position] =
'position' >
    <ng-template #title>
        <div>Matt sent you a friend request</div>
    </ng-template>
    <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
    </ng-template>
</ejs-toast>
})
export class AppComponent {
    @ViewChild('element') toastObj?: ToastComponent;
    @ViewChild('dropDown') dropDownList?: DropDownList;
    public position = { X: 'Right', Y: 'Bottom' };
}

```

```

    public dropdownDB = ['Top Left', 'Top Right', 'Top Center', 'Bottom
Left', 'Bottom Right', 'Bottom Center' ];
    public customFlag = false;
    onCreate(args: any) {
        this.toastShow();
    }
    btnClick(args: any) {
        if (this.customFlag) {
            this.setcustomPosValue();
        }
        this.toastObj?.hide('All');
        this.toastShow();
    }
    positionChange(e: any) {
        if (e.event.target.checked) {
            this.toastObj?.hide('All');
            (document.getElementById('dropdownChoose') as
HTMLElement).style.display = 'table-cell';
            (document.getElementById('customChoose') as HTMLElement
).style.display = 'none';
            this.setToastPosValue((this.dropDownList as
DropDownList).value.toString());
            this.customFlag = false;
            this.toastShow();
        }
    }
    customePosition(e: any) {
        if (e.event.target.checked) {
            this.toastObj?.hide('All');
            (document.getElementById('dropdownChoose') as HTMLElement
).style.display = 'none';
            (document.getElementById('customChoose') as
HTMLElement).style.display = 'table-cell';
            this.setcustomPosValue();
            this.customFlag = true;
            this.toastShow();
        }
    }
    dropDownChange(e: any) {
        this.toastObj?.hide('All');
        this.setToastPosValue(e.value);
        this.toastShow();
    }
    globalTargetChange(e: any) {
        if (e.event.target.checked) {
            this.toastObj?.hide('All');
            this.toastObj!.target = document.body;
            this.toastShow();
        }
    }
    targetChange(e: any) {
        if (e.event.target.checked) {
            this.toastObj?.hide('All');
            this.toastObj!.target = '#toast_pos_target';
            this.toastShow();
        }
    }
}

```



```

    setcustomPosValue(): void {
        this.toastObj!.position.X =
            parseInt((<any>document.getElementById('xPos')).value, 10);
        this.toastObj!.position.Y =
            parseInt((<any>document.getElementById('yPos')).value, 10);
    }
    setToastPosValue(value: string): void {
        value = value.toLowerCase().replace(' ', '');
        let toastObj: ToastComponent = this.toastObj as ToastComponent;
        switch (value) {
            case 'topleft':
                toastObj.position.X = 'Left'; toastObj.position.Y = 'Top'; break;
            case 'topright':
                toastObj.position.X = 'Right'; toastObj.position.Y = 'Top';
break;
            case 'topcenter':
                toastObj.position.X = 'Center'; toastObj.position.Y = 'Top';
break;
            case 'topfullwidth':
                toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Top'; break;
            case 'bottomleft':
                toastObj.position.X = 'Left'; toastObj.position.Y = 'Bottom';
break;
            case 'bottomright':
                toastObj.position.X = 'Right'; toastObj.position.Y = 'Bottom';
break;
            case 'bottomcenter':
                toastObj.position.X = 'Center'; toastObj.position.Y = 'Bottom';
break;
            case 'bottomfullwidth':
                toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Bottom'; break;
        }
    }
    toastShow() {
        setTimeout(
            () => {
                this.toastObj?.show();
            }, 700);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Render toast with different positions](#)

Action buttons in Angular Toast component

You can include action Buttons into toast by adding [buttons](#) property. You can bind collections of Essential JS 2 Button Model to `model` property inside buttons property, You can also include click event callback function, for each button.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { closest } from '@syncfusion/ej2-base';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast id='elementToastTime' #element
    (created)="onCreate($event)" width='230px' height='250px' title= 'Anjolie
    Stokes' [buttons]='buttons'
    content= '<p></p>' [position] = 'position' >
    </ejs-toast>`
  })
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: "Right", Y: "Bottom" };
  public buttons = [{ model: { content: "Ignore" }, click:
  this.btnToastClick.bind(this)}, {model: { content: "reply" }}];
  btnToastClick(e: any) {
    const toastEle = closest(e.target, '.e-toast');
    this.element.hide(toastEle);
  }
  onCreate(args: any) {
    this.toastShow();
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show();
      }, 700);
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

See Also

- [Configuring options](#)

Timeout in Angular Toast component

Toast can be expired based on [timeOut](#) property, toast will live till the timeOut reaches without user interaction, a timeOut value was considered as the millisecond.

- [timeOut](#) delay can be visually represented through [Progress Bar](#).
- [extendedTimeOut](#) property can make how long the toast will display after a user hovers over it.

You can terminate the process by using [showCloseButton](#) property for destroying toast at any time.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id='sample_container'>
      <div id='container'>
        <div class="e-float-input">
          <input class="e-input" id="toast_input_index" required=""
value="5000">
          <span class="e-float-line"></span>
          <label class="e-float-text">Enter timeOut</label>
        </div>
      </div>
      <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>
    </div>
    <ejs-toast #element (created)="onCreate($event)" [position] =
'position' >
      <ng-template #title>
```

```

        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
      </ng-template>
    </ejs-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right', Y: 'Bottom' };
  onCreate(args: any) {
    this.element.show();
  }
  btnClick(args: any) {
    const value = parseInt((document.getElementById('toast_input_index') as
HTMLInputElement).value, 10);
    this.element.show({timeOut: value});
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Static toast

We can prevent auto hiding in a toast as visible like static. For this, we need to set zero (0) value in timeOut Property.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id='sample_container'>
      <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>

```

```

        </div>
        <ejs-toast #element (created)="onCreate($event)" showCloseButton=true
timeOut = 0 [position] = 'position' >
            <ng-template #title>
                <div>Matt sent you a friend request</div>
            </ng-template>
            <ng-template #content>
                <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
            </ng-template>
        </ejs-toast>`
    })
    export class AppComponent {
        @ViewChild('element') public element: any;
        public position = { X: 'Right', Y: 'Top' };
        onCreate(args: any) {
            this.element.show();
        }
        btnClick(args: any) {
            this.element.show();
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Hide the toast on click](#)

Template in Angular Toast component

Template property can be given as the **HTML element** that is either a **string** or a **query selector**.

The HTML element tag can be given as a string for the template property.

`typescript

template: "<div>Toast Content</div>"

,

The template property also allows getting template content through **query selector**. Here, element 'ID' attribute is specified in the template.

`typescript

template: "#Template"

,

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { ChangeArgs } from '@syncfusion/ej2-angular-buttons';
import { DropDownListComponent } from '@syncfusion/ej2-angular-dropdowns';
import { ToastComponent } from '@syncfusion/ej2-angular-notifications';
@Component({
imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
],
standalone: true,
selector: 'app-root',
template: `
    <div id="toast_target"></div>
    <div id='sample_container'>
    <div id='container'>
        <div class='row' style="padding-top: 20px" id= "toast_pos_target">
            <div id="toast_pos"> </div>
            <div id="toast_pos_property">
                <table style="width: 100%">
                    <tr>
                        <td>
                            <div style='padding:25px 0 0 0;'>
                                <ejs-radiobutton
                                (change)="positionChange($event)" label="Position" name="position"
                                value="position" checked="true"></ejs-radiobutton>
                            </div>
                        </td>
                        <td>
                            <div style='padding:25px 0 0 0;'>
                                <ejs-radiobutton
                                (change)="customePosition($event)" label="Custom" name="position"
                                value="position"></ejs-radiobutton>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td id="dropdownChoose" colspan="2">
                            <div id="dropdown" style='padding-top:25px;'>
                                <ejs-dropdownlist #dropDown (change)="dropDownChange($event)"
                                [dataSource]='dropdownDB' index='4'>
                                    </ejs-dropdownlist>
                                </div>
                            </td>
                        <tr>
                            <td colspan="2" id="customChoose" style="display:
                                none">
                                    <form id="formId" class="form-horizontal">
                                        <div class='e-row'>
                                            <div class="e-float-input">

```

```

        <input class="e-input" id="xPos"
name="Digits" value=50 digits="true" data-digits-message="Please enter digits
only." required="">
        <span class="e-float-line"></span>
        <label class="e-float-text"
for="Digits">X Position</label>
    </div>
</div>
<div class='e-row'>
    <div class="e-float-input">
        <input class="e-input" id="yPos"
name="Digits" value=50 digits="true" data-digits-message="Please enter digits
only." required="">
        <span class="e-float-line"></span>
        <label class="e-float-text"
for="Digits">Y Position</label>
    </div>
</div>
</form>
</td>
</tr>
<tr>
    <td>
        <div style='padding:25px 0 0 0; '>
            <ejs-radiobutton
(change)="targetChange($event)" label="Target" name="target"
value="Bottom"></ejs-radiobutton>
        </div>
    </td>
    <td>
        <div style='padding:25px 0 0 0; '>
            <ejs-radiobutton
(change)="globalTargetChange($event)" label="Global" name="target"
value="Bottom" checked="true"></ejs-radiobutton>
        </div>
    </td>
</tr>
</table>
<div id="toast_btn" style="padding-top: 25px">
    <button ejs-button [isPrimary]="true"
(click)="btnClick()">Show Toast</button>
</div>
</div>
</div>
</div></div>
<ejs-toast #element (created)="onCreate()" [position] = 'position' >
    <ng-template #title>
        <div>Matt sent you a friend request</div>
    </ng-template>
    <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
    </ng-template>
</ejs-toast>
    ,
})
export class AppComponent {

```

```

@ViewChild('element') toastObj: any;
@ViewChild('dropDown') dropDownList : any;
public position = { X: 'Right', Y: 'Bottom' };
public dropdownDB = ['Top Left', 'Top Right', 'Top Center', 'Bottom
Left', 'Bottom Right', 'Bottom Center' ];
public customFlag = false;
onCreate() {
    this.toastShow();
}
btnClick() {
    if (this.customFlag) {
        this.setcustomPosValue();
    }
    this.toastObj.hide('All');
    this.toastShow();
}
positionChange(e: any) {
    if (e.event.target.checked) {
        this.toastObj.hide('All');
        (document.getElementById('dropdownChoose') as any).style.display =
'table-cell';
        (document.getElementById('customChoose') as any).style.display =
'none';
        this.setToastPosValue(this.dropDownList.value.toString());
        this.customFlag = false;
        this.toastShow();
    }
}
customePosition(e: ChangeArgs) {
    debugger;
    if (e.event && (e.event.target as any).checked) {
        this.toastObj.hide('All');
        (document.getElementById('dropdownChoose') as any).style.display =
'none';
        (document.getElementById('customChoose') as any).style.display =
'table-cell';
        this.setcustomPosValue();
        this.customFlag = true;
        this.toastShow();
    }
}
dropDownChange(e: ChangeArgs) {
    this.toastObj.hide('All');
    this.setToastPosValue(e.value as any);
    this.toastShow();
}
globalTargetChange(e: ChangeArgs) {
    if (e.event && (e.event.target as any).checked) {
        this.toastObj.hide('All');
        this.toastObj.target = document.body;
        this.toastShow();
    }
}
targetChange(e: ChangeArgs) {
    if (e.event && (e.event.target as any).checked) {
        this.toastObj.hide('All');
        this.toastObj.target = '#toast_pos_target';
    }
}

```



```

        this.toastShow();
    }
}
setcustomPosValue(): void {
    this.toastObj.position.X =
    parseInt((<any>document.getElementById('xPos')).value, 10);
    this.toastObj.position.Y =
    parseInt((<any>document.getElementById('yPos')).value, 10);
}
setToastPosValue(value: string): void {
    value = value.toLowerCase().replace(' ', '');
    let toastObj = this.toastObj;
    switch (value) {
        case 'topleft':
            toastObj.position.X = 'Left'; toastObj.position.Y = 'Top'; break;
        case 'topright':
            toastObj.position.X = 'Right'; toastObj.position.Y = 'Top';
break;
        case 'topcenter':
            toastObj.position.X = 'Center'; toastObj.position.Y = 'Top';
break;
        case 'topfullwidth':
            toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Top'; break;
        case 'bottomleft':
            toastObj.position.X = 'Left'; toastObj.position.Y = 'Bottom';
break;
        case 'bottomright':
            toastObj.position.X = 'Right'; toastObj.position.Y = 'Bottom';
break;
        case 'bottomcenter':
            toastObj.position.X = 'Center'; toastObj.position.Y = 'Bottom';
break;
        case 'bottomfullwidth':
            toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Bottom'; break;
    }
}
toastShow() {
    setTimeout(
        () => {
            this.toastObj.show();
        }, 700);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

See Also

- [Add template dynamically](#)

Animation in Angular Toast component

Toasts support custom animations for both shows and hide actions from the provided animation option of **Animation** library.

Default animation is given as **FadeIn** for showing the toast and **FadeOut** for hiding the toast.

The sample demonstrates some types of animation that suits Toast. You can check all the animation effects [here](#).

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id='sample_container'>
      <div id='container'>
        <div class='row' style="margin: 10px">
          <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
            <label> Show Animation </label>
          </div>
          <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
            <ejs-dropdownlist #dropDownShow
              (change)="showAnimationChange($event)" id='games'
              [dataSource]='AnimationShowDB' index='0'>
              </ejs-dropdownlist>
            </div>
          </div>
          <div class='row' style="margin: 10px">
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
              <label> Hide Animation </label>
            </div>
            <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
              <ejs-dropdownlist #dropDownHide
                (change)="hideAnimationChange($event)" id='games'
                [dataSource]='AnimationHideDB' index='0'>
                </ejs-dropdownlist>
              </div>
            <div class='row' style="margin: 10px">
```

```

        <button ejs-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>
      </div>
    </div></div>
    <ejs-toast #element (created)="onCreate($event)" [position] =
'position' [animation] = 'animation' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
      </ng-template>
    </ejs-toast>
  })
  export class AppComponent {
    @ViewChild('element') public toastObj: any;
    @ViewChild('dropDownShow') public dropShow: any;
    @ViewChild('dropDownHide') public drophide: any;
    public position = { X: 'Right', Y : 'Bottom' };
    public animation = { show: { effect: 'SlideRightIn' }, hide: { effect:
'SlideLeftOut' } };
  };
  public AnimationShowDB = ['FadeIn', 'FadeZoomIn', 'FadeZoomOut',
'FlipLeftDownIn', 'FlipLeftDownOut', 'FlipLeftUpIn', 'FlipLeftUpOut',
'FlipRightDownIn', 'FlipRightDownOut', 'SlideBottomIn', 'SlideBottomOut',
'ZoomIn', 'ZoomOut'];
  public AnimationHideDB = ['Fadeout', 'FadeZoomIn', 'FadeZoomOut',
'FlipLeftDownIn', 'FlipLeftDownOut', 'FlipLeftUpIn', 'FlipLeftUpOut',
'FlipRightDownIn', 'FlipRightDownOut', 'SlideBottomIn', 'SlideBottomOut',
'ZoomIn', 'ZoomOut'];
  onCreate(args: any) {
    this.toastShow();
  }
  btnClick(args: any) {
    this.toastShow();
  }
  showAnimationChange(e: any) {
    this.toastObj.animation.show.effect = this.dropShow.value;
  }
  hideAnimationChange(e: any) {
    this.toastObj.animation.hide.effect = this.drophide.value;
  }
  toastShow() {
    setTimeout(
      () => {
        this.toastObj.show();
      }, 700);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';

```

```
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Toast services in Angular Toast component

The Toast component provides a built-in utility function to render the toast with minimal code. The utility function will render the toast without the need of rendering the container element in the DOM where the toast is appended. So that, the toast can now be rendered on the go. The following are the option to render the toast using the utility function.

Show Toast with predefined types

The Toast component support 4 types of predefined toast with essential colors for various situations which can be shown using the `ToastUtility.show` by just defining the type of the toast without defining any class names. The following options are used as an argument on calling the utility function for predefined types:

Options	Description
content	Specifies the content that can be displayed on the Toast.
type	Specifies the type of the predefined Toasts. The 4 types of predefined toasts are Information , Success , Error , Warning
timeOut	Specifies the Toast display time duration on the page in milliseconds. Once the time expires, Toast message will be removed. Setting 0 as a time out value displays the Toast on the page until the user closes it manually.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { ToastComponent } from '@syncfusion/ej2-angular-notifications';
import { closest } from '@syncfusion/ej2-base';
import { ToastUtility } from '@syncfusion/ej2-notifications';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div>
    <button ej2-button cssClass='e-btn e-control e-info'
    (click)="infoToast($event)">Info Message</button>
    <button ej2-button cssClass='e-btn e-control e-success'
    (click)="successToast($event)">Success Message</button>
    <button ej2-button cssClass='e-btn e-control e-warning'
    (click)="warningToast($event)">Warning Message</button>
```

```

    <button ej-button cssClass='e-btn e-control e-danger'
    (click)="dangerToast($event)">Danger Message</button>
  </div>
  <br/>
  <div style="text-align: center;">
    <button ej-button cssClass='e-btn e-control'
    (click)="hideToast($event)">Hide All</button>
  </div>`
  })
  export class AppComponent {
    public toastObj?: ToastComponent;
    public infoToast(args: any): void {
      this.toastObj = ToastUtility.show('Please read the comments
carefully', 'Information', 20000) as ToastComponent;
    }
    public successToast(args: any): void {
      this.toastObj = ToastUtility.show('Your message has been sent
successfully', 'Success', 20000) as ToastComponent;
    }
    public warningToast(args: any): void {
      this.toastObj = ToastUtility.show('There was a problem with your
network connection', 'Warning', 20000) as ToastComponent;
    }
    public dangerToast(args: any): void {
      this.toastObj = ToastUtility.show('A problem has been occurred while
submitting the data', 'Error', 20000) as ToastComponent;
    }
    public hideToast(args: any): void {
      this.toastObj?.hide('All');
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show Toast with ToastModel

The utility function can be called using the [ToastModel](#) as argument to show the toast where all the properties in the [ToastModel](#) like any events, position, close icon, action buttons, etc. can be used in the [ToastUtility.show](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
import { ToastComponent } from '@syncfusion/ej2-angular-notifications';

```

```

import { closest } from '@syncfusion/ej2-base';
import { ToastUtility } from '@syncfusion/ej2-notifications';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `<div style="text-align: center;">
    <button ej2-button cssClass='e-btn e-control'
      (click)="showToast($event)">Show Toast</button>
    </div>`
})
export class AppComponent {
  public toastObj?: ToastComponent;
  public showToast(args: any): void {
    this.toastObj = ToastUtility.show({
      title: 'Toast Title',
      content: 'Toast shown using utility function with ToastModel',
      timeout: 20000,
      position: { X: 'Right', Y: 'Bottom' },
      showCloseButton: true,
      click: this.toastClick.bind(this),
      buttons: [{
        model: { content: 'Close' }, click:
this.toastClose.bind(this)
      }]
    }) as ToastComponent;
  }
  public toastClose(): void {
    this.toastObj?.hide();
  }
  public toastClick(): void {
    console.log('Toast click event is triggered');
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Style in Angular Toast component

The following content provides the exact CSS structure that can be used to modify the component's appearance based on the user preference.

Customizing the toast title

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

`CSS

/ To change color, font family and font size /

```
.e-toast-container .e-toast .e-toast-message .e-toast-title {  
  color: red;  
  font-size: 18px;  
  font-weight: bold;  
}  
`
```

Customizing the toast content

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

`CSS

/ To change color, font family and font size /

```
.e-toast-container .e-toast .e-toast-message .e-toast-content {  
  color: aqua;  
  font-size: 13px;  
  font-weight: normal;  
}  
`
```

Customizing the toast icon

Use the following CSS to customize the default toast icon color.

`CSS

/ To change icon color /

```
.e-toast-container .e-toast .e-toast-icon {  
  color: yellow;  
}  
`
```

Customizing the toast background

Use the following CSS to customize the default toast's background color.

`CSS

/ To change background color /

```
.e-toast-container .e-toast {  
  background-color: navy;  
}  
`
```

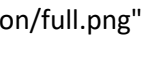
Accessibility in Angular Toast component

The Toast 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 Toast component is outlined below.

| Accessibility Criteria | Compatibility |

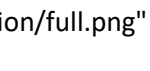
| -- | -- |

| [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 Toast component 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.

It helps to provides information about the elements in a document for assistive technology.

The component implements the keyboard navigation support by following the [WAI-ARIA practices](#) and tested in major screen readers.

ARIA attributes

<!-- markdownlint-disable MD033 -->

Class	Description
role	alert: Identifies the element as the container where alert content will be added or updated.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)" [position] =
    'position' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
        hoverboards?</div>
      </ng-template>
    </ejs-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right' };
  onCreate(args: any) {
    this.toastShow();
  }
}
```

```

    }
    btnClick(args: any) {
        this.toastShow();
    }
    toastShow() {
        setTimeout(() => {
            this.element.show();
        }, 700);
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Ensuring accessibility

The Toast component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Toast component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Toast component with accessibility tools.

See also

- [Accessibility in Syncfusion Angular components](#)

How To

Prevent duplicate toast display in Angular Toast component

You can prevent identical same toast displaying in a screen by event function. You can terminate the toast displaying process by setting cancel event property in [beforeOpen](#) Event.

Here below sample demonstrates preventing duplicate title Toast element displaying, with custom code blocks.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',

```

```

template: `
  <div id="toast_target"></div>
  <button ej-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>
  <ejs-toast #element (created)="onCreate($event)"
(beforeOpen)="onBeforeOpen($event)" (close)="onClose($event)"
[position]='position' > </ejs-toast>
`
))
export class AppComponent {
  @ViewChild('element') public element?: any;
  public position = { X: 'Center' };
  public prevDuplicates: boolean = false;
  public toastFlag: number = 0;
  public toasts = [
    { title: 'Warning !', content: 'There was a problem with your network
connection.', isOpen: false },
    { title: 'Success !', content: 'Your message has been sent
successfully.', isOpen: false },
    { title: 'Error !', content: 'A problem has been occurred while
submitting your data.', isOpen: false }
  ];
  onBeforeOpen(e: any) {
    if(this.preventDuplicate(e)){
      e.cancel = true;
    };
  }
  preventDuplicate(e: any) {
    for (let i: number = 0; i < this.toasts.length; i++) {
      if (this.toasts[i].title === e.options.title && !this.toasts[i].isOpen)
    {
      this.toasts[i].isOpen = true;
      return false;
    }
  }
  return true;
}
  onClose(e: any) {
    for (let i: number = 0; i < this.toasts.length; i++) {
      if (this.toasts[i].title === e.options.title) {
        this.toasts[i].isOpen = false;
      }
    }
  }
  onCreate(args: any) {
    this.element.show(this.toasts[this.toastFlag]);
    ++this.toastFlag;
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show(this.toasts[this.toastFlag]);
        ++this.toastFlag;
        if (this.toastFlag === (this.toasts.length)) {

```

```

        this.toastFlag = 0;
    }
    }, 0);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Restrict the maximum toast to show in Angular Toast component

You can restrict the maximum toast count by event callback function. You can terminate the toast displaying process by setting cancel event property in [beforeOpen](#) Event.

Here below sample demonstrates restrict toast displaying up to 3. You can restrict by your own count with custom code blocks.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ejs-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)"
    (beforeOpen)="onBeforeOpen($event)" [position] = 'position' > </ejs-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right', Y: 'Bottom' };
  public toasts = [
    { title: 'Warning !', content: 'There was a problem with your network
    connection.' },
    { title: 'Success !', content: 'Your message has been sent successfully.'
    },
    { title: 'Error !', content: 'A problem has been occurred while
    submitting your data.' },
  ],

```

```

    { title: 'Information !', content: 'Please read the comments carefully.'
  }];
  public maxCount: number = 3;
  public toastFlag: number = 0;
  onBeforeOpen(e: any) {
    if (this.maxCount === this.element.element.childElementCount) {
      e.cancel = true;
    } else {
      e.cancel = false;
    }
  }
  onCreate(args: any) {
    this.element.show(this.toasts[this.toastFlag]);
    ++this.toastFlag;
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show(this.toasts[this.toastFlag]);
        ++this.toastFlag;
        if (this.toastFlag === (this.toasts.length)) {
          this.toastFlag = 0;
        }
      }, 0);
  }
  ngAfterViewInit() {
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Customize progress bar theme and sizing in Angular Toast component

In default, the Progress bar will appear based on the theme stylings and dimensions. You can customize progress bar stylings through custom CSS or Event functions.

Here below sample demonstrates customize the progress bar Stylings using [beforeOpen](#) Event.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({

```

```

imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
],
standalone: true,
selector: 'app-root',
template: `
    <div id="toast_target"></div>
    <div id='#sample_container '>
    <div id='container'>
        <button ejb-button [isPrimary]="true" (click)="btnClick($event)">Show
Toast</button>
        <div class="row" style="padding-top: 20px">
        <div class="e-float-input">
            <input class="e-input" id="progressHeight" name="Digits"
value="4" required>
            <span class="e-float-line"></span>
            <label class="e-float-text" for="Digits">Progress Bar
Height</label>
        </div>
        </div>
        <div class="row" style="padding-top: 20px">
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
            <label> Progress Bar Color </label>
        </div>
        <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
            <ejb-dropdownlist #dropDown (change)="dropDownChange($event)"
id='games' #sample1 [dataSource]='progressBarColor' index='0'>
            </ejb-dropdownlist>
        </div>
        </div>
        </div>
        </div>
        <ejb-toast #element showProgressBar=true
(created)="onCreate($event)" (beforeOpen)="onBeforeOpen($event)" [position]
= 'position' >
            <ng-template #title>
                <div>Matt sent you a friend request</div>
            </ng-template>
            <ng-template #content>
                <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
            </ng-template>
        </ejb-toast>
    `
}))
export class AppComponent {
    @ViewChild('element') public element: any;
    @ViewChild('dropDown') public dropDownListObj: any;
    public position = { X: 'Right', Y: 'Bottom' };
    public progressBarColor = ['Red', 'Cyan', 'Blue', 'Yellow', 'Pink'];
    onBeforeOpen(e: any) {
        const progress = e.element.querySelector('.e-toast-progress');
        progress.style.height = (document.getElementById('progressHeight') as
any).value + 'px';
        progress.style.backgroundColor = this.dropDownListObj.value;
    }
}

```

```

dropDownChange(e: any) {
    const progressEles: NodeList = this.element.element.querySelectorAll('.e-toast-progress');
    progressEles.forEach((ele: HTMLElement | any) => {
        ele.style.backgroundColor = this.dropDownListObj.value;
    });
}
onCreate(args: any) {
    this.element.show();
}
btnClick(args: any) {
    this.toastShow();
}
toastShow() {
    setTimeout(
        () => {
            this.element.show();
        }, 0);
}
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Play an audio before open the toast in Angular Toast component

Here below sample demonstrates to playing audio background while opening toast. Here we have included audio play codes into beforeOpen event Function.

If you want to stop the audio after displaying toast use [open](#) event in Toast. please check the Toast Events [api's](#) for further customization.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
  `
})

```

```

    <button ejs-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (beforeOpen)="onBeforeOpen($event)" [position]
    = 'position' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
        hoverboards?</div>
      </ng-template>
    </ejs-toast>

  })
  export class AppComponent {
    @ViewChild('element') public element: any;
    public position = { X: 'Right', Y: 'Bottom' };
    onBeforeOpen(e: any) {
      let audio: HTMLAudioElement = new
      Audio('https://drive.google.com/uc?export=download&id=1M95VOpto1cQ4FQHzNBaLf0
      WFQglrtWi7');
      audio.play();
    }
    btnClick(args: any) {
      this.toastShow();
    }
    toastShow() {
      setTimeout(() => {
        this.element.show();
      }, 0);
    }
    ngAfterViewInit() {
    }
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Show different types of toast in Angular Toast component

The Essential JS 2 Toast has the following predefined styles that can be defined using the [cssClass](#) property for achieving different types of toast.

Class	Description
e-success	Used to represent a positive Toast.
e-info	Used to represent an informative Toast.
e-warning	Used to represent a Toast with caution.

| e-danger | Used to represent a negative Toast. |

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)" [position] =
    'position' > </ejs-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right', Y: 'Bottom' };
  public toasts = [
    { title: 'Warning !', content: 'There was a problem with your network
    connection.', cssClass: 'e-toast-warning' },
    { title: 'Success !', content: 'Your message has been sent
    successfully.', cssClass: 'e-toast-success' },
    { title: 'Error !', content: 'A problem has been occurred while
    submitting your data.', cssClass: 'e-toast-danger' },
    { title: 'Information !', content: 'Please read the comments carefully.',
    cssClass: 'e-toast-info' } ];
  public toastFlag: number = 0;
  onCreate(args: any) {
    this.element.show(this.toasts[this.toastFlag]);
    ++this.toastFlag;
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show(this.toasts[this.toastFlag]);
        ++this.toastFlag;
        if (this.toastFlag === (this.toasts.length)) {
          this.toastFlag = 0;
        }
      }, 0);
  }
}
```

```
}

```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```

Show multiple toasts in various positions in Angular Toast component

In default Toast position only updates once visible toasts get destroyed. If You needs to display multiple toasts with different position means needs to initiate another toast for achieving this.

Here below sample demonstrates to add multiple toasts adding in the different position.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div style="display:inline-block">
      <div style="margin-right:10px;display:inline-block">
        <button ejs-button (click)="btnRightClick($event)">Show Left
Position Toast</button>
      </div>
      <div style="display:inline-block">
        <button ejs-button (click)="btnClick($event)" >Show Right Position
Toast</button>
      </div>
    </div>
    <ejs-toast #element (created)="onCreate($event)" [position] =
'position' >
      <ng-template #title>
        <div>Warning !</div>
      </ng-template>
      <ng-template #content>
        <div>There was a problem with your network
connection.</div>
      </ng-template>
    </ejs-toast>
    <ejs-toast #rightToast (created)="onCreatel($event)" [position] =
'position1' >
```

```

        <ng-template #title>
            <div>Success !</div>
        </ng-template>
        <ng-template #content>
            <div>Your message has been sent successfully.</div>
        </ng-template>
    </ejs-toast>
    ,
    ))
    export class AppComponent {
        @ViewChild('element') public element: any;
        @ViewChild('rightToast') rightToast: any;
        public position = { X: 'Left', Y: 'Bottom' };
        public position1 = { X: 'Right', Y: 'Bottom' };
        onCreate1(args: any) {
            this.rightToast.show();
        }
        onCreate(args: any) {
            this.element.show();
        }
        btnRightClick(args: any) {
            this.element.show();
        }
        btnClick(args: any) {
            this.rightToast.show();
        }
    }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Close the toast with click tap in Angular Toast component

In default, toast gets expired based on timeOut value. You can customize toast hiding process, You can customize only hide with click/tap action by setting event args in [clicked](#) callback function with [static Toast](#).

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
    imports: [
        ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
        DropDownListModule, DatePickerModule
    ],

```

```

standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button (click)="btnClick($event)" >Show
Toast</button>
    <ejs-toast #element (created)="onCreate($event)" (click)=
"onClick($event)" timeOut='0' [position] = 'position' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
hoverboards?</div>
      </ng-template>
    </ejs-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Left', Y: 'Bottom' };
  onClick(e: any) {
    e.clickToClose = true;
  }
  onCreate(args: any) {
    this.element.show();
  }
  btnClick(args: any) {
    this.element.show();
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Add dynamic template in Angular Toast component

Toast can support to change templates in dynamically, with displaying in multiple toasts. We can change Toast properties while calling in [show](#) method.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [

```

```

    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <div id='templateToast' style="display: none;color:red"> System
    affected by virus !!! </div>
    <button ejb-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejb-toast #element (created)="onCreate($event)" [position] =
    'position' > </ejb-toast>
  `
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right', Y: 'Bottom' };
  public toasts = [ { template: '2 Mail has received' }, { template: 'User
  Guest Logged in' }, { template: 'Logging in as Guest' }, { template: 'Ticket has
  reserved' }, { template: '#templateToast' } ]
  public toastFlag: number = 0;
  onCreate(args: any) {
    this.element.show(this.toasts[this.toastFlag]);
    ++this.toastFlag;
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show(this.toasts[this.toastFlag]);
        ++this.toastFlag;
        if (this.toastFlag === (this.toasts.length)) {
          this.toastFlag = 0;
        }
      }, 0);
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Render template in toast using angular template in Angular Toast component

You can also render the template support in Toast using the Angular **ng-template**. We need to use this ng-template within the e-toast tag with #template attribute, which is mandatory to render template property. Also you don't need to use the template property if you used this ng-template.

APP.COMPONENT.TS

```

import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule , RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule , RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #element (created)="onCreate($event)" [position] =
    'position' >
      <ng-template #template>
        <div id="template_toast_ele">
          <ejs-datepicker></ejs-datepicker>
        </div>
      </ng-template>
    </ejs-toast>`
})
export class AppComponent {
  @ViewChild('element') public element: any;
  public position = { X: 'Right', Y: 'Bottom' };
  onCreate(args: any) {
    setTimeout(
      () => {
        this.element.show();
      }, 200);
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    this.element.show();
  }
}

```

MAIN.TS

```

import { bootstrapApplication } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import 'zone.js';
bootstrapApplication(AppComponent).catch((err) => console.error(err));

```

Prevent toast close with mobile swipe in Angular Toast component

You can prevent the toast close with mobile swipe action by setting [beforeClose](#) argument cancel value to true while argument type as a swipe. The following code shows how to prevent toast close with mobile swipe.

The following sample demonstrates preventing toast close with mobile swipe element displaying with custom code blocks.

APP.COMPONENT.TS

```
import { NgModule } from '@angular/core'
import { BrowserModule } from '@angular/platform-browser'
import { ToastModule } from '@syncfusion/ej2-angular-notifications'
import { ButtonModule, CheckBoxModule, RadioButtonModule } from
 '@syncfusion/ej2-angular-buttons'
import { DropDownListModule } from '@syncfusion/ej2-angular-dropdowns'
import { DatePickerModule } from '@syncfusion/ej2-angular-calendars'
import { Component, ViewChild } from '@angular/core';
@Component({
  imports: [
    ToastModule, ButtonModule, CheckBoxModule, RadioButtonModule,
    DropDownListModule, DatePickerModule
  ],
  standalone: true,
  selector: 'app-root',
  template: `
    <div id="toast_target"></div>
    <button ej2-button [isPrimary]="true" (click)="btnClick($event)">Show
    Toast</button>
    <ejs-toast #toastObj (created)="onCreate($event)"
    (beforeClose)="onBeforeClose($event)" [position] = 'position' >
      <ng-template #title>
        <div>Matt sent you a friend request</div>
      </ng-template>
      <ng-template #content>
        <div>Hey, wanna dress up as wizards and ride our
        hoverboards?</div>
      </ng-template>
    </ejs-toast>`
})
export class AppComponent {
  @ViewChild('toastObj') element: any;
  public position = { X: 'Right' };
  onCreate(args: any) {
    this.toastShow();
  }
  btnClick(args: any) {
    this.toastShow();
  }
  toastShow() {
    setTimeout(
      () => {
        this.element.show();
      }, 700);
  }
  onBeforeClose(e: any) {
    if (e.type === "swipe") {
      e.cancel = true;
    }
  }
}
```

MAIN.TS

```
import { bootstrapApplication } from '@angular/platform-browser';  
import { AppComponent } from './app.component';  
import 'zone.js';  
bootstrapApplication(AppComponent).catch((err) => console.error(err));
```