

USER GUIDE

Essential Studio for EJ2 React

Version - v25.1.35 | Release Date - March 15, 2024

DateTimePicker	30
Getting started	30
Dependencies.....	30
Installation and configuration.....	30
Adding Syncfusion packages	30
Adding Style sheet to the Application.....	31
Adding DateTimePicker component to the Application	31
Run the application	32
Setting the min and max	33
See Also	34
Globalization in React Datetimepicker component.....	35
Right-To-Left	552
Strict mode in React Datetimepicker component	1130
Date time range in React Datetimepicker component	1134
Date time masking in React Datetimepicker component	1135
Configure Mask Placeholder	1138
Customization in React Datetimepicker component.....	1141
Day and Time Cell format.....	1141
Adding mandatory asterisk to placeholder and float label.....	1143
See Also	1144
Accessibility in React Datetimepicker component	1144
WAI-ARIA attributes.....	1145
Keyboard Interaction	1145
Ensuring accessibility	1149
See also	1149
Style appearance in React Datetimepicker component	1149
Customizing the appearance of DateTimePicker wrapper element.....	1149
Customizing the DateTimePicker icons element	1149
Customizing the time picker popup in the DateTimePicker	1149
Customizing the Calendar popup of the DateTimePicker	1150
Full screen mode support in mobiles and tablets.....	1150
How To	1152
Disable the datetimepicker component in React Datetimepicker component	1152
Customize the datetimepicker day header in React Datetimepicker component	1152
Set the placeholder in React Datetimepicker component.....	1154

Ej1 api migration in React Datetimepicker component.....	1155
DateTime Selection	1155
DateTime Format	1155
Calendar Views.....	1155
Date Range.....	1155
Disabled Dates	1156
Customization	1156
Accessibility.....	1158
Persistence	1158
Validation	1158
Common.....	1159
Globalization	1161
Strict Mode	1161
Open and Close	1161
View Navigation	1162
Diagram.....	1162
Getting Started.....	1162
Getting started.....	1162
Creating a Next.js Application Using Syncfusion React Components	1180
Getting Started with the Diagram Component in the Preact Framework.....	1183
Nodes in React Diagram component	1189
Create node.....	1189
Add node through nodes collection.....	1190
Add/Remove node at runtime	1191
Add node from palette.....	1192
Create node through data source.....	1193
Draw nodes	1193
Position	1193
Flip.....	1195
Appearance	1197
Customize the style of main node on multi-selection.	1198
Gradient	1199
Linear gradient.....	1199
Radial gradient	1201
Shadow.....	1203

Customizing shadow	1204
Icon.....	1206
Customizing expand icon	1208
Customizing collapse icon	1209
Interaction.....	1209
Constraints	1209
Custom properties	1209
Stack order	1209
Data flow	1209
See Also	1212
Shapes in React Diagram component.....	1212
Text	1212
Image.....	1214
Image alignment	1218
HTML.....	1222
HTML Node With Template	1224
Native	1225
SVG content alignment	1229
Basic shapes	1231
Path	1233
Flow Shapes	1234
Bpmn shapes in React Diagram component.....	1236
Event	1244
Gateway	1299
Activity	1308
Tasks.....	1310
Subprocess	1319
Event subprocess	1321
Transaction subprocess.....	1323
Process	1326
Loop.....	1326
Compensation	1334
Call.....	1337
Adhoc	1338
Boundary	1340

Data	1345
Datasource	1349
Artifact	1351
Text annotation.....	1351
Group	1353
BPMN flows.....	1355
Association	1355
Sequence.....	1357
Message	1359
UML diagram in React Diagram component.....	1361
UML Class Diagram	1361
Uml Class Diagram Shapes	1362
UML Class Relationships	1368
How to add UML child at runtime.....	1377
Adding UML Nodes in Symbol palette	1383
Editing	1388
UML Activity diagram.....	1388
UML Activity diagram Shapes	1388
Connectors in React Diagram component.....	1402
Create connector	1402
Add connectors through connectors collection.....	1402
Add connector at runtime.....	1403
Connectors from palette.....	1405
Draw connectors	1405
Update connector at runtime	1405
Connect nodes	1407
Connections with ports	1411
Segments.....	1419
Straight.....	1419
Orthogonal	1423
Avoid overlapping	1428
How to customize Orthogonal Segment Thumb Shape.....	1432
Bezier	1434
Avoid overlapping with bezier	1440
Decorator	1448

Padding	1451
Hit padding.....	1454
Flip.....	1455
Bridging	1457
Corner radius.....	1461
Appearance	1464
Segment appearance	1464
Decorator appearance	1467
Interaction.....	1469
Automatic line routing	1469
Constraints	1475
Custom properties	1477
Stack order	1477
Line Routing	1479
Enable Connector Splitting.....	1481
See Also	1483
Group in React Diagram component	1483
Create group	1483
Add group when initializing diagram	1483
Add group at runtime	1488
Add children To group at runtime	1490
Remove children from group at runtime.....	1490
Container.....	1492
Difference between a basic group and containers	1497
Interaction.....	1497
See Also	1498
Swim lane in React Diagram component.....	1498
Create a swimlane.....	1498
Lanes	1506
Phase.....	1527
Add swimlane to palette	1538
Limitations.....	1543
Labels in React Diagram component	1543
Create annotation	1543
Add annotations at runtime.....	1545

Remove annotation	1547
Update annotation at runtime.....	1549
Alignment.....	1550
Offset.....	1550
Horizontal and vertical alignment.....	1552
Annotation alignment with respect to segments	1563
Margin	1565
Text align	1567
Hyperlink	1568
Template Support for Annotation	1570
Wrapping.....	1572
Text overflow	1576
Appearance	1578
Interaction.....	1581
Edit	1583
Read-only annotations.....	1583
Drag Limit	1585
Multiple annotations	1586
Constraints	1588
Ports in React Diagram component	1588
Create port.....	1591
Add ports when initializing nodes.....	1591
Add ports at runtime.....	1593
Remove ports at runtime.....	1595
Update port at runtime.....	1598
Appearance	1600
Offset.....	1602
Constraints	1602
Constraints in React Diagram component	1602
Diagram constraints	1602
Node constraints	1603
Connector constraints.....	1604
Port constraints.....	1605
Annotation constraints	1605
Selector constraints	1606

Snap constraints.....	1607
Boundary constraints.....	1608
Inherit behaviors.....	1609
Bitwise operations	1610
Add operation	1610
Remove Operation.....	1610
Check operation.....	1610
Interaction in React Diagram component.....	1610
Selection.....	1610
Single selection	1610
Selecting a group.....	1611
Multiple selection	1611
Select/Unselect elements using program.....	1612
Select entire elements in diagram programmatically.....	1613
Drag.....	1613
Resize	1613
Customize the resize-thumb.....	1614
Rotate.....	1616
Connection editing.....	1617
End point handles	1617
Straight segment editing.....	1618
Orthogonal thumbs.....	1620
Bezier thumbs	1621
Drag and drop nodes over other elements.....	1622
User handles	1622
Alignment.....	1622
Offset.....	1623
Side.....	1623
Horizontal and vertical alignments	1623
Margin.....	1623
Notification for the mouse button clicked.....	1623
Appearance	1624
Zoom pan	1627
Zoom pan status.....	1628
Keyboard	1629

See Also	1631
Tools in React Diagram component	1631
Drawing tools	1631
Shapes	1631
Connectors	1635
Text	1636
Polyline Connector	1639
Tool selection	1641
Events	1644
Freehand Drawing	1645
Grid lines in React Diagram component	1646
Customize the gridlines visibility	1646
Appearance	1648
Line intervals	1650
Snapping	1653
Snap to lines	1653
Customization of snap intervals	1654
Snap to objects	1656
Page settings in React Diagram component	1658
Page size and appearance	1659
Set background image	1662
Multiple page and page breaks	1664
Boundary constraints	1667
Scroll settings in React Diagram component	1669
Get current scroll status	1669
Define scroll status	1670
Update scroll status	1671
AutoScroll	1672
Autoscroll border	1674
Scroll limit	1676
Scroll Padding	1678
Scrollable Area	1679
UpdateViewport	1681
Data binding in React Diagram component	1681
Local data	1682

Remote data.....	1686
CRUD	1690
Read DataSource.....	1690
How to perform Editing at runtime	1690
InsertData.....	1691
UpdateData	1692
DeleteData	1694
See Also	1695
Automatic layout in React Diagram component.....	1695
Layout modes.....	1695
Hierarchical layout	1696
Radial tree layout	1700
Organizational Chart	1705
Symmetric layout	1729
Mind Map layout.....	1733
Tree Orientation in layout.....	1733
Complex hierarchical tree	1738
Customize layout.....	1749
Accessibility in React Diagram component.....	1774
WAI-ARIA attributes.....	1775
Aria-label	1775
Keyboard interaction	1776
Ensuring accessibility	1776
See also	1777
Commands in React Diagram component	1777
Align	1777
Distribute	1781
Sizing	1784
Clipboard	1787
Grouping	1790
Z-Order command.....	1793
Zoom	1802
Nudge command.....	1803
Nudge by using arrow keys	1804
BringIntoView	1805

BringToCenter	1806
FitToPage command	1807
Command manager.....	1808
Custom command	1808
Modify the existing command	1810
See Also	1812
Undo redo in React Diagram component	1812
Undo and redo	1812
Undo/redo through shortcut keys	1812
Undo/redo through public APIs	1812
canLog	1816
History change event	1819
Stack Limit	1819
Retain Selection	1821
Virtualization in React Diagram component.....	1822
Virtualization in Diagram	1822
Serialization in React Diagram component.....	1823
Save	1823
Load.....	1823
Prevent Default Values	1824
Export in React Diagram component.....	1824
Exporting options.....	1824
File Name	1824
Format.....	1825
Margin.....	1825
Mode	1826
Region	1826
PageSettings.....	1826
Content	1827
Custom bounds	1828
Export diagram with stretch option.....	1828
Print.....	1829
Limitations.....	1830
Tooltip in React Diagram component	1830
Default tooltip.....	1830

Common tooltip for all nodes and connectors	1832
Tooltip for a specific node/connector.....	1835
Tooltip for Ports	1837
Tooltip template content	1840
Tooltip alignments	1843
Tooltip animation.....	1847
User handle in React Diagram component	1849
Alignment.....	1850
Fixed user handles	1854
Initialization an fixed user handles	1854
Customization	1855
Customizing the node fixed user handle	1858
Customizing the connector fixed user handle	1867
Style in React Diagram component.....	1878
Customizing the connector end point handle.....	1878
Customizing the connector end point handle when connected.....	1878
Customizing the connector end point handle when disabled	1878
Customizing the bezier connector handle	1878
Customizing the bezier connector line	1879
Customizing the resize handle	1879
Customizing the selector pivot line.....	1879
Customizing the selector border.....	1879
Customizing the rotate handle	1880
Customizing the symbolpalette while hovering	1880
Customizing the symbolpalette when selected	1880
Customizing the ruler.....	1880
Customizing the ruler overlap.....	1880
Customizing the text edit.....	1881
Customizing the text edit on selection	1881
Ruler in React Diagram component	1881
Adding Rulers to the Diagram	1881
Customizing the Ruler	1882
Layers in React Diagram component	1884
Visible.....	1884
Lock	1886

Objects	1888
AddInfo.....	1891
Context menu in React Diagram component	1900
Customize context menu	1900
Template Support for Context menu.....	1908
Context menu events.....	1913
Symbol palette in React Diagram component	1917
Create symbol palette.....	1918
Add palettes to SymbolPalette	1918
Restrict expansion of the palette panel.....	1939
Stretch the symbols into the palette	1943
Add/Remove symbols to palette at runtime	1946
Customize the size of symbols	1946
Symbol preview.....	1948
Default settings	1951
Adding symbol description for symbols in the palette	1954
Appearance of symbol description	1958
Tooltip for symbols in symbol palette	1963
Palette interaction	1969
DragEnter	1969
DragLeave	1969
DragOver	1969
See Also	1969
Overview in React Diagram component	1969
Create overview	1970
How to load EJ1 diagram in EJ2 diagram	1976
Summary of Dialog component	1977
Dialog	1977
Getting Started.....	1977
Dependencies.....	1977
Installation and configuration.....	1978
Adding Syncfusion packages	1978
Adding Dialog to the application.....	1978
Adding CSS reference.....	1981
Run the application	1982

Modal Dialog	1984
Enable header	1987
Enable footer.....	1988
Draggable	1992
Positioning.....	1995
See Also	2001
Getting Started.....	2001
Dependencies.....	2001
Installation and configuration.....	2002
Adding Syncfusion packages	2002
Adding Dialog to the application.....	2003
Adding CSS reference.....	2004
Run the application	2004
Modal Dialog	2006
Enable header	2007
Enable footer.....	2007
Draggable	2009
Positioning.....	2011
See Also	2016
Template in React Dialog component.....	2016
Header.....	2016
Footer.....	2016
Content	2017
See Also	2023
Animation in React Dialog component	2023
Resize in React Dialog component.....	2026
Localization in React Dialog component.....	2030
Loading translations.....	2030
Accessibility in React Dialog component	2033
WAI-ARIA attributes.....	2034
Keyboard interaction	2034
See Also	2038
Ensuring accessibility	2038
See also	2038
Style in React Dialog component	2038

Customizing the dialog header	2038
Customizing the dialog content	2038
Customizing modal dialog overlay	2039
Customizing the dialog resize icon.....	2039
Customizing the dialog close button.....	2039
Customizing the dialog footer button.....	2039
How To	2040
Create nested dialog in React Dialog component.....	2040
Position the dialog on center of the page on scrolling in React Dialog component.....	2043
Load dialog content using ajax in React Dialog component	2044
Render a dialog without header in React Dialog component.....	2044
Show dialog with full screen in React Dialog component.....	2047
Display a dialog with custom position in React Dialog component.....	2050
Prevent closing of modal dialog in React Dialog component	2053
Prevent the focus on the first element in React Dialog component	2058
Prevent opening of the dialog in React Dialog component	2062
Read all the values from dialog on button click in React Dialog component	2069
Customize the dialog appearance in React Dialog component	2077
Close dialog while click on outside of dialog in React Dialog component.....	2080
Add an icons to dialog buttons in React Dialog component.....	2084
Add a minimize maximize buttons in React Dialog component	2090
Setting max height to the dialog in React Dialog component	2101
React hooks dialog in React Dialog component.....	2106
Ej1 api migration in React Dialog component.....	2108
Accessibility and Localization.....	2108
Header.....	2108
Footer.....	2109
Content	2110
Animation.....	2110
Draggable and resizing.....	2111
Target.....	2111
Position	2112
Visibility.....	2112
Dialog Mode.....	2112
Tooltip	2112

Control State	2112
State Maintenance.....	2113
Common.....	2113
Document Editor.....	2115
Overview	2115
Key Features.....	2115
Supported Web platforms	2116
Getting Started.....	2116
Dependencies.....	2117
Setup for Local Development.....	2117
Adding Syncfusion packages	2118
Adding CSS reference.....	2118
Adding Component	2119
Frequently Asked Questions	2124
Creating a Next.js Application Using Syncfusion React Components.....	2125
What is Next.js?	2125
Prerequisites	2125
Create a Next.js application	2125
Install Syncfusion React packages.....	2126
Import Syncfusion CSS styles	2126
Add Syncfusion React component	2126
Run the application	2127
Feature module in React Document editor component.....	2127
Accessibility in React Document editor component.....	2130
Keyboard interaction	2131
Ensuring accessibility	2131
See also	2131
Import in React Document editor component	2131
Import document from local machine.....	2133
Convert word documents into SFDT	2136
Compatibility with Microsoft Word	2139
See Also.....	2140
Export in React Document editor component.....	2140
SFDT export.....	2141
Word export.....	2143

Template export.....	2145
Text export	2147
Export as blob	2149
See Also	2152
Web services in React Document editor component	2153
Which operations require server-side interaction.....	2153
Required Web API structure	2154
Customize the expected method name.....	2154
Add the custom headers to XMLHttpRequest	2155
Modify the XMLHttpRequest before request send	2156
Server Deployment	2158
Word processor server docker image overview in React Document editor component	2158
Provide your license key for activation.....	2159
Provide your license key for activation.....	2159
Provide your license key for activation.....	2160
Provide your license key for activation.....	2161
How to deploy word processor server docker container in azure app service in React Document editor component	2163
How to deploy word processor server docker container in azure kubernetes service in React Document editor component	2164
How to publish documenteditor web api application in azure app service from visual studio in React Document editor component	2167
Collaborative Editing.....	2169
Prerequisites	2169
How to enable collaborative editing in client side.....	2169
How to enable collaborative editing in ASP.NET Core.....	2173
Image in React Document editor component	2179
Image resizing	2181
Changing size.....	2182
Text wrapping style	2182
Positioning the image	2182
See Also	2182
Shapes in React Document editor component	2183
Supported shapes	2183
Text box Shape	2183
Shape Resizer	2183

Text wrapping style	2184
Positioning the shape.....	2184
Text wrapping style in React Document editor component	2184
In-Line with Text.....	2184
In Front of Text.....	2184
Top and Bottom	2185
Behind	2185
Square	2185
Bookmark in React Document editor component	2185
Add bookmark.....	2186
Select Bookmark	2186
Delete Bookmark	2186
Get Bookmark from document	2186
Get Bookmark from selection	2186
Replace bookmark content	2187
Show or Hide bookmark.....	2187
Bookmark Dialog	2187
See Also	2188
Link in React Document editor component	2188
Navigate a hyperlink	2188
Copy link.....	2192
Add hyperlink	2192
Customize screen tip.....	2194
Remove hyperlink	2195
Hyperlink dialog	2195
See Also	2197
Table in React Document editor component.....	2197
Create a table	2197
Insert rows	2198
Delete table.....	2199
Delete row.....	2199
Delete column.....	2199
Merge cells.....	2199
Positioning the table	2200
How to work with tables	2200

See Also	2203
Table of contents in React Document editor component	2203
Inserting table of contents	2203
Update or edit table of contents	2205
See Also	2207
Header footer in React Document editor component.....	2207
Go to header footer region	2208
Link to previous.....	2208
Header and footer distance	2209
Close header footer region	2209
See Also	2209
Text format in React Document editor component	2209
Bold	2209
Italic.....	2210
Underline property	2210
Strikethrough property	2210
Superscript property	2211
Subscript property	2211
Size	2212
Color	2212
Font	2212
Highlight color.....	2212
Toolbar with options for text formatting.....	2213
See Also	2217
Paragraph format in React Document editor component.....	2217
Indentation.....	2217
Increase indent	2217
Decrease indent	2218
Text alignment	2218
Line spacing and its type	2218
Paragraph spacing.....	2218
Paragraph Border	2219
Pagination properties.....	2219
Show or Hide Paragraph marks.....	2220
Toolbar with paragraph formatting options	2220

See Also	2225
Styles in React Document editor component	2225
Styles definition overview	2225
Default style	2225
Style hierarchy	2225
Defining new styles	2226
Applying a style	2231
List format in React Document editor component	2232
Create bullet list	2232
Create numbered list	2232
Clear list	2232
Working with lists	2233
Editing numbered list	2240
See Also	2241
Table format in React Document editor component	2241
Cell margins	2241
Background color	2241
Cell spacing	2242
Cell vertical alignment	2242
Table alignment	2242
Cell width	2242
Table width	2242
Apply borders	2243
Working with row formatting	2245
See Also	2247
Section format in React Document editor component	2247
Page size	2247
Page margins	2247
Header distance	2247
Footer distance	2248
Columns	2248
Breaks	2248
See Also	2249
Comments in React Document editor component	2249
Add a new comment	2249

Comment navigation.....	2249
Delete comment	2249
Delete all comment.....	2249
Protect the document in comments only mode.....	2249
Mention support in Comments.....	2251
Track changes in React Document editor component.....	2253
Enable track changes in Document Editor	2253
Get all tracked revisions.....	2254
Accept or Reject all changes programmatically	2254
Accept or reject a specific revision	2255
Navigate between the tracked changes	2255
Filtering changes based on user.....	2256
Protect the document in track changes only mode.....	2256
Event	2258
Fields in React Document editor component	2259
Adding Fields	2259
Update fields.....	2260
Get field info	2260
Set field info	2260
See Also	2261
Form fields in React Document editor component	2261
Insert form field	2261
Get form field names	2262
Get form field properties	2262
Set form field properties.....	2262
Export form field data	2263
Import form field data	2263
Reset form fields	2263
Protect the document in form filling mode	2263
Clipboard in React Document editor component.....	2265
Copy	2265
Cut	2265
Paste.....	2265
Local paste (copy/paste within control)	2265
Paste with formatting	2267

See Also	2268
History in React Document editor component	2268
Enable or disable history	2268
Undo and redo	2269
Stack size	2269
See Also	2269
Find and replace in React Document editor component	2270
Options pane	2270
Search	2271
Search results	2272
Customize find and replace	2273
See Also	2275
Keyboard shortcut in React Document editor component	2275
Text formatting	2275
Paragraph formatting	2276
Clipboard	2276
Keyboard shortcut to navigate around the document	2276
Keyboard shortcut to extend selection	2277
Find and Replace	2277
Create, Save and Print document	2277
Edit Operation	2278
Insert special characters	2278
Dialog	2278
See Also	2278
Scrolling zooming in React Document editor component	2278
Zooming	2288
Page Fit Type	2290
Zoom option using UI	2292
Print in React Document editor component	2301
Improve print quality	2306
Print using window object	2306
Page setup	2308
See Also	2313
Dialog in React Document editor component	2313
Font Dialog	2313

Paragraph dialog	2315
Table dialog	2316
Bookmark dialog	2316
Hyperlink dialog	2317
Table of contents dialog.....	2318
Styles Dialog	2319
Style dialog.....	2320
List dialog	2321
Borders and shading dialog.....	2322
Table options dialog.....	2323
Table properties dialog	2324
Page setup dialog	2325
See Also	2326
Right to left in React Document editor component	2326
Chart in React Document editor component.....	2340
Supported Chart Types	2377
Restrict editing in React Document editor component	2378
Set current user	2378
Highlighting the text area	2378
Restrict Editing Pane	2378
See Also	2390
Spell check in React Document editor component.....	2390
Features	2391
Enable SpellCheck	2391
Disable SpellCheck	2391
Spell check settings	2391
Context menu.....	2393
Global local in React Document editor component.....	2395
Document Editor	2395
Color Picker	2400
Notes in React Document editor component	2400
Insert footnotes	2401
Insert endnotes	2402
Update or edit footnotes and endnotes	2403
View in React Document Editor Component	2404

Web Layout	2404
Ruler	2404
Heading Navigation Pane	2407
How To	2408
Override the keyboard shortcuts in React Document editor component	2408
Customize context menu in React Document editor component	2410
Customize tool bar in React Document editor component	2419
Change document view in React Document editor component	2421
Open default document in React Document editor component	2422
Read by default in React Document editor component	2441
Open document by address in React Document editor component	2452
Deploy document editor component for mobile in React Document editor component	2454
Disable optimized text measuring in React Document editor component	2456
Get the selected content in React Document editor component	2458
Set default format in document editor in React Document editor component	2461
Show hide spinner in React Document editor component	2466
Resize document editor in React Document editor component	2472
Export document as pdf in React Document editor component	2474
Customize font family drop down in React Document editor component	2479
Auto save document in document editor in React Document editor component	2480
Retrieve the bookmark content as text in React Document editor component	2484
Get current word in React Document editor component	2489
Insert page number and navigate to page in React Document editor component	2491
Move selection to specific position in React Document editor component	2495
Disable header and footer edit in document editor in React Document editor component	2497
Insert text in current position in React Document editor component	2503
Change the cursor color in document editor in React Document editor component	2507
Hide tool bar and properties pane in React Document editor component	2508
Insert text or image in table programmatically in React Document editor component	2509
Change the default search highlight color in React Document editor component	2512
How to optimize the SFDT file	2513
How to disable auto focus in Syncfusion React Document Editor component	2514
How to disable drag and drop in document editor in React Document editor component	2514
Enable ruler	2515
Customize color picker in React Document editor control	2520

DropDownButton.....	2521
Getting Started.....	2521
Dependencies.....	2521
Installation and configuration.....	2522
Adding syncfusion packages	2522
Adding CSS reference.....	2522
Adding DropDownButton component.....	2523
Binding data source	2523
Run the application	2524
See Also	2525
Icons in React Drop down button component.....	2526
DropDownButton icons.....	2526
Vertical button	2530
See Also	2531
Popup items in React Drop down button component.....	2531
Icons	2531
Navigations	2533
Template	2534
Separator.....	2539
See Also	2541
Accessibility in React DropDownButton component.....	2541
WAI-ARIA attributes.....	2542
Keyboard interaction	2542
Ensuring accessibility	2542
See also	2543
Style and appearance in React Drop down button component	2543
How To	2543
Change caret icon in React Drop down button component	2543
Create dropdownbutton with rounded corner in React Drop down button component	2545
Create right to left dropdownbutton in React Drop down button component	2546
Customize icon and width in React Drop down button component	2547
Disable a dropdownbutton in React Drop down button component.....	2548
Group popup items with listview component in React Drop down button component.....	2550
Hide dropdown arrow in React Drop down button component	2551
Open a dialog on popup item click in React Drop down button component	2552

Position popup open in React Drop down button component	2554
Underline a character in the item text in React Drop down button component	2556
DropDownList	2557
Getting Started.....	2557
Getting Started.....	2557
Getting Started with the React DropDownList Component in the Preact.....	2564
Data binding in React Drop down list component	2568
Binding local data	2568
Binding remote data	2574
See Also	2576
Value binding in DropDownList	2577
Primitive Data Types	2577
Object Data Types	2578
Templates in React Drop down list component	2579
Item template	2579
Value template.....	2582
Group template.....	2586
Header template	2589
Footer template	2592
No records template	2594
Action failure template	2596
See Also	2599
Virtualization in DropDown List	2599
Binding local data	2599
Binding remote data	2600
Grouping	2602
Filtering with Virtualization.....	2604
Grouping in React Drop down list component	2605
Customization	2608
See Also	2608
Filtering in React Drop down list component	2608
Limit the minimum filter character	2611
Change the filter type	2615
Case sensitive filtering	2618
Diacritics Filtering.....	2621

See Also	2623
Localization in React Drop down list component	2623
Loading translations.....	2624
See Also	2627
Style in React Drop down list component	2627
Customizing the appearance of wrapper element	2627
Customizing the dropdown icon's color	2627
Customizing the focus color	2627
Customizing the outline theme's focus color	2628
Customizing the disabled component's text color	2628
Customizing the float label element's focusing color	2628
Customizing the color of the placeholder text	2629
Customizing the background color of focus, hover, and active item's	2629
Customizing the appearance of pop-up element	2629
Adding mandatory asterisk to placeholder and float label.....	2629
Accessibility in React Drop down list component.....	2631
WAI-ARIA attributes.....	2632
Keyboard interaction	2632
Ensuring accessibility	2636
See also	2636
How To	2636
Add item in React Drop down list component.....	2636
Cascading in React Drop down list component	2638
Clear item in React Drop down list component.....	2642
Close popup in React Drop down list component	2643
Group header in React Drop down list component	2644
Highlight filtering in React Drop down list component	2645
Icons support in React Drop down list component	2647
Incremental search in React Drop down list component	2648
Modify data in React Drop down list component.....	2649
Multiple cascading in React Drop down list component	2650
Remote data bind in React Drop down list component	2658
Remove item in React Drop down list component.....	2660
Search on filtering in React Drop down list component	2662
Tooltip in React Drop down list component	2664

Value change in React Drop down list component.....	2666
Value support in React Drop down list component.....	2668
Ej1 api migration in React Drop down list component	2668
DataBinding.....	2668
Filtering	2669
Template	2670
Virtual Scrolling.....	2670
Applying CSS.....	2670
Sorting.....	2671
Popup.....	2671
Placeholder	2672
Grouping	2673
Accessibility.....	2673
Miscellaneous	2673
Selection.....	2674
Common.....	2675
Dropdown Tree	2676
Getting Started.....	2676
Dependencies.....	2676
Installation and configuration.....	2677
Adding syncfusion packages	2677
Adding Dropdown Tree component	2677
Adding CSS reference.....	2678
Binding data source	2678
Run the application	2681
Data binding in React Drop down tree component.....	2683
Local data	2683
Remote data.....	2687
Prevent Node selection.....	2689
Templates in React Drop down tree component	2691
Item template	2691
Header template	2692
Footer template	2694
No records template	2695
Action failure template	2696

Custom template to show selected items in input	2697
Checkbox in React Drop down tree component.....	2700
Auto Check	2702
Select All.....	2704
Localization in React Drop down tree component	2705
Accessibility in React Dropdown Tree component	2706
WAI-ARIA attributes.....	2707
Keyboard interaction	2707
Ensuring accessibility	2708
See also	2708

DateTimePicker

Getting started

This section explains you the steps required to create a simple DateTimePicker and demonstrate the basic usage of the DateTimePicker component.

To get start quickly with React DateTime Picker, you can check on this video:

Dependencies

The below list of dependencies are required to use the `DateTimePicker` component in your application.

```
`javascript
|-- @syncfusion/ej2-react-calendars
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-calendars
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
`,`
```

Installation and configuration

You can use [create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`
npm install -g create-react-app
`,`
```

- To setup basic `React` sample use following commands.

```
`
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
`,`
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](#) public registry. You can choose the component that you want to install. For this application, we are going to use `DateTimePicker` component.

To install DateTimePicker component, use the following command

```
`bash
npm install @syncfusion/ej2-react-calendars --save
`
```

Adding Style sheet to the Application

To render the DateTimePicker component, need to import DateTimePicker and its dependent component's styles as given below in `App.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-lists/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-react-calendars/styles/material.css";
`
```

Note: If you want to refer the combined component styles, please make use of our [CRG](#) (Custom Resource Generator) in your application.

Adding DateTimePicker component to the Application

- To include the DateTimePicker component in application import the `DateTimePickerComponent` from `ej2-react-calendars` package in `App.tsx`.
- Then add the DateTimePicker component as shown in below code example.

[src/App.tsx]

[Class-component]

```
`ts
// import the DateTimePickerComponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import './App.css';
export default class App extends React.Component<{}, {}> {
  public render() {
    return <DateTimePickerComponent id="datetimepicker" />;
  }
}
```

[Functional-component]

```

`ts
// import the DateTimePickerComponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import './App.css';
function App() {
return <DateTimePickerComponent id="datetimepicker" />;
}

```

Run the application

Now run the **npm start** command in the console, it will run your application and open the browser window.

```

npm start

```

The below examples shows the basic DateTimePicker component.

[Class-component]**INDEX.JSX**

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  render() {
    return <DateTimePickerComponent id="datetimepicker"
placeholder="Select a date and time"/>;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
  public render() {
    return <DateTimePickerComponent id="datetimepicker"
placeholder="Select a date and time"/>;
  }
}

```

```

    }
  }
  ReactDOM.render(<App />, document.getElementById('element'));

```

[Functional-component]

INDEX.JSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  return <DateTimePickerComponent id="datetimepicker" placeholder="Select
a date and time"/>;
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  return <DateTimePickerComponent id="datetimepicker" placeholder="Select
a date and time"/>;
}
ReactDOM.render(<App />, document.getElementById('element'));

```

Setting the min and max

The minimum and maximum date time can be defined with the help of `min` and `max` property. The following example demonstrates to set the `min` and `max` on initializing the DateTimePicker. To know more about range restriction in DateTimePicker, please refer this [page](#).

[Class-component]

INDEX.JSX

```

// import the datetimepicker component
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // initialize the value, min and max
  minDate = new Date("11/22/2019 12:00 PM");
  maxDate = new Date("11/25/2019 5:00 PM");
  render() {
    return <DateTimePickerComponent id="datetimepicker"
min={this.minDate} max={this.maxDate}/>;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```
// import the datetimepicker component
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // initialize the value, min and max
    private minDate: Date = new Date("11/22/2019 12:00 PM");
    private maxDate: Date = new Date("11/25/2019 5:00 PM");
    public render() {
        return <DateTimePickerComponent id="datetimepicker"
min={this.minDate} max={this.maxDate} />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]**INDEX.JSX**

```
// import the datetimepicker component
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // initialize the value, min and max
    const minDate = new Date("11/22/2019 12:00 PM");
    const maxDate = new Date("11/25/2019 5:00 PM");
    return <DateTimePickerComponent id="datetimepicker" min={minDate}
max={maxDate}/>;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
// import the datetimepicker component
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // initialize the value, min and max
    const minDate: Date = new Date("11/22/2019 12:00 PM");
    const maxDate: Date = new Date("11/25/2019 5:00 PM");
    return <DateTimePickerComponent id="datetimepicker" min={minDate}
max={maxDate} />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

See Also

- [Render DateTimePicker with specific culture](#)

Globalization in React Datetimepicker component

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [Internationalization](#), and also add culture specific customization and translation to the text [localization](#).

By default, DateTimePicker date format, week, month, time format and meridian names are specific to the American English culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the `loadCldr` method to load culture specific CLDR JSON data. To use a different culture other than English, follow the steps below:

- Install the `CLDR-Data` package by using the following command (installs all the CLDR JSON data). To know more about CLDR-Data refer to the [CLDR-Data](#) link.

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

Once the package is installed, you can find the culture specific JSON data under the location `\node_modules\cldr-data`.

- Import the installed CLDR JSON data into the `app.ts` file.
- Use the [loadCldr](#) method to load the culture specific CLDR JSON data from the installed location to `app.ts` file.
- DateTimePicker displayed `Sunday` as the first day of week based on default culture ("en-US"). If you want to display the DateTimePicker with loaded culture's first day of week, you need to import `weekdata.json` file from the `cldr-data/supplemental` as given in the code example.

```
`ts
import { loadCldr } from '@syncfusion/ej2-base';
import { loadCldr } from "@syncfusion/ej2-base";
import * as gregorian from 'cldr-data/main/de/ca-gregorian.json';
import * as numbers from 'cldr-data/main/de/numbers.json';
import * as timeZoneNames from 'cldr-data/main/de/timeZoneNames.json';
import * as numberingSystems from 'cldr-data/supplemental/numberingSystems.json';
import * as weekData from 'cldr-data/supplemental/weekData.json'; // To load the culture based first
day of week
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames, weekData);
```

if you are facing the error `/node_modules/cldr-data/main/de/*.json (1,1): unused expression, expected an assignment or function call` when you are adding the json files to render the culture sample, then add the below configuration in your `tslint.json` file

```
`ts
"linterOptions": {
"exclude": [
"*.json",
"/*.json"
]
}
`
```

The **Localization** library allows you to localize default text content of the DateTimePicker. The DateTimePicker component has static text for **today** feature that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

Locale keywords | Text

today | Name of the button to choose Today date.

placeholder | Hint to describe expected value in input element.

- Before changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through **load** method of **L10n** class.

```
`ts
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized placeholder value
L10n.load({
'de': {
'datetimepicker': {
placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
today:'heute'
}
}
});
`
```

- Set the culture by using the [locale](#) property. In the following code example, the DateTimePicker is initialized in **German** culture with corresponding localized text.

The following example demonstrates the DateTimePicker in **German** culture.

[Class-component]**CA-GREGORIAN.JSON**

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 12879 $",
          "_cldrVersion": "30.0.3"
        },
        "language": "de"
      },
      "dates": {
        "calendars": {
          "gregorian": {
            "months": {
              "format": {
                "abbreviated": {
                  "1": "Jan.",
                  "2": "Feb.",
                  "3": "März",
                  "4": "Apr.",
                  "5": "Mai",
                  "6": "Juni",
                  "7": "Juli",
                  "8": "Aug.",
                  "9": "Sep.",
                  "10": "Okt.",
                  "11": "Nov.",
                  "12": "Dez."
                },
                "narrow": {
                  "1": "J",
                  "2": "F",
                  "3": "M",
                  "4": "A",
                  "5": "M",
                  "6": "J",
                  "7": "J",
                  "8": "A",
                  "9": "S",
                  "10": "O",
                  "11": "N",
                  "12": "D"
                },
                "wide": {
                  "1": "Januar",
                  "2": "Februar",
                  "3": "März",
                  "4": "April",
                  "5": "Mai",
                  "6": "Juni",
                  "7": "Juli",
                  "8": "August",
                  "9": "September",

```

```

        "10": "Oktober",
        "11": "November",
        "12": "Dezember"
    }
},
"stand-alone": {
    "abbreviated": {
        "1": "Jan",
        "2": "Feb",
        "3": "Mär",
        "4": "Apr",
        "5": "Mai",
        "6": "Jun",
        "7": "Jul",
        "8": "Aug",
        "9": "Sep",
        "10": "Okt",
        "11": "Nov",
        "12": "Dez"
    },
    "narrow": {
        "1": "J",
        "2": "F",
        "3": "M",
        "4": "A",
        "5": "M",
        "6": "J",
        "7": "J",
        "8": "A",
        "9": "S",
        "10": "O",
        "11": "N",
        "12": "D"
    },
    "wide": {
        "1": "Januar",
        "2": "Februar",
        "3": "März",
        "4": "April",
        "5": "Mai",
        "6": "Juni",
        "7": "Juli",
        "8": "August",
        "9": "September",
        "10": "Oktober",
        "11": "November",
        "12": "Dezember"
    }
}
},
"days": {
    "format": {
        "abbreviated": {
            "sun": "So.",
            "mon": "Mo.",
            "tue": "Di.",
            "wed": "Mi.",

```

```
        "thu": "Do.",
        "fri": "Fr.",
        "sat": "Sa."
    },
    "narrow": {
        "sun": "S",
        "mon": "M",
        "tue": "D",
        "wed": "M",
        "thu": "D",
        "fri": "F",
        "sat": "S"
    },
    "short": {
        "sun": "So.",
        "mon": "Mo.",
        "tue": "Di.",
        "wed": "Mi.",
        "thu": "Do.",
        "fri": "Fr.",
        "sat": "Sa."
    },
    "wide": {
        "sun": "Sonntag",
        "mon": "Montag",
        "tue": "Dienstag",
        "wed": "Mittwoch",
        "thu": "Donnerstag",
        "fri": "Freitag",
        "sat": "Samstag"
    }
},
"stand-alone": {
    "abbreviated": {
        "sun": "So",
        "mon": "Mo",
        "tue": "Di",
        "wed": "Mi",
        "thu": "Do",
        "fri": "Fr",
        "sat": "Sa"
    },
    "narrow": {
        "sun": "S",
        "mon": "M",
        "tue": "D",
        "wed": "M",
        "thu": "D",
        "fri": "F",
        "sat": "S"
    },
    "short": {
        "sun": "So.",
        "mon": "Mo.",
        "tue": "Di.",
        "wed": "Mi.",
        "thu": "Do.",
```

```

        "fri": "Fr.",
        "sat": "Sa."
    },
    "wide": {
        "sun": "Sonntag",
        "mon": "Montag",
        "tue": "Dienstag",
        "wed": "Mittwoch",
        "thu": "Donnerstag",
        "fri": "Freitag",
        "sat": "Samstag"
    }
},
"quarters": {
    "format": {
        "abbreviated": {
            "1": "Q1",
            "2": "Q2",
            "3": "Q3",
            "4": "Q4"
        },
        "narrow": {
            "1": "1",
            "2": "2",
            "3": "3",
            "4": "4"
        },
        "wide": {
            "1": "1. Quartal",
            "2": "2. Quartal",
            "3": "3. Quartal",
            "4": "4. Quartal"
        }
    },
    "stand-alone": {
        "abbreviated": {
            "1": "Q1",
            "2": "Q2",
            "3": "Q3",
            "4": "Q4"
        },
        "narrow": {
            "1": "1",
            "2": "2",
            "3": "3",
            "4": "4"
        },
        "wide": {
            "1": "1. Quartal",
            "2": "2. Quartal",
            "3": "3. Quartal",
            "4": "4. Quartal"
        }
    }
},
"dayPeriods": {

```

```

    "format": {
      "abbreviated": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "morgens",
        "morning2": "vormittags",
        "afternoon1": "mittags",
        "afternoon2": "nachmittags",
        "evening1": "abends",
        "night1": "nachts"
      },
      "narrow": {
        "midnight": "Mitternacht",
        "am": "vm.",
        "pm": "nm.",
        "morning1": "morgens",
        "morning2": "vormittags",
        "afternoon1": "mittags",
        "afternoon2": "nachmittags",
        "evening1": "abends",
        "night1": "nachts"
      },
      "wide": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "morgens",
        "morning2": "vormittags",
        "afternoon1": "mittags",
        "afternoon2": "nachmittags",
        "evening1": "abends",
        "night1": "nachts"
      }
    },
    "stand-alone": {
      "abbreviated": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "Morgen",
        "morning2": "Vormittag",
        "afternoon1": "Mittag",
        "afternoon2": "Nachmittag",
        "evening1": "Abend",
        "night1": "Nacht"
      },
      "narrow": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "Morgen",
        "morning2": "Vormittag",
        "afternoon1": "Mittag",
        "afternoon2": "Nachmittag",
        "evening1": "Abend",
        "night1": "Nacht"
      }
    }
  }

```

```

    },
    "wide": {
      "midnight": "Mitternacht",
      "am": "vorm.",
      "pm": "nachm.",
      "morning1": "Morgen",
      "morning2": "Vormittag",
      "afternoon1": "Mittag",
      "afternoon2": "Nachmittag",
      "evening1": "Abend",
      "night1": "Nacht"
    }
  },
  "eras": {
    "eraNames": {
      "0": "v. Chr.",
      "0-alt-variant": "vor unserer Zeitrechnung",
      "1": "n. Chr.",
      "1-alt-variant": "unserer Zeitrechnung"
    },
    "eraAbbr": {
      "0": "v. Chr.",
      "0-alt-variant": "v. u. Z.",
      "1": "n. Chr.",
      "1-alt-variant": "u. Z."
    },
    "eraNarrow": {
      "0": "v. Chr.",
      "0-alt-variant": "v. u. Z.",
      "1": "n. Chr.",
      "1-alt-variant": "u. Z."
    }
  },
  "dateFormats": {
    "full": "EEEE, d. MMMM y",
    "long": "d. MMMM y",
    "medium": "dd.MM.y",
    "short": "dd.MM.yy"
  },
  "timeFormats": {
    "full": "HH:mm:ss zzzz",
    "long": "HH:mm:ss z",
    "medium": "HH:mm:ss",
    "short": "HH:mm"
  },
  "dateTimeFormats": {
    "full": "{1} 'um' {0}",
    "long": "{1} 'um' {0}",
    "medium": "{1}, {0}",
    "short": "{1}, {0}",
    "availableFormats": {
      "d": "d",
      "E": "ccc",
      "Ed": "E, d.",
      "Ehm": "E h:mm a",
      "EHm": "E, HH:mm",

```

```

    "Ehms": "E, h:mm:ss a",
    "EHms": "E, HH:mm:ss",
    "Gy": "y G",
    "GyMMM": "MMM y G",
    "GyMMMd": "d. MMM y G",
    "GyMMMED": "E, d. MMM y G",
    "h": "h 'Uhr' a",
    "H": "HH 'Uhr'",
    "hm": "h:mm a",
    "Hm": "HH:mm",
    "hms": "h:mm:ss a",
    "Hms": "HH:mm:ss",
    "hmsv": "h:mm:ss a v",
    "Hmsv": "HH:mm:ss v",
    "hmv": "h:mm a v",
    "Hmv": "HH:mm v",
    "M": "L",
    "Md": "d.M.",
    "MEd": "E, d.M.",
    "MMd": "d.MM.",
    "MMdd": "dd.MM.",
    "MMM": "LLL",
    "MMMd": "d. MMM",
    "MMMED": "E, d. MMM",
    "MMMMd": "d. MMMM",
    "MMMMEd": "E, d. MMMM",
    "MMMMMW": "'Woche' W 'im' MMM",
    "MMMMW": "'Woche' W 'im' MMM",
    "ms": "mm:ss",
    "y": "y",
    "yM": "M.y",
    "yMd": "d.M.y",
    "yMEd": "E, d.M.y",
    "yMM": "MM.y",
    "yMMdd": "dd.MM.y",
    "yMMM": "MMM y",
    "yMMMd": "d. MMM y",
    "yMMMED": "E, d. MMM y",
    "yMMMM": "MMMM y",
    "yQQQ": "QQQ y",
    "yQQQQ": "QQQQ y",
    "yw": "'Woche' w 'des' 'Jahres' y",
    "yw": "'Woche' w 'des' 'Jahres' y"
  },
  "appendItems": {
    "Day": "{0} ({2}: {1})",
    "Day-Of-Week": "{0} {1}",
    "Era": "{1} {0}",
    "Hour": "{0} ({2}: {1})",
    "Minute": "{0} ({2}: {1})",
    "Month": "{0} ({2}: {1})",
    "Quarter": "{0} ({2}: {1})",
    "Second": "{0} ({2}: {1})",
    "Timezone": "{0} {1}",
    "Week": "{0} ({2}: {1})",
    "Year": "{1} {0}"
  },

```

```

"intervalFormats": {
  "intervalFormatFallback": "{0} - {1}",
  "d": {
    "d": "d.-d."
  },
  "h": {
    "a": "h 'Uhr' a - h 'Uhr' a",
    "h": "h - h 'Uhr' a"
  },
  "H": {
    "H": "HH-HH 'Uhr'"
  },
  "hm": {
    "a": "h:mm a - h:mm a",
    "h": "h:mm-h:mm a",
    "m": "h:mm-h:mm a"
  },
  "Hm": {
    "H": "HH:mm-HH:mm 'Uhr'",
    "m": "HH:mm-HH:mm 'Uhr'"
  },
  "hmv": {
    "a": "h:mm a - h:mm a v",
    "h": "h:mm-h:mm a v",
    "m": "h:mm-h:mm a v"
  },
  "Hmv": {
    "H": "HH:mm-HH:mm 'Uhr' v",
    "m": "HH:mm-HH:mm 'Uhr' v"
  },
  "hv": {
    "a": "h a - h a v",
    "h": "h-h a v"
  },
  "Hv": {
    "H": "HH-HH 'Uhr' v"
  },
  "M": {
    "M": "M.-M."
  },
  "Md": {
    "d": "dd.MM. - dd.MM.",
    "M": "dd.MM. - dd.MM."
  },
  "MED": {
    "d": "E, dd.MM. - E, dd.MM.",
    "M": "E, dd.MM. - E, dd.MM."
  },
  "MMM": {
    "M": "MMM-MMM"
  },
  "MMMd": {
    "d": "d.-d. MMM",
    "M": "d. MMM - d. MMM"
  },
  "MMMED": {
    "d": "E, d. - E, d. MMM",

```



```
"M": "E, d. MMM - E, d. MMM"  
},  
"MMMM": {  
    "M": "LLLL-LLLL"  
},  
"Y": {  
    "y": "y-y"  
},  
"YM": {  
    "M": "MM.y - MM.y",  
    "y": "MM.y - MM.y"  
},  
"YMd": {  
    "d": "dd.MM.y - dd.MM.y",  
    "M": "dd.MM.y - dd.MM.y",  
    "y": "dd.MM.y - dd.MM.y"  
},  
"YMEd": {  
    "d": "E, dd.MM.y - E, dd.MM.y",  
    "M": "E, dd.MM.y - E, dd.MM.y",  
    "y": "E, dd.MM.y - E, dd.MM.y"  
},  
"YMMM": {  
    "M": "MMM-MMM y",  
    "y": "MMM y - MMM y"  
},  
"YMMMd": {  
    "d": "d.-d. MMM y",  
    "M": "d. MMM - d. MMM y",  
    "y": "d. MMM y - d. MMM y"  
},  
"YMMEd": {  
    "d": "E, d. - E, d. MMM y",  
    "M": "E, d. MMM - E, d. MMM y",  
    "y": "E, d. MMM y - E, d. MMM y"  
},  
"YMMMM": {  
    "M": "MMMM-MMMM y",  
    "y": "MMMM y - MMMM y"  
}  
  
}  
  
}  
  
}  
  
}
```

CA-GREGORIAN.JSX

```
{
    "main";
    {
        "de";
        {

```

```
"identity";
{
  "version";
  {
    "_number";
    "$Revision: 12879 $",
    "_cldrVersion";
    "30.0.3";
  }
  "language";
  "de";
}
"dates";
{
  "calendars";
  {
    "gregorian";
    {
      "months";
      {
        "format";
        {
          "abbreviated";
          {
            "1";
            "Jan.",
            "2";
            "Feb.",
            "3";
            "März",
            "4";
            "Apr.",
            "5";
            "Mai",
            "6";
            "Juni",
            "7";
            "Juli",
            "8";
            "Aug.",
            "9";
            "Sep.",
            "10";
            "Okt.",
            "11";
            "Nov.",
            "12";
            "Dez.";
          }
          "narrow";
          {
            "1";
            "J",
            "2";
            "F",
            "3";
            "M",
```

```
        "4";
        "A",
        "5";
        "M",
        "6";
        "J",
        "7";
        "J",
        "8";
        "A",
        "9";
        "S",
        "10";
        "O",
        "11";
        "N",
        "12";
        "D";
    }
    "wide";
    {
        "1";
        "Januar",
        "2";
        "Februar",
        "3";
        "März",
        "4";
        "April",
        "5";
        "Mai",
        "6";
        "Juni",
        "7";
        "Juli",
        "8";
        "August",
        "9";
        "September",
        "10";
        "Oktober",
        "11";
        "November",
        "12";
        "Dezember";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "Jan",
        "2";
        "Feb",
        "3";
        "Mär",
```

```
        "4";
        "Apr",
        "5";
        "Mai",
        "6";
        "Jun",
        "7";
        "Jul",
        "8";
        "Aug",
        "9";
        "Sep",
        "10";
        "Okt",
        "11";
        "Nov",
        "12";
        "Dez";
    }
    "narrow";
    {
        "1";
        "J",
        "2";
        "F",
        "3";
        "M",
        "4";
        "A",
        "5";
        "M",
        "6";
        "J",
        "7";
        "J",
        "8";
        "A",
        "9";
        "S",
        "10";
        "O",
        "11";
        "N",
        "12";
        "D";
    }
    "wide";
    {
        "1";
        "Januar",
        "2";
        "Februar",
        "3";
        "März",
        "4";
        "April",
        "5";
```

```

        "Mai",
        "6";
        "Juni",
        "7";
        "Juli",
        "8";
        "August",
        "9";
        "September",
        "10";
        "Oktober",
        "11";
        "November",
        "12";
        "Dezember";
    }
}
}
"days";
{
    "format";
    {
        "abbreviated";
        {
            "sun";
            "So.",
            "mon";
            "Mo.",
            "tue";
            "Di.",
            "wed";
            "Mi.",
            "thu";
            "Do.",
            "fri";
            "Fr.",
            "sat";
            "Sa.";
        }
        "narrow";
        {
            "sun";
            "S",
            "mon";
            "M",
            "tue";
            "D",
            "wed";
            "M",
            "thu";
            "D",
            "fri";
            "F",
            "sat";
            "S";
        }
        "short";
    }
}

```

```
{
  "sun";
  "So.",
    "mon";
  "Mo.",
    "tue";
  "Di.",
    "wed";
  "Mi.",
    "thu";
  "Do.",
    "fri";
  "Fr.",
    "sat";
  "Sa.";
}
"wide";
{
  "sun";
  "Sonntag",
    "mon";
  "Montag",
    "tue";
  "Dienstag",
    "wed";
  "Mittwoch",
    "thu";
  "Donnerstag",
    "fri";
  "Freitag",
    "sat";
  "Samstag";
}
}
"stand-alone";
{
  "abbreviated";
  {
    "sun";
    "So",
      "mon";
    "Mo",
      "tue";
    "Di",
      "wed";
    "Mi",
      "thu";
    "Do",
      "fri";
    "Fr",
      "sat";
    "Sa";
  }
  "narrow";
  {
    "sun";
    "S",
```

```

        "mon";
        "M",
        "tue";
        "D",
        "wed";
        "M",
        "thu";
        "D",
        "fri";
        "F",
        "sat";
        "S";
    }
    "short";
    {
        "sun";
        "So.",
        "mon";
        "Mo.",
        "tue";
        "Di.",
        "wed";
        "Mi.",
        "thu";
        "Do.",
        "fri";
        "Fr.",
        "sat";
        "Sa.";
    }
    "wide";
    {
        "sun";
        "Sonntag",
        "mon";
        "Montag",
        "tue";
        "Dienstag",
        "wed";
        "Mittwoch",
        "thu";
        "Donnerstag",
        "fri";
        "Freitag",
        "sat";
        "Samstag";
    }
}
"quarters";
{
    "format";
    {
        "abbreviated";
        {
            "1";
            "Q1",

```

```

        "2";
        "Q2",
        "3";
        "Q3",
        "4";
        "Q4";
    }
    "narrow";
    {
        "1";
        "1",
        "2";
        "2",
        "3";
        "3",
        "4";
        "4";
    }
    "wide";
    {
        "1";
        "1. Quartal",
        "2";
        "2. Quartal",
        "3";
        "3. Quartal",
        "4";
        "4. Quartal";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "Q1",
        "2";
        "Q2",
        "3";
        "Q3",
        "4";
        "Q4";
    }
    "narrow";
    {
        "1";
        "1",
        "2";
        "2",
        "3";
        "3",
        "4";
        "4";
    }
    "wide";
    {
        "1";

```



```

        "1. Quartal",
        "2";
        "2. Quartal",
        "3";
        "3. Quartal",
        "4";
        "4. Quartal";
    }
}
}
"dayPeriods";
{
    "format";
    {
        "abbreviated";
        {
            "midnight";
            "Mitternacht",
            "am";
            "vorm.",
            "pm";
            "nachm.",
            "morning1";
            "morgens",
            "morning2";
            "vormittags",
            "afternoon1";
            "mittags",
            "afternoon2";
            "nachmittags",
            "evening1";
            "abends",
            "night1";
            "nachts";
        }
        "narrow";
        {
            "midnight";
            "Mitternacht",
            "am";
            "vm.",
            "pm";
            "nm.",
            "morning1";
            "morgens",
            "morning2";
            "vormittags",
            "afternoon1";
            "mittags",
            "afternoon2";
            "nachmittags",
            "evening1";
            "abends",
            "night1";
            "nachts";
        }
        "wide";
    }
}

```

```
{
  "midnight";
  "Mitternacht",
    "am";
  "vorm.",
    "pm";
  "nachm.",
    "morning1";
  "morgens",
    "morning2";
  "vormittags",
    "afternoon1";
  "mittags",
    "afternoon2";
  "nachmittags",
    "evening1";
  "abends",
    "night1";
  "nachts";
}
}
"stand-alone";
{
  "abbreviated";
  {
    "midnight";
    "Mitternacht",
      "am";
    "vorm.",
      "pm";
    "nachm.",
      "morning1";
    "Morgen",
      "morning2";
    "Vormittag",
      "afternoon1";
    "Mittag",
      "afternoon2";
    "Nachmittag",
      "evening1";
    "Abend",
      "night1";
    "Nacht";
  }
  "narrow";
  {
    "midnight";
    "Mitternacht",
      "am";
    "vorm.",
      "pm";
    "nachm.",
      "morning1";
    "Morgen",
      "morning2";
    "Vormittag",
      "afternoon1";
```

```

        "Mittag",
        "afternoon2";
        "Nachmittag",
        "evening1";
        "Abend",
        "night1";
        "Nacht";
    }
    "wide";
    {
        "midnight";
        "Mitternacht",
        "am";
        "vorm.",
        "pm";
        "nachm.",
        "morning1";
        "Morgen",
        "morning2";
        "Vormittag",
        "afternoon1";
        "Mittag",
        "afternoon2";
        "Nachmittag",
        "evening1";
        "Abend",
        "night1";
        "Nacht";
    }
}
"eras";
{
    "eraNames";
    {
        "0";
        "v. Chr.",
        "0-alt-variant";
        "vor unserer Zeitrechnung",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "unserer Zeitrechnung";
    }
    "eraAbbr";
    {
        "0";
        "v. Chr.",
        "0-alt-variant";
        "v. u. Z.",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "u. Z.";
    }
    "eraNarrow";
    {

```

```

        "0";
        "v. Chr.",
        "0-alt-variant";
        "v. u. Z.",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "u. Z.";
    }
}
"dateFormats";
{
    "full";
    "EEEE, d. MMMM y",
    "long";
    "d. MMMM y",
    "medium";
    "dd.MM.y",
    "short";
    "dd.MM.yy";
}
"timeFormats";
{
    "full";
    "HH:mm:ss zzzz",
    "long";
    "HH:mm:ss z",
    "medium";
    "HH:mm:ss",
    "short";
    "HH:mm";
}
"dateTimeFormats";
{
    "full";
    "{1} 'um' {0}",
    "long";
    "{1} 'um' {0}",
    "medium";
    "{1}, {0}",
    "short";
    "{1}, {0}",
    "availableFormats";
    {
        "d";
        "d",
        "E";
        "ccc",
        "Ed";
        "E, d.",
        "Ehm";
        "E h:mm a",
        "EHm";
        "E, HH:mm",
        "Ehms";
        "E, h:mm:ss a",
        "EHms";
    }
}

```

```

"E, HH:mm:ss",
  "Gy";
"y G",
  "GyMMM";
"MMM y G",
  "GyMMMd";
"d. MMM y G",
  "GyMMMED";
"E, d. MMM y G",
  "h";
"h 'Uhr' a",
  "H";
"HH 'Uhr'",
  "hm";
"h:mm a",
  "Hm";
"HH:mm",
  "hms";
"h:mm:ss a",
  "Hms";
"HH:mm:ss",
  "hmsv";
"h:mm:ss a v",
  "Hmsv";
"HH:mm:ss v",
  "hmv";
"h:mm a v",
  "Hmv";
"HH:mm v",
  "M";
"L",
  "Md";
"d.M.",
  "MEd";
"E, d.M.",
  "MMd";
"d.MM.",
  "MMdd";
"dd.MM.",
  "MMM";
"LLL",
  "MMMd";
"d. MMM",
  "MMMED";
"E, d. MMM",
  "MMMMd";
"d. MMMM",
  "MMMMEd";
"E, d. MMMM",
  "MMMMW";
"'Woche' W 'im' MMM",
  "MMMMW";
"'Woche' W 'im' MMM",
  "ms";
"mm:ss",
  "y";
"y",

```

```

        "yM";
        "M.y",
        "yMd";
        "d.M.y",
        "yMEd";
        "E, d.M.y",
        "yMM";
        "MM.y",
        "yMMdd";
        "dd.MM.y",
        "yMMM";
        "MMM y",
        "yMMMd";
        "d. MMM y",
        "yMMMEd";
        "E, d. MMM y",
        "yMMMM";
        "MMMM y",
        "yQQQ";
        "QQQ y",
        "yQQQQ";
        "QQQQ y",
        "yw";
        "'Woche' w 'des' 'Jahres' y",
        "yw";
        "'Woche' w 'des' 'Jahres' y";
    }
    "appendItems";
    {
        "Day";
        "{0} ({2}: {1})",
        "Day-Of-Week";
        "{0} {1}",
        "Era";
        "{1} {0}",
        "Hour";
        "{0} ({2}: {1})",
        "Minute";
        "{0} ({2}: {1})",
        "Month";
        "{0} ({2}: {1})",
        "Quarter";
        "{0} ({2}: {1})",
        "Second";
        "{0} ({2}: {1})",
        "Timezone";
        "{0} {1}",
        "Week";
        "{0} ({2}: {1})",
        "Year";
        "{1} {0}";
    }
    "intervalFormats";
    {
        "intervalFormatFallback";
        "{0} - {1}",
        "d";
    }

```

```

{
    "d";
    "d.-d.";
}
"h";
{
    "a";
    "h 'Uhr' a - h 'Uhr' a",
    "h";
    "h - h 'Uhr' a";
}
"H";
{
    "H";
    "HH-HH 'Uhr'";
}
"hm";
{
    "a";
    "h:mm a - h:mm a",
    "h";
    "h:mm-h:mm a",
    "m";
    "h:mm-h:mm a";
}
"Hm";
{
    "H";
    "HH:mm-HH:mm 'Uhr'",
    "m";
    "HH:mm-HH:mm 'Uhr'";
}
"hm v";
{
    "a";
    "h:mm a - h:mm a v",
    "h";
    "h:mm-h:mm a v",
    "m";
    "h:mm-h:mm a v";
}
"Hm v";
{
    "H";
    "HH:mm-HH:mm 'Uhr' v",
    "m";
    "HH:mm-HH:mm 'Uhr' v";
}
"hv";
{
    "a";
    "h a - h a v",
    "h";
    "h-h a v";
}
"Hv";
{

```

```

        "H";
        "HH-HH 'Uhr' v";
    }
    "M";
    {
        "M";
        "M.-M.";
    }
    "Md";
    {
        "d";
        "dd.MM. - dd.MM.",
        "M";
        "dd.MM. - dd.MM.";
    }
    "MEd";
    {
        "d";
        "E, dd.MM. - E, dd.MM.",
        "M";
        "E, dd.MM. - E, dd.MM.";
    }
    "MMM";
    {
        "M";
        "MMM-MMM";
    }
    "MMMd";
    {
        "d";
        "d.-d. MMM",
        "M";
        "d. MMM - d. MMM";
    }
    "MMMEd";
    {
        "d";
        "E, d. - E, d. MMM",
        "M";
        "E, d. MMM - E, d. MMM";
    }
    "MMMM";
    {
        "M";
        "LLLL-LLLL";
    }
    "Y";
    {
        "Y";
        "Y-Y";
    }
    "YM";
    {
        "M";
        "MM.Y - MM.Y",
        "Y";
        "MM.Y - MM.Y";
    }

```



```

"yMd";
{
    "d";
    "dd.MM.y - dd.MM.y",
    "M";
    "dd.MM.y - dd.MM.y",
    "Y";
    "dd.MM.y - dd.MM.y";
}
"yMEd";
{
    "d";
    "E, dd.MM.y - E, dd.MM.y",
    "M";
    "E, dd.MM.y - E, dd.MM.y",
    "Y";
    "E, dd.MM.y - E, dd.MM.y";
}
"yMMM";
{
    "M";
    "MMM-MMM y",
    "Y";
    "MMM y - MMM y";
}
"yMMMd";
{
    "d";
    "d.-d. MMM y",
    "M";
    "d. MMM - d. MMM y",
    "Y";
    "d. MMM y - d. MMM y";
}
"yMMMEd";
{
    "d";
    "E, d. - E, d. MMM y",
    "M";
    "E, d. MMM - E, d. MMM y",
    "Y";
    "E, d. MMM y - E, d. MMM y";
}
"yMMMM";
{
    "M";
    "MMMM-MMMM y",
    "Y";
    "MMMM y - MMMM y";
}

```

```
}
}
```

CURRENCIES.JSON

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 13259 $",
          "_cldrVersion": "31"
        },
        "language": "de"
      },
      "numbers": {
        "currencies": {
          "ADP": {
            "displayName": "Andorranische Pesete",
            "displayName-count-one": "Andorranische Pesete",
            "displayName-count-other": "Andorranische Peseten",
            "symbol": "ADP"
          },
          "AED": {
            "displayName": "VAE-Dirham",
            "displayName-count-one": "VAE-Dirham",
            "displayName-count-other": "VAE-Dirham",
            "symbol": "AED"
          },
          "AFA": {
            "displayName": "Afghanische Afghani (1927-2002)",
            "displayName-count-one": "Afghanische Afghani (1927-2002)",
            "displayName-count-other": "Afghanische Afghani (1927-2002)",
            "symbol": "AFA"
          },
          "AFN": {
            "displayName": "Afghanischer Afghani",
            "displayName-count-one": "Afghanischer Afghani",
            "displayName-count-other": "Afghanische Afghani",
            "symbol": "AFN"
          },
          "ALK": {
            "displayName": "Albanischer Lek (1946-1965)",
            "displayName-count-one": "Albanischer Lek (1946-1965)",
            "displayName-count-other": "Albanische Lek (1946-1965)"
          },
          "ALL": {
            "displayName": "Albanischer Lek",
            "displayName-count-one": "Albanischer Lek",
            "displayName-count-other": "Albanische Lek",
            "symbol": "ALL"
          },
          "AMD": {
            "displayName": "Armenischer Dram",
            "displayName-count-one": "Armenischer Dram",
            "displayName-count-other": "Armenische Dram",
```

```

        "symbol": "AMD"
    },
    "ANG": {
        "displayName": "Niederländische-Antillen-Gulden",
        "displayName-count-one": "Niederländische-Antillen-Gulden",
        "displayName-count-other": "Niederländische-Antillen-Gulden",
        "symbol": "ANG"
    },
    "AOA": {
        "displayName": "Angolanischer Kwanza",
        "displayName-count-one": "Angolanischer Kwanza",
        "displayName-count-other": "Angolanische Kwanza",
        "symbol": "AOA",
        "symbol-alt-narrow": "Kz"
    },
    "AOK": {
        "displayName": "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one": "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other": "Angolanische Kwanza (1977-1990)",
        "symbol": "AOK"
    },
    "AON": {
        "displayName": "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one": "Angolanischer Neuer Kwanza (1990-
2000)",
        "displayName-count-other": "Angolanische Neue Kwanza (1990-
2000)",
        "symbol": "AON"
    },
    "AOR": {
        "displayName": "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one": "Angolanischer Kwanza Reajustado (1995-
1999)",
        "displayName-count-other": "Angolanische Kwanza Reajustado
(1995-1999)",
        "symbol": "AOR"
    },
    "ARA": {
        "displayName": "Argentinischer Austral",
        "displayName-count-one": "Argentinischer Austral",
        "displayName-count-other": "Argentinische Austral",
        "symbol": "ARA"
    },
    "ARL": {
        "displayName": "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one": "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other": "Argentinische Pesos Ley (1970-
1983)",
        "symbol": "ARL"
    },
    "ARM": {
        "displayName": "Argentinischer Peso (1881-1970)",
        "displayName-count-one": "Argentinischer Peso (1881-1970)",
        "displayName-count-other": "Argentinische Pesos (1881-1970)",
        "symbol": "ARM"
    },
    "ARP": {

```

```

        "displayName": "Argentinischer Peso (1983-1985)",
        "displayName-count-one": "Argentinischer Peso (1983-1985)",
        "displayName-count-other": "Argentinische Peso (1983-1985)",
        "symbol": "ARP"
    },
    "ARS": {
        "displayName": "Argentinischer Peso",
        "displayName-count-one": "Argentinischer Peso",
        "displayName-count-other": "Argentinische Pesos",
        "symbol": "ARS",
        "symbol-alt-narrow": "$"
    },
    "ATS": {
        "displayName": "Österreichischer Schilling",
        "displayName-count-one": "Österreichischer Schilling",
        "displayName-count-other": "Österreichische Schilling",
        "symbol": "öS"
    },
    "AUD": {
        "displayName": "Australischer Dollar",
        "displayName-count-one": "Australischer Dollar",
        "displayName-count-other": "Australische Dollar",
        "symbol": "AU$",
        "symbol-alt-narrow": "$"
    },
    "AWG": {
        "displayName": "Aruba-Florin",
        "displayName-count-one": "Aruba-Florin",
        "displayName-count-other": "Aruba-Florin",
        "symbol": "AWG"
    },
    "AZM": {
        "displayName": "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-one": "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-other": "Aserbaidtschan-Manat (1993-2006)",
        "symbol": "AZM"
    },
    "AZN": {
        "displayName": "Aserbaidtschan-Manat",
        "displayName-count-one": "Aserbaidtschan-Manat",
        "displayName-count-other": "Aserbaidtschan-Manat",
        "symbol": "AZN"
    },
    "BAD": {
        "displayName": "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-one": "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-other": "Bosnien und Herzegowina Dinar (1992-1994)",
        "symbol": "BAD"
    },
    "BAM": {
        "displayName": "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one": "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-other": "Bosnien und Herzegowina Konvertierbare Mark",
    }
}

```

```

        "symbol": "BAM",
        "symbol-alt-narrow": "KM"
    },
    "BAN": {
        "displayName": "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-one": "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-other": "Bosnien und Herzegowina Neue Dinar (1994-1997)",
        "symbol": "BAN"
    },
    "BBD": {
        "displayName": "Barbados-Dollar",
        "displayName-count-one": "Barbados-Dollar",
        "displayName-count-other": "Barbados-Dollar",
        "symbol": "BBD",
        "symbol-alt-narrow": "$"
    },
    "BDT": {
        "displayName": "Bangladesch-Taka",
        "displayName-count-one": "Bangladesch-Taka",
        "displayName-count-other": "Bangladesch-Taka",
        "symbol": "BDT",
        "symbol-alt-narrow": "ট"
    },
    "BEC": {
        "displayName": "Belgischer Franc (konvertibel)",
        "displayName-count-one": "Belgischer Franc (konvertibel)",
        "displayName-count-other": "Belgische Franc (konvertibel)",
        "symbol": "BEC"
    },
    "BEF": {
        "displayName": "Belgischer Franc",
        "displayName-count-one": "Belgischer Franc",
        "displayName-count-other": "Belgische Franc",
        "symbol": "BEF"
    },
    "BEL": {
        "displayName": "Belgischer Finanz-Franc",
        "displayName-count-one": "Belgischer Finanz-Franc",
        "displayName-count-other": "Belgische Finanz-Franc",
        "symbol": "BEL"
    },
    "BGL": {
        "displayName": "Bulgarische Lew (1962-1999)",
        "displayName-count-one": "Bulgarische Lew (1962-1999)",
        "displayName-count-other": "Bulgarische Lew (1962-1999)",
        "symbol": "BGL"
    },
    "BGM": {
        "displayName": "Bulgarischer Lew (1952-1962)",
        "displayName-count-one": "Bulgarischer Lew (1952-1962)",
        "displayName-count-other": "Bulgarische Lew (1952-1962)",
        "symbol": "BGK"
    },
    },

```

```

    "BGN": {
      "displayName": "Bulgarischer Lew",
      "displayName-count-one": "Bulgarischer Lew",
      "displayName-count-other": "Bulgarische Lew",
      "symbol": "BGN"
    },
    "BGO": {
      "displayName": "Bulgarischer Lew (1879-1952)",
      "displayName-count-one": "Bulgarischer Lew (1879-1952)",
      "displayName-count-other": "Bulgarische Lew (1879-1952)",
      "symbol": "BGJ"
    },
    "BHD": {
      "displayName": "Bahrain-Dinar",
      "displayName-count-one": "Bahrain-Dinar",
      "displayName-count-other": "Bahrain-Dinar",
      "symbol": "BHD"
    },
    "BIF": {
      "displayName": "Burundi-Franc",
      "displayName-count-one": "Burundi-Franc",
      "displayName-count-other": "Burundi-Francs",
      "symbol": "BIF"
    },
    "BMD": {
      "displayName": "Bermuda-Dollar",
      "displayName-count-one": "Bermuda-Dollar",
      "displayName-count-other": "Bermuda-Dollar",
      "symbol": "BMD",
      "symbol-alt-narrow": "$"
    },
    "BND": {
      "displayName": "Brunei-Dollar",
      "displayName-count-one": "Brunei-Dollar",
      "displayName-count-other": "Brunei-Dollar",
      "symbol": "BND",
      "symbol-alt-narrow": "$"
    },
    "BOB": {
      "displayName": "Bolivanischer Boliviano",
      "displayName-count-one": "Bolivanischer Boliviano",
      "displayName-count-other": "Bolivianische Bolivianos",
      "symbol": "BOB",
      "symbol-alt-narrow": "Bs"
    },
    "BOL": {
      "displayName": "Bolivianischer Boliviano (1863-1963)",
      "displayName-count-one": "Bolivianischer Boliviano (1863-1963)",
      "displayName-count-other": "Bolivianische Bolivianos (1863-
1963)",
      "symbol": "BOL"
    },
    "BOP": {
      "displayName": "Bolivianischer Peso",
      "displayName-count-one": "Bolivianischer Peso",
      "displayName-count-other": "Bolivianische Peso",
      "symbol": "BOP"
    }
  }

```

```

    },
    "BOV": {
      "displayName": "Boliviansiche Mvdol",
      "displayName-count-one": "Boliviansiche Mvdol",
      "displayName-count-other": "Bolivianische Mvdol",
      "symbol": "BOV"
    },
    "BRB": {
      "displayName": "Brasilianischer Cruzeiro Novo (1967-1986)",
      "displayName-count-one": "Brasilianischer Cruzeiro Novo (1967-
1986)",
      "displayName-count-other": "Brasilianische Cruzeiro Novo (1967-
1986)",
      "symbol": "BRB"
    },
    "BRC": {
      "displayName": "Brasilianischer Cruzado (1986-1989)",
      "displayName-count-one": "Brasilianischer Cruzado (1986-1989)",
      "displayName-count-other": "Brasilianische Cruzado (1986-1989)",
      "symbol": "BRC"
    },
    "BRE": {
      "displayName": "Brasilianischer Cruzeiro (1990-1993)",
      "displayName-count-one": "Brasilianischer Cruzeiro (1990-1993)",
      "displayName-count-other": "Brasilianische Cruzeiro (1990-
1993)",
      "symbol": "BRE"
    },
    "BRL": {
      "displayName": "Brasilianischer Real",
      "displayName-count-one": "Brasilianischer Real",
      "displayName-count-other": "Brasilianische Real",
      "symbol": "R$",
      "symbol-alt-narrow": "R$"
    },
    "BRN": {
      "displayName": "Brasilianischer Cruzado Novo (1989-1990)",
      "displayName-count-one": "Brasilianischer Cruzado Novo (1989-
1990)",
      "displayName-count-other": "Brasilianische Cruzado Novo (1989-
1990)",
      "symbol": "BRN"
    },
    "BRR": {
      "displayName": "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-one": "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-other": "Brasilianische Cruzeiro (1993-
1994)",
      "symbol": "BRR"
    },
    "BRZ": {
      "displayName": "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-one": "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-other": "Brasilianischer Cruzeiro (1942-
1967)",
      "symbol": "BRZ"
    }
  },

```

```

"BSD": {
  "displayName": "Bahamas-Dollar",
  "displayName-count-one": "Bahamas-Dollar",
  "displayName-count-other": "Bahamas-Dollar",
  "symbol": "BSD",
  "symbol-alt-narrow": "$"
},
"BTN": {
  "displayName": "Bhutan-Ngultrum",
  "displayName-count-one": "Bhutan-Ngultrum",
  "displayName-count-other": "Bhutan-Ngultrum",
  "symbol": "BTN"
},
"BUK": {
  "displayName": "Birmanischer Kyat",
  "displayName-count-one": "Birmanischer Kyat",
  "displayName-count-other": "Birmanische Kyat",
  "symbol": "BUK"
},
"BWP": {
  "displayName": "Botswanischer Pula",
  "displayName-count-one": "Botswanischer Pula",
  "displayName-count-other": "Botswanische Pula",
  "symbol": "BWP",
  "symbol-alt-narrow": "P"
},
"BYB": {
  "displayName": "Belarus-Rubel (1994-1999)",
  "displayName-count-one": "Belarus-Rubel (1994-1999)",
  "displayName-count-other": "Belarus-Rubel (1994-1999)",
  "symbol": "BYB"
},
"BYN": {
  "displayName": "Weißrussischer Rubel",
  "displayName-count-one": "Weißrussischer Rubel",
  "displayName-count-other": "Weißrussische Rubel",
  "symbol": "BYN",
  "symbol-alt-narrow": "p."
},
"BYR": {
  "displayName": "Weißrussischer Rubel (2000-2016)",
  "displayName-count-one": "Weißrussischer Rubel (2000-2016)",
  "displayName-count-other": "Weißrussische Rubel (2000-2016)",
  "symbol": "BYR"
},
"BZD": {
  "displayName": "Belize-Dollar",
  "displayName-count-one": "Belize-Dollar",
  "displayName-count-other": "Belize-Dollar",
  "symbol": "BZD",
  "symbol-alt-narrow": "$"
},
"CAD": {
  "displayName": "Kanadischer Dollar",
  "displayName-count-one": "Kanadischer Dollar",
  "displayName-count-other": "Kanadische Dollar",
  "symbol": "CA$",

```



```

        "symbol-alt-narrow": "$"
    },
    "CDF": {
        "displayName": "Kongo-Franc",
        "displayName-count-one": "Kongo-Franc",
        "displayName-count-other": "Kongo-Francs",
        "symbol": "CDF"
    },
    "CHE": {
        "displayName": "WIR-Euro",
        "displayName-count-one": "WIR-Euro",
        "displayName-count-other": "WIR-Euro",
        "symbol": "CHE"
    },
    "CHF": {
        "displayName": "Schweizer Franken",
        "displayName-count-one": "Schweizer Franken",
        "displayName-count-other": "Schweizer Franken",
        "symbol": "CHF"
    },
    "CHW": {
        "displayName": "WIR Franken",
        "displayName-count-one": "WIR Franken",
        "displayName-count-other": "WIR Franken",
        "symbol": "CHW"
    },
    "CLE": {
        "displayName": "Chilenischer Escudo",
        "displayName-count-one": "Chilenischer Escudo",
        "displayName-count-other": "Chilenische Escudo",
        "symbol": "CLE"
    },
    "CLF": {
        "displayName": "Chilenische Unidades de Fomento",
        "displayName-count-one": "Chilenische Unidades de Fomento",
        "displayName-count-other": "Chilenische Unidades de Fomento",
        "symbol": "CLF"
    },
    "CLP": {
        "displayName": "Chilenischer Peso",
        "displayName-count-one": "Chilenischer Peso",
        "displayName-count-other": "Chilenische Pesos",
        "symbol": "CLP",
        "symbol-alt-narrow": "$"
    },
    "CNX": {
        "displayName": "Dollar der Chinesischen Volksbank",
        "displayName-count-one": "Dollar der Chinesischen Volksbank",
        "displayName-count-other": "Dollar der Chinesischen Volksbank",
        "symbol": "CNX"
    },
    "CNY": {
        "displayName": "Renminbi Yuan",
        "displayName-count-one": "Chinesischer Yuan",
        "displayName-count-other": "Renminbi Yuan",
        "symbol": "CN¥",
        "symbol-alt-narrow": "¥"
    }

```

```

    },
    "COP": {
      "displayName": "Kolumbianischer Peso",
      "displayName-count-one": "Kolumbianischer Peso",
      "displayName-count-other": "Kolumbianische Pesos",
      "symbol": "COP",
      "symbol-alt-narrow": "$"
    },
    "COU": {
      "displayName": "Kolumbianische Unidades de valor real",
      "displayName-count-one": "Kolumbianische Unidad de valor real",
      "displayName-count-other": "Kolumbianische Unidades de valor
real",
      "symbol": "COU"
    },
    "CRC": {
      "displayName": "Costa-Rica-Colón",
      "displayName-count-one": "Costa-Rica-Colón",
      "displayName-count-other": "Costa-Rica-Colón",
      "symbol": "CRC",
      "symbol-alt-narrow": "₡"
    },
    "CSD": {
      "displayName": "Serbischer Dinar (2002-2006)",
      "displayName-count-one": "Serbischer Dinar (2002-2006)",
      "displayName-count-other": "Serbische Dinar (2002-2006)",
      "symbol": "CSD"
    },
    "CSK": {
      "displayName": "Tschechoslowakische Krone",
      "displayName-count-one": "Tschechoslowakische Kronen",
      "displayName-count-other": "Tschechoslowakische Kronen",
      "symbol": "CSK"
    },
    "CUC": {
      "displayName": "Kubanischer Peso (konvertibel)",
      "displayName-count-one": "Kubanischer Peso (konvertibel)",
      "displayName-count-other": "Kubanische Pesos (konvertibel)",
      "symbol": "CUC",
      "symbol-alt-narrow": "Cub$"
    },
    "CUP": {
      "displayName": "Kubanischer Peso",
      "displayName-count-one": "Kubanischer Peso",
      "displayName-count-other": "Kubanische Pesos",
      "symbol": "CUP",
      "symbol-alt-narrow": "$"
    },
    "CVE": {
      "displayName": "Cabo-Verde-Escudo",
      "displayName-count-one": "Cabo-Verde-Escudo",
      "displayName-count-other": "Cabo-Verde-Escudos",
      "symbol": "CVE"
    },
    "CYP": {
      "displayName": "Zypern-Pfund",
      "displayName-count-one": "Zypern Pfund",

```

```

    "displayName-count-other": "Zypern Pfund",
    "symbol": "CYP"
  },
  "CZK": {
    "displayName": "Tschechische Krone",
    "displayName-count-one": "Tschechische Krone",
    "displayName-count-other": "Tschechische Kronen",
    "symbol": "CZK",
    "symbol-alt-narrow": "Kč"
  },
  "DDM": {
    "displayName": "Mark der DDR",
    "displayName-count-one": "Mark der DDR",
    "displayName-count-other": "Mark der DDR",
    "symbol": "DDM"
  },
  "DEM": {
    "displayName": "Deutsche Mark",
    "displayName-count-one": "Deutsche Mark",
    "displayName-count-other": "Deutsche Mark",
    "symbol": "DM"
  },
  "DJF": {
    "displayName": "Dschibuti-Franc",
    "displayName-count-one": "Dschibuti-Franc",
    "displayName-count-other": "Dschibuti-Franc",
    "symbol": "DJF"
  },
  "DKK": {
    "displayName": "Dänische Krone",
    "displayName-count-one": "Dänische Krone",
    "displayName-count-other": "Dänische Kronen",
    "symbol": "DKK",
    "symbol-alt-narrow": "kr"
  },
  "DOP": {
    "displayName": "Dominikanischer Peso",
    "displayName-count-one": "Dominikanischer Peso",
    "displayName-count-other": "Dominikanische Pesos",
    "symbol": "DOP",
    "symbol-alt-narrow": "$"
  },
  "DZD": {
    "displayName": "Algerischer Dinar",
    "displayName-count-one": "Algerischer Dinar",
    "displayName-count-other": "Algerische Dinar",
    "symbol": "DZD"
  },
  "ECS": {
    "displayName": "Ecuadorianischer Sucre",
    "displayName-count-one": "Ecuadorianischer Sucre",
    "displayName-count-other": "Ecuadorianische Sucre",
    "symbol": "ECS"
  },
  "ECV": {
    "displayName": "Verrechnungseinheit für Ecuador",
    "displayName-count-one": "Verrechnungseinheiten für Ecuador",

```

```

        "displayName-count-other": "Verrechnungseinheiten für Ecuador",
        "symbol": "ECV"
    },
    "EEK": {
        "displayName": "Estnische Krone",
        "displayName-count-one": "Estnische Krone",
        "displayName-count-other": "Estnische Kronen",
        "symbol": "EEK"
    },
    "EGP": {
        "displayName": "Ägyptisches Pfund",
        "displayName-count-one": "Ägyptisches Pfund",
        "displayName-count-other": "Ägyptische Pfund",
        "symbol": "EGP",
        "symbol-alt-narrow": "E£"
    },
    "ERN": {
        "displayName": "Eritreischer Nakfa",
        "displayName-count-one": "Eritreischer Nakfa",
        "displayName-count-other": "Eritreische Nakfa",
        "symbol": "ERN"
    },
    "ESA": {
        "displayName": "Spanische Peseta (A-Konten)",
        "displayName-count-one": "Spanische Peseta (A-Konten)",
        "displayName-count-other": "Spanische Peseten (A-Konten)",
        "symbol": "ESA"
    },
    "ESB": {
        "displayName": "Spanische Peseta (konvertibel)",
        "displayName-count-one": "Spanische Peseta (konvertibel)",
        "displayName-count-other": "Spanische Peseten (konvertibel)",
        "symbol": "ESB"
    },
    "ESP": {
        "displayName": "Spanische Peseta",
        "displayName-count-one": "Spanische Peseta",
        "displayName-count-other": "Spanische Peseten",
        "symbol": "ESP",
        "symbol-alt-narrow": "₧"
    },
    "ETB": {
        "displayName": "Äthiopischer Birr",
        "displayName-count-one": "Äthiopischer Birr",
        "displayName-count-other": "Äthiopische Birr",
        "symbol": "ETB"
    },
    "EUR": {
        "displayName": "Euro",
        "displayName-count-one": "Euro",
        "displayName-count-other": "Euro",
        "symbol": "€",
        "symbol-alt-narrow": "€"
    },
    "FIM": {
        "displayName": "Finnische Mark",
        "displayName-count-one": "Finnische Mark",

```

```

        "displayName-count-other": "Finnische Mark",
        "symbol": "FIM"
    },
    "FJD": {
        "displayName": "Fidschi-Dollar",
        "displayName-count-one": "Fidschi-Dollar",
        "displayName-count-other": "Fidschi-Dollar",
        "symbol": "FJD",
        "symbol-alt-narrow": "$"
    },
    "FKP": {
        "displayName": "Falkland-Pfund",
        "displayName-count-one": "Falkland-Pfund",
        "displayName-count-other": "Falkland-Pfund",
        "symbol": "FKP",
        "symbol-alt-narrow": "Fl£"
    },
    "FRF": {
        "displayName": "Französischer Franc",
        "displayName-count-one": "Französischer Franc",
        "displayName-count-other": "Französische Franc",
        "symbol": "FRF"
    },
    "GBP": {
        "displayName": "Britisches Pfund",
        "displayName-count-one": "Britisches Pfund",
        "displayName-count-other": "Britische Pfund",
        "symbol": "£",
        "symbol-alt-narrow": "£"
    },
    "GEC": {
        "displayName": "Georgischer Kupon Larit",
        "displayName-count-one": "Georgischer Kupon Larit",
        "displayName-count-other": "Georgische Kupon Larit",
        "symbol": "GEC"
    },
    "GEL": {
        "displayName": "Georgischer Lari",
        "displayName-count-one": "Georgischer Lari",
        "displayName-count-other": "Georgische Lari",
        "symbol": "GEL",
        "symbol-alt-narrow": "ლ",
        "symbol-alt-variant": "ლ"
    },
    "GHC": {
        "displayName": "Ghanaischer Cedi (1979-2007)",
        "displayName-count-one": "Ghanaischer Cedi (1979-2007)",
        "displayName-count-other": "Ghanaische Cedi (1979-2007)",
        "symbol": "GHC"
    },
    "GHS": {
        "displayName": "Ghanaischer Cedi",
        "displayName-count-one": "Ghanaischer Cedi",
        "displayName-count-other": "Ghanaische Cedi",
        "symbol": "GHS"
    },
    "GIP": {

```

```

        "displayName": "Gibraltar-Pfund",
        "displayName-count-one": "Gibraltar-Pfund",
        "displayName-count-other": "Gibraltar Pfund",
        "symbol": "GIP",
        "symbol-alt-narrow": "£"
    },
    "GMD": {
        "displayName": "Gambia-Dalasi",
        "displayName-count-one": "Gambia-Dalasi",
        "displayName-count-other": "Gambia-Dalasi",
        "symbol": "GMD"
    },
    "GNF": {
        "displayName": "Guinea-Franc",
        "displayName-count-one": "Guinea-Franc",
        "displayName-count-other": "Guinea-Franc",
        "symbol": "GNF",
        "symbol-alt-narrow": "F.G."
    },
    "GNS": {
        "displayName": "Guineischer Syli",
        "displayName-count-one": "Guineischer Syli",
        "displayName-count-other": "Guineische Syli",
        "symbol": "GNS"
    },
    "GQE": {
        "displayName": "Äquatorialguinea-Ekwele",
        "displayName-count-one": "Äquatorialguinea-Ekwele",
        "displayName-count-other": "Äquatorialguinea-Ekwele",
        "symbol": "GQE"
    },
    "GRD": {
        "displayName": "Griechische Drachme",
        "displayName-count-one": "Griechische Drachme",
        "displayName-count-other": "Griechische Drachmen",
        "symbol": "GRD"
    },
    "GTQ": {
        "displayName": "Guatemaltekinscher Quetzal",
        "displayName-count-one": "Guatemaltekinscher Quetzal",
        "displayName-count-other": "Guatemaltekinsche Quetzales",
        "symbol": "GTQ",
        "symbol-alt-narrow": "Q"
    },
    "GWE": {
        "displayName": "Portugiesisch Guinea Escudo",
        "displayName-count-one": "Portugiesisch Guinea Escudo",
        "displayName-count-other": "Portugiesisch Guinea Escudo",
        "symbol": "GWE"
    },
    "GWP": {
        "displayName": "Guinea-Bissau Peso",
        "displayName-count-one": "Guinea-Bissau Peso",
        "displayName-count-other": "Guinea-Bissau Pesos",
        "symbol": "GWP"
    },
    "GYD": {

```

```

    "displayName": "Guyana-Dollar",
    "displayName-count-one": "Guyana-Dollar",
    "displayName-count-other": "Guyana-Dollar",
    "symbol": "GYD",
    "symbol-alt-narrow": "$"
  },
  "HKD": {
    "displayName": "Hongkong-Dollar",
    "displayName-count-one": "Hongkong-Dollar",
    "displayName-count-other": "Hongkong-Dollar",
    "symbol": "HK$",
    "symbol-alt-narrow": "$"
  },
  "HNL": {
    "displayName": "Honduras-Lempira",
    "displayName-count-one": "Honduras-Lempira",
    "displayName-count-other": "Honduras-Lempira",
    "symbol": "HNL",
    "symbol-alt-narrow": "L"
  },
  "HRD": {
    "displayName": "Kroatischer Dinar",
    "displayName-count-one": "Kroatischer Dinar",
    "displayName-count-other": "Kroatische Dinar",
    "symbol": "HRD"
  },
  "HRK": {
    "displayName": "Kroatischer Kuna",
    "displayName-count-one": "Kroatischer Kuna",
    "displayName-count-other": "Kroatische Kuna",
    "symbol": "HRK",
    "symbol-alt-narrow": "kn"
  },
  "HTG": {
    "displayName": "Haitianische Gourde",
    "displayName-count-one": "Haitianische Gourde",
    "displayName-count-other": "Haitianische Gourdes",
    "symbol": "HTG"
  },
  "HUF": {
    "displayName": "Ungarischer Forint",
    "displayName-count-one": "Ungarischer Forint",
    "displayName-count-other": "Ungarische Forint",
    "symbol": "HUF",
    "symbol-alt-narrow": "Ft"
  },
  "IDR": {
    "displayName": "Indonesische Rupiah",
    "displayName-count-one": "Indonesische Rupiah",
    "displayName-count-other": "Indonesische Rupiah",
    "symbol": "IDR",
    "symbol-alt-narrow": "Rp"
  },
  "IEP": {
    "displayName": "Irisches Pfund",
    "displayName-count-one": "Irisches Pfund",
    "displayName-count-other": "Irische Pfund",

```

```

        "symbol": "IEP"
    },
    "ILP": {
        "displayName": "Israelisches Pfund",
        "displayName-count-one": "Israelisches Pfund",
        "displayName-count-other": "Israelische Pfund",
        "symbol": "ILP"
    },
    "ILR": {
        "displayName": "Israelischer Schekel (1980-1985)",
        "displayName-count-one": "Israelischer Schekel (1980-1985)",
        "displayName-count-other": "Israelische Schekel (1980-1985)"
    },
    "ILS": {
        "displayName": "Israelischer Neuer Schekel",
        "displayName-count-one": "Israelischer Neuer Schekel",
        "displayName-count-other": "Israelische Neue Schekel",
        "symbol": "₪",
        "symbol-alt-narrow": "₪"
    },
    "INR": {
        "displayName": "Indische Rupie",
        "displayName-count-one": "Indische Rupie",
        "displayName-count-other": "Indische Rupien",
        "symbol": "₹",
        "symbol-alt-narrow": "₹"
    },
    "IQD": {
        "displayName": "Irakischer Dinar",
        "displayName-count-one": "Irakischer Dinar",
        "displayName-count-other": "Irakische Dinar",
        "symbol": "IQD"
    },
    "IRR": {
        "displayName": "Iranischer Rial",
        "displayName-count-one": "Iranischer Rial",
        "displayName-count-other": "Iranische Rial",
        "symbol": "IRR"
    },
    "ISJ": {
        "displayName": "Isländische Krone (1918-1981)",
        "displayName-count-one": "Isländische Krone (1918-1981)",
        "displayName-count-other": "Isländische Kronen (1918-1981)"
    },
    "ISK": {
        "displayName": "Isländische Krone",
        "displayName-count-one": "Isländische Krone",
        "displayName-count-other": "Isländische Kronen",
        "symbol": "ISK",
        "symbol-alt-narrow": "kr"
    },
    "ITL": {
        "displayName": "Italienische Lira",
        "displayName-count-one": "Italienische Lira",
        "displayName-count-other": "Italienische Lire",
        "symbol": "ITL"
    },

```



```

    "JMD": {
      "displayName": "Jamaika-Dollar",
      "displayName-count-one": "Jamaika-Dollar",
      "displayName-count-other": "Jamaika-Dollar",
      "symbol": "JMD",
      "symbol-alt-narrow": "$"
    },
    "JOD": {
      "displayName": "Jordanischer Dinar",
      "displayName-count-one": "Jordanischer Dinar",
      "displayName-count-other": "Jordanische Dinar",
      "symbol": "JOD"
    },
    "JPY": {
      "displayName": "Japanischer Yen",
      "displayName-count-one": "Japanischer Yen",
      "displayName-count-other": "Japanische Yen",
      "symbol": "¥",
      "symbol-alt-narrow": "¥"
    },
    "KES": {
      "displayName": "Kenia-Schilling",
      "displayName-count-one": "Kenia-Schilling",
      "displayName-count-other": "Kenia-Schilling",
      "symbol": "KES"
    },
    "KGS": {
      "displayName": "Kirgisischer Som",
      "displayName-count-one": "Kirgisischer Som",
      "displayName-count-other": "Kirgisische Som",
      "symbol": "KGS"
    },
    "KHR": {
      "displayName": "Kambodschanischer Riel",
      "displayName-count-one": "Kambodschanischer Riel",
      "displayName-count-other": "Kambodschanische Riel",
      "symbol": "KHR",
      "symbol-alt-narrow": "៛"
    },
    "KMF": {
      "displayName": "Komoren-Franc",
      "displayName-count-one": "Komoren-Franc",
      "displayName-count-other": "Komoren-Francs",
      "symbol": "KMF",
      "symbol-alt-narrow": "FC"
    },
    "KPW": {
      "displayName": "Nordkoreanischer Won",
      "displayName-count-one": "Nordkoreanischer Won",
      "displayName-count-other": "Nordkoreanische Won",
      "symbol": "KPW",
      "symbol-alt-narrow": "₩"
    },
    "KRH": {
      "displayName": "Südkoreanischer Hwan (1953-1962)",
      "displayName-count-one": "Südkoreanischer Hwan (1953-1962)",

```

```

        "displayName-count-other": "Südkoreanischer Hwan (1953-1962)",
        "symbol": "KRH"
    },
    "KRO": {
        "displayName": "Südkoreanischer Won (1945-1953)",
        "displayName-count-one": "Südkoreanischer Won (1945-1953)",
        "displayName-count-other": "Südkoreanischer Won (1945-1953)",
        "symbol": "KRO"
    },
    "KRW": {
        "displayName": "Südkoreanischer Won",
        "displayName-count-one": "Südkoreanischer Won",
        "displayName-count-other": "Südkoreanische Won",
        "symbol": "₩",
        "symbol-alt-narrow": "₩"
    },
    "KWD": {
        "displayName": "Kuwait-Dinar",
        "displayName-count-one": "Kuwait-Dinar",
        "displayName-count-other": "Kuwait-Dinar",
        "symbol": "KWD"
    },
    "KYD": {
        "displayName": "Kaiman-Dollar",
        "displayName-count-one": "Kaiman-Dollar",
        "displayName-count-other": "Kaiman-Dollar",
        "symbol": "KYD",
        "symbol-alt-narrow": "$"
    },
    "KZT": {
        "displayName": "Kasachischer Tenge",
        "displayName-count-one": "Kasachischer Tenge",
        "displayName-count-other": "Kasachische Tenge",
        "symbol": "KZT",
        "symbol-alt-narrow": "₸"
    },
    "LAK": {
        "displayName": "Laotischer Kip",
        "displayName-count-one": "Laotischer Kip",
        "displayName-count-other": "Laotische Kip",
        "symbol": "LAK",
        "symbol-alt-narrow": "₭"
    },
    "LBP": {
        "displayName": "Libanesisches Pfund",
        "displayName-count-one": "Libanesisches Pfund",
        "displayName-count-other": "Libanesische Pfund",
        "symbol": "LBP",
        "symbol-alt-narrow": "L£"
    },
    "LKR": {
        "displayName": "Sri-Lanka-Rupie",
        "displayName-count-one": "Sri-Lanka-Rupie",
        "displayName-count-other": "Sri-Lanka-Rupien",
        "symbol": "LKR",
        "symbol-alt-narrow": "Rs"
    },
    },

```

```
"LRD": {
  "displayName": "Liberianischer Dollar",
  "displayName-count-one": "Liberianischer Dollar",
  "displayName-count-other": "Liberianische Dollar",
  "symbol": "LRD",
  "symbol-alt-narrow": "$"
},
"LSL": {
  "displayName": "Loti",
  "displayName-count-one": "Loti",
  "displayName-count-other": "Loti",
  "symbol": "LSL"
},
"LTL": {
  "displayName": "Litauischer Litas",
  "displayName-count-one": "Litauischer Litas",
  "displayName-count-other": "Litauische Litas",
  "symbol": "LTL",
  "symbol-alt-narrow": "Lt"
},
"LTT": {
  "displayName": "Litauischer Talonas",
  "displayName-count-one": "Litauische Talonas",
  "displayName-count-other": "Litauische Talonas",
  "symbol": "LTT"
},
"LUC": {
  "displayName": "Luxemburgischer Franc (konvertibel)",
  "displayName-count-one": "Luxemburgische Franc (konvertibel)",
  "displayName-count-other": "Luxemburgische Franc (konvertibel)",
  "symbol": "LUC"
},
"LUF": {
  "displayName": "Luxemburgischer Franc",
  "displayName-count-one": "Luxemburgische Franc",
  "displayName-count-other": "Luxemburgische Franc",
  "symbol": "LUF"
},
"LUL": {
  "displayName": "Luxemburgischer Finanz-Franc",
  "displayName-count-one": "Luxemburgische Finanz-Franc",
  "displayName-count-other": "Luxemburgische Finanz-Franc",
  "symbol": "LUL"
},
"LVL": {
  "displayName": "Lettischer Lats",
  "displayName-count-one": "Lettischer Lats",
  "displayName-count-other": "Lettische Lats",
  "symbol": "LVL",
  "symbol-alt-narrow": "Ls"
},
"LVR": {
  "displayName": "Lettischer Rubel",
  "displayName-count-one": "Lettische Rubel",
  "displayName-count-other": "Lettische Rubel",
  "symbol": "LVR"
},
}
```

```

"LYD": {
  "displayName": "Libyscher Dinar",
  "displayName-count-one": "Libyscher Dinar",
  "displayName-count-other": "Libysche Dinar",
  "symbol": "LYD"
},
"MAD": {
  "displayName": "Marokkanischer Dirham",
  "displayName-count-one": "Marokkanischer Dirham",
  "displayName-count-other": "Marokkanische Dirham",
  "symbol": "MAD"
},
"MAF": {
  "displayName": "Marokkanischer Franc",
  "displayName-count-one": "Marokkanische Franc",
  "displayName-count-other": "Marokkanische Franc",
  "symbol": "MAF"
},
"MCF": {
  "displayName": "Monegassischer Franc",
  "displayName-count-one": "Monegassischer Franc",
  "displayName-count-other": "Monegassische Franc",
  "symbol": "MCF"
},
"MDC": {
  "displayName": "Moldau-Cupon",
  "displayName-count-one": "Moldau-Cupon",
  "displayName-count-other": "Moldau-Cupon",
  "symbol": "MDC"
},
"MDL": {
  "displayName": "Moldau-Leu",
  "displayName-count-one": "Moldau-Leu",
  "displayName-count-other": "Moldau-Leu",
  "symbol": "MDL"
},
"MGA": {
  "displayName": "Madagaskar-Ariary",
  "displayName-count-one": "Madagaskar-Ariary",
  "displayName-count-other": "Madagaskar-Ariary",
  "symbol": "MGA",
  "symbol-alt-narrow": "Ar"
},
"MGF": {
  "displayName": "Madagaskar-Franc",
  "displayName-count-one": "Madagaskar-Franc",
  "displayName-count-other": "Madagaskar-Franc",
  "symbol": "MGF"
},
"MKD": {
  "displayName": "Mazedonischer Denar",
  "displayName-count-one": "Mazedonischer Denar",
  "displayName-count-other": "Mazedonische Denari",
  "symbol": "MKD"
},
"MKN": {
  "displayName": "Mazedonischer Denar (1992-1993)",

```

```

    "displayName-count-one": "Mazedonischer Denar (1992-1993)",
    "displayName-count-other": "Mazedonische Denar (1992-1993)",
    "symbol": "MKD"
  },
  "MLF": {
    "displayName": "Malischer Franc",
    "displayName-count-one": "Malische Franc",
    "displayName-count-other": "Malische Franc",
    "symbol": "MLF"
  },
  "MMK": {
    "displayName": "Myanmarischer Kyat",
    "displayName-count-one": "Myanmarischer Kyat",
    "displayName-count-other": "Myanmarische Kyat",
    "symbol": "MMK",
    "symbol-alt-narrow": "K"
  },
  "MNT": {
    "displayName": "Mongolischer Tögrög",
    "displayName-count-one": "Mongolischer Tögrög",
    "displayName-count-other": "Mongolische Tögrög",
    "symbol": "MNT",
    "symbol-alt-narrow": "₮"
  },
  "MOP": {
    "displayName": "Macao-Pataca",
    "displayName-count-one": "Macao-Pataca",
    "displayName-count-other": "Macao-Pataca",
    "symbol": "MOP"
  },
  "MRO": {
    "displayName": "Mauretanischer Ouguiya",
    "displayName-count-one": "Mauretanischer Ouguiya",
    "displayName-count-other": "Mauretanische Ouguiya",
    "symbol": "MRO"
  },
  "MTL": {
    "displayName": "Maltesische Lira",
    "displayName-count-one": "Maltesische Lira",
    "displayName-count-other": "Maltesische Lira",
    "symbol": "MTL"
  },
  "MTP": {
    "displayName": "Maltesisches Pfund",
    "displayName-count-one": "Maltesische Pfund",
    "displayName-count-other": "Maltesische Pfund",
    "symbol": "MTP"
  },
  "MUR": {
    "displayName": "Mauritius-Rupie",
    "displayName-count-one": "Mauritius-Rupie",
    "displayName-count-other": "Mauritius-Rupien",
    "symbol": "MUR",
    "symbol-alt-narrow": "Rs"
  },
  "MVP": {
    "displayName": "Malediven-Rupie (alt)",

```

```

        "displayName-count-one": "Malediven-Rupie (alt)",
        "displayName-count-other": "Malediven-Rupien (alt)"
    },
    "MVR": {
        "displayName": "Malediven-Rufiyaa",
        "displayName-count-one": "Malediven-Rufiyaa",
        "displayName-count-other": "Malediven-Rupien",
        "symbol": "MVR"
    },
    "MWK": {
        "displayName": "Malawi-Kwacha",
        "displayName-count-one": "Malawi-Kwacha",
        "displayName-count-other": "Malawi-Kwacha",
        "symbol": "MWK"
    },
    "MXN": {
        "displayName": "Mexikanischer Peso",
        "displayName-count-one": "Mexikanischer Peso",
        "displayName-count-other": "Mexikanische Pesos",
        "symbol": "MX$",
        "symbol-alt-narrow": "$"
    },
    "MXP": {
        "displayName": "Mexikanischer Silber-Peso (1861-1992)",
        "displayName-count-one": "Mexikanische Silber-Peso (1861-1992)",
        "displayName-count-other": "Mexikanische Silber-Pesos (1861-
1992)",
        "symbol": "MXP"
    },
    "MXV": {
        "displayName": "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-one": "Mexicanischer Unidad de Inversion
(UDI)",
        "displayName-count-other": "Mexikanische Unidad de Inversion
(UDI)",
        "symbol": "MXV"
    },
    "MYR": {
        "displayName": "Malaysischer Ringgit",
        "displayName-count-one": "Malaysischer Ringgit",
        "displayName-count-other": "Malaysische Ringgit",
        "symbol": "MYR",
        "symbol-alt-narrow": "RM"
    },
    "MZE": {
        "displayName": "Mosambikanischer Escudo",
        "displayName-count-one": "Mozambikanische Escudo",
        "displayName-count-other": "Mozambikanische Escudo",
        "symbol": "MZE"
    },
    "MZM": {
        "displayName": "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one": "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other": "Mosambikanische Meticais (1980-
2006)",
        "symbol": "MZM"
    },
    },

```

```

    "MZN": {
      "displayName": "Mosambikanischer Metical",
      "displayName-count-one": "Mosambikanischer Metical",
      "displayName-count-other": "Mosambikanische Meticais",
      "symbol": "MZN"
    },
    "NAD": {
      "displayName": "Namibia-Dollar",
      "displayName-count-one": "Namibia-Dollar",
      "displayName-count-other": "Namibia-Dollar",
      "symbol": "NAD",
      "symbol-alt-narrow": "$"
    },
    "NGN": {
      "displayName": "Nigerianischer Naira",
      "displayName-count-one": "Nigerianischer Naira",
      "displayName-count-other": "Nigerianische Naira",
      "symbol": "NGN",
      "symbol-alt-narrow": "₦"
    },
    "NIC": {
      "displayName": "Nicaraguanischer Córdoba (1988–1991)",
      "displayName-count-one": "Nicaraguanischer Córdoba (1988–1991)",
      "displayName-count-other": "Nicaraguanische Córdoba (1988–
1991)",
      "symbol": "NIC"
    },
    "NIO": {
      "displayName": "Nicaragua-Córdoba",
      "displayName-count-one": "Nicaragua-Córdoba",
      "displayName-count-other": "Nicaragua-Córdobas",
      "symbol": "NIO",
      "symbol-alt-narrow": "C$"
    },
    "NLG": {
      "displayName": "Niederländischer Gulden",
      "displayName-count-one": "Niederländischer Gulden",
      "displayName-count-other": "Niederländische Gulden",
      "symbol": "NLG"
    },
    "NOK": {
      "displayName": "Norwegische Krone",
      "displayName-count-one": "Norwegische Krone",
      "displayName-count-other": "Norwegische Kronen",
      "symbol": "NOK",
      "symbol-alt-narrow": "kr"
    },
    "NPR": {
      "displayName": "Nepalesische Rupie",
      "displayName-count-one": "Nepalesische Rupie",
      "displayName-count-other": "Nepalesische Rupien",
      "symbol": "NPR",
      "symbol-alt-narrow": "Rs"
    },
    "NZD": {
      "displayName": "Neuseeland-Dollar",
      "displayName-count-one": "Neuseeland-Dollar",

```

```

        "displayName-count-other": "Neuseeland-Dollar",
        "symbol": "NZ$",
        "symbol-alt-narrow": "$"
    },
    "OMR": {
        "displayName": "Omanischer Rial",
        "displayName-count-one": "Omanischer Rial",
        "displayName-count-other": "Omanische Rials",
        "symbol": "OMR"
    },
    "PAB": {
        "displayName": "Panamaischer Balboa",
        "displayName-count-one": "Panamaischer Balboa",
        "displayName-count-other": "Panamaische Balboas",
        "symbol": "PAB"
    },
    "PEI": {
        "displayName": "Peruanischer Inti",
        "displayName-count-one": "Peruanische Inti",
        "displayName-count-other": "Peruanische Inti",
        "symbol": "PEI"
    },
    "PEN": {
        "displayName": "Peruanischer Sol",
        "displayName-count-one": "Peruanischer Sol",
        "displayName-count-other": "Peruanische Sol",
        "symbol": "PEN"
    },
    "PES": {
        "displayName": "Peruanischer Sol (1863-1965)",
        "displayName-count-one": "Peruanischer Sol (1863-1965)",
        "displayName-count-other": "Peruanische Sol (1863-1965)",
        "symbol": "PES"
    },
    "PGK": {
        "displayName": "Papua-Neuguineischer Kina",
        "displayName-count-one": "Papua-Neuguineischer Kina",
        "displayName-count-other": "Papua-Neuguineische Kina",
        "symbol": "PGK"
    },
    "PHP": {
        "displayName": "Philippinischer Peso",
        "displayName-count-one": "Philippinischer Peso",
        "displayName-count-other": "Philippinische Pesos",
        "symbol": "PHP",
        "symbol-alt-narrow": "₱"
    },
    "PKR": {
        "displayName": "Pakistanische Rupie",
        "displayName-count-one": "Pakistanische Rupie",
        "displayName-count-other": "Pakistanische Rupien",
        "symbol": "PKR",
        "symbol-alt-narrow": "Rs"
    },
    "PLN": {
        "displayName": "Polnischer Złoty",
        "displayName-count-one": "Polnischer Złoty",

```



```

        "displayName-count-other": "Polnische Złoty",
        "symbol": "PLN",
        "symbol-alt-narrow": "zł"
    },
    "PLZ": {
        "displayName": "Polnischer Zloty (1950-1995)",
        "displayName-count-one": "Polnischer Zloty (1950-1995)",
        "displayName-count-other": "Polnische Zloty (1950-1995)",
        "symbol": "PLZ"
    },
    "PTE": {
        "displayName": "Portugiesischer Escudo",
        "displayName-count-one": "Portugiesische Escudo",
        "displayName-count-other": "Portugiesische Escudo",
        "symbol": "PTE"
    },
    "PYG": {
        "displayName": "Paraguayischer Guaraní",
        "displayName-count-one": "Paraguayischer Guaraní",
        "displayName-count-other": "Paraguayische Guaranies",
        "symbol": "PYG",
        "symbol-alt-narrow": "₲"
    },
    "QAR": {
        "displayName": "Katar-Riyal",
        "displayName-count-one": "Katar-Riyal",
        "displayName-count-other": "Katar-Riyal",
        "symbol": "QAR"
    },
    "RHD": {
        "displayName": "Rhodesischer Dollar",
        "displayName-count-one": "Rhodesische Dollar",
        "displayName-count-other": "Rhodesische Dollar",
        "symbol": "RHD"
    },
    "ROL": {
        "displayName": "Rumänischer Leu (1952-2006)",
        "displayName-count-one": "Rumänischer Leu (1952-2006)",
        "displayName-count-other": "Rumänische Leu (1952-2006)",
        "symbol": "ROL"
    },
    "RON": {
        "displayName": "Rumänischer Leu",
        "displayName-count-one": "Rumänischer Leu",
        "displayName-count-other": "Rumänische Leu",
        "symbol": "RON",
        "symbol-alt-narrow": "L"
    },
    "RSD": {
        "displayName": "Serbischer Dinar",
        "displayName-count-one": "Serbischer Dinar",
        "displayName-count-other": "Serbische Dinaren",
        "symbol": "RSD"
    },
    "RUB": {
        "displayName": "Russischer Rubel",
        "displayName-count-one": "Russischer Rubel",

```

```

        "displayName-count-other": "Russische Rubel",
        "symbol": "RUB",
        "symbol-alt-narrow": "₽"
    },
    "RUR": {
        "displayName": "Russischer Rubel (1991-1998)",
        "displayName-count-one": "Russischer Rubel (1991-1998)",
        "displayName-count-other": "Russische Rubel (1991-1998)",
        "symbol": "RUR",
        "symbol-alt-narrow": "p."
    },
    "RWF": {
        "displayName": "Ruanda-Franc",
        "displayName-count-one": "Ruanda-Franc",
        "displayName-count-other": "Ruanda-Francs",
        "symbol": "RWF",
        "symbol-alt-narrow": "F.Rw"
    },
    "SAR": {
        "displayName": "Saudi-Rial",
        "displayName-count-one": "Saudi-Rial",
        "displayName-count-other": "Saudi-Rial",
        "symbol": "SAR"
    },
    "SBD": {
        "displayName": "Salomonen-Dollar",
        "displayName-count-one": "Salomonen-Dollar",
        "displayName-count-other": "Salomonen-Dollar",
        "symbol": "SBD",
        "symbol-alt-narrow": "$"
    },
    "SCR": {
        "displayName": "Seychellen-Rupie",
        "displayName-count-one": "Seychellen-Rupie",
        "displayName-count-other": "Seychellen-Rupien",
        "symbol": "SCR"
    },
    "SDD": {
        "displayName": "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one": "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other": "Sudanesische Dinar (1992-2007)",
        "symbol": "SDD"
    },
    "SDG": {
        "displayName": "Sudanesisches Pfund",
        "displayName-count-one": "Sudanesisches Pfund",
        "displayName-count-other": "Sudanesische Pfund",
        "symbol": "SDG"
    },
    "SDP": {
        "displayName": "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one": "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other": "Sudanesische Pfund (1957-1998)",
        "symbol": "SDP"
    },
    "SEK": {
        "displayName": "Schwedische Krone",

```

```

        "displayName-count-one": "Schwedische Krone",
        "displayName-count-other": "Schwedische Kronen",
        "symbol": "SEK",
        "symbol-alt-narrow": "kr"
    },
    "SGD": {
        "displayName": "Singapur-Dollar",
        "displayName-count-one": "Singapur-Dollar",
        "displayName-count-other": "Singapur-Dollar",
        "symbol": "SGD",
        "symbol-alt-narrow": "$"
    },
    "SHP": {
        "displayName": "St. Helena-Pfund",
        "displayName-count-one": "St. Helena-Pfund",
        "displayName-count-other": "St. Helena-Pfund",
        "symbol": "SHP",
        "symbol-alt-narrow": "£"
    },
    "SIT": {
        "displayName": "Slowenischer Tolar",
        "displayName-count-one": "Slowenischer Tolar",
        "displayName-count-other": "Slowenische Tolar",
        "symbol": "SIT"
    },
    "SKK": {
        "displayName": "Slowakische Krone",
        "displayName-count-one": "Slowakische Kronen",
        "displayName-count-other": "Slowakische Kronen",
        "symbol": "SKK"
    },
    "SLL": {
        "displayName": "Sierra-leonischer Leone",
        "displayName-count-one": "Sierra-leonischer Leone",
        "displayName-count-other": "Sierra-leonische Leones",
        "symbol": "SLL"
    },
    "SOS": {
        "displayName": "Somalia-Schilling",
        "displayName-count-one": "Somalia-Schilling",
        "displayName-count-other": "Somalia-Schilling",
        "symbol": "SOS"
    },
    "SRD": {
        "displayName": "Suriname-Dollar",
        "displayName-count-one": "Suriname-Dollar",
        "displayName-count-other": "Suriname-Dollar",
        "symbol": "SRD",
        "symbol-alt-narrow": "$"
    },
    "SRG": {
        "displayName": "Suriname Gulden",
        "displayName-count-one": "Suriname-Gulden",
        "displayName-count-other": "Suriname-Gulden",
        "symbol": "SRG"
    },
    "SSP": {

```

```

        "displayName": "Südsudanesisches Pfund",
        "displayName-count-one": "Südsudanesisches Pfund",
        "displayName-count-other": "Südsudanesische Pfund",
        "symbol": "SSP",
        "symbol-alt-narrow": "£"
    },
    "STD": {
        "displayName": "São-toméischer Dobra",
        "displayName-count-one": "São-toméischer Dobra",
        "displayName-count-other": "São-toméische Dobra",
        "symbol": "STD",
        "symbol-alt-narrow": "Db"
    },
    "SUR": {
        "displayName": "Sowjetischer Rubel",
        "displayName-count-one": "Sowjetische Rubel",
        "displayName-count-other": "Sowjetische Rubel",
        "symbol": "SUR"
    },
    "SVC": {
        "displayName": "El Salvador Colon",
        "displayName-count-one": "El Salvador-Colon",
        "displayName-count-other": "El Salvador-Colon",
        "symbol": "SVC"
    },
    "SYP": {
        "displayName": "Syrisches Pfund",
        "displayName-count-one": "Syrisches Pfund",
        "displayName-count-other": "Syrische Pfund",
        "symbol": "SYP",
        "symbol-alt-narrow": "SYP"
    },
    "SZL": {
        "displayName": "Swasiländischer Lilangeni",
        "displayName-count-one": "Swasiländischer Lilangeni",
        "displayName-count-other": "Swasiländische Emalangeni",
        "symbol": "SZL"
    },
    "THB": {
        "displayName": "Thailändischer Baht",
        "displayName-count-one": "Thailändischer Baht",
        "displayName-count-other": "Thailändische Baht",
        "symbol": "฿",
        "symbol-alt-narrow": "฿"
    },
    "TJR": {
        "displayName": "Tadschikistan Rubel",
        "displayName-count-one": "Tadschikistan-Rubel",
        "displayName-count-other": "Tadschikistan-Rubel",
        "symbol": "TJR"
    },
    "TJS": {
        "displayName": "Tadschikistan-Somoni",
        "displayName-count-one": "Tadschikistan-Somoni",
        "displayName-count-other": "Tadschikistan-Somoni",
        "symbol": "TJS"
    },
    },

```

```

    "TMM": {
      "displayName": "Turkmenistan-Manat (1993-2009)",
      "displayName-count-one": "Turkmenistan-Manat (1993-2009)",
      "displayName-count-other": "Turkmenistan-Manat (1993-2009)",
      "symbol": "TMM"
    },
    "TMT": {
      "displayName": "Turkmenistan-Manat",
      "displayName-count-one": "Turkmenistan-Manat",
      "displayName-count-other": "Turkmenistan-Manat",
      "symbol": "TMT"
    },
    "TND": {
      "displayName": "Tunesischer Dinar",
      "displayName-count-one": "Tunesischer Dinar",
      "displayName-count-other": "Tunesische Dinar",
      "symbol": "TND"
    },
    "TOP": {
      "displayName": "Tongaischer Pa'anga",
      "displayName-count-one": "Tongaischer Pa'anga",
      "displayName-count-other": "Tongaische Pa'anga",
      "symbol": "TOP",
      "symbol-alt-narrow": "T$"
    },
    "TPE": {
      "displayName": "Timor-Escudo",
      "displayName-count-one": "Timor-Escudo",
      "displayName-count-other": "Timor-Escudo",
      "symbol": "TPE"
    },
    "TRL": {
      "displayName": "Türkische Lira (1922-2005)",
      "displayName-count-one": "Türkische Lira (1922-2005)",
      "displayName-count-other": "Türkische Lira (1922-2005)",
      "symbol": "TRL"
    },
    "TRY": {
      "displayName": "Türkische Lira",
      "displayName-count-one": "Türkische Lira",
      "displayName-count-other": "Türkische Lira",
      "symbol": "TRY",
      "symbol-alt-narrow": "₺",
      "symbol-alt-variant": "TL"
    },
    "TTD": {
      "displayName": "Trinidad und Tobago-Dollar",
      "displayName-count-one": "Trinidad und Tobago-Dollar",
      "displayName-count-other": "Trinidad und Tobago-Dollar",
      "symbol": "TTD",
      "symbol-alt-narrow": "$"
    },
    "TWD": {
      "displayName": "Neuer Taiwan-Dollar",
      "displayName-count-one": "Neuer Taiwan-Dollar",
      "displayName-count-other": "Neue Taiwan-Dollar",
      "symbol": "NT$",

```

```

        "symbol-alt-narrow": "NT$"
    },
    "TZS": {
        "displayName": "Tansania-Schilling",
        "displayName-count-one": "Tansania-Schilling",
        "displayName-count-other": "Tansania-Schilling",
        "symbol": "TZS"
    },
    "UAH": {
        "displayName": "Ukrainische Hrywnja",
        "displayName-count-one": "Ukrainische Hrywnja",
        "displayName-count-other": "Ukrainische Hrywen",
        "symbol": "UAH",
        "symbol-alt-narrow": "₴"
    },
    "UAK": {
        "displayName": "Ukrainischer Karbovanetz",
        "displayName-count-one": "Ukrainische Karbovanetz",
        "displayName-count-other": "Ukrainische Karbovanetz",
        "symbol": "UAK"
    },
    "UGS": {
        "displayName": "Uganda-Schilling (1966-1987)",
        "displayName-count-one": "Uganda-Schilling (1966-1987)",
        "displayName-count-other": "Uganda-Schilling (1966-1987)",
        "symbol": "UGS"
    },
    "UGX": {
        "displayName": "Uganda-Schilling",
        "displayName-count-one": "Uganda-Schilling",
        "displayName-count-other": "Uganda-Schilling",
        "symbol": "UGX"
    },
    "USD": {
        "displayName": "US-Dollar",
        "displayName-count-one": "US-Dollar",
        "displayName-count-other": "US-Dollar",
        "symbol": "$",
        "symbol-alt-narrow": "$"
    },
    "USN": {
        "displayName": "US Dollar (Nächster Tag)",
        "displayName-count-one": "US-Dollar (Nächster Tag)",
        "displayName-count-other": "US-Dollar (Nächster Tag)",
        "symbol": "USN"
    },
    "USS": {
        "displayName": "US Dollar (Gleicher Tag)",
        "displayName-count-one": "US-Dollar (Gleicher Tag)",
        "displayName-count-other": "US-Dollar (Gleicher Tag)",
        "symbol": "USS"
    },
    "UYI": {
        "displayName": "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
        "displayName-count-one": "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",

```

```

        "displayName-count-other": "Uruguayische Pesos (Indexierte
Rechnungseinheiten)",
        "symbol": "UYI"
    },
    "UYP": {
        "displayName": "Uruguayischer Peso (1975-1993)",
        "displayName-count-one": "Uruguayischer Peso (1975-1993)",
        "displayName-count-other": "Uruguayische Pesos (1975-1993)",
        "symbol": "UYP"
    },
    "UYU": {
        "displayName": "Uruguayischer Peso",
        "displayName-count-one": "Uruguayischer Peso",
        "displayName-count-other": "Uruguayische Pesos",
        "symbol": "UYU",
        "symbol-alt-narrow": "$"
    },
    "UZS": {
        "displayName": "Usbekistan-Sum",
        "displayName-count-one": "Usbekistan-Sum",
        "displayName-count-other": "Usbekistan-Sum",
        "symbol": "UZS"
    },
    "VEB": {
        "displayName": "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-one": "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-other": "Venezolanische Bolíva
2008)",
        "symbol": "VEB"
    },
    "VEF": {
        "displayName": "Venezolanischer Bolívar",
        "displayName-count-one": "Venezolanischer Bolívar",
        "displayName-count-other": "Venezolanische Bolíva
res",
        "symbol": "VEF",
        "symbol-alt-narrow": "Bs"
    },
    "VND": {
        "displayName": "Vietnamesischer Dong",
        "displayName-count-one": "Vietnamesischer Dong",
        "displayName-count-other": "Vietnamesische Dong",
        "symbol": "₫",
        "symbol-alt-narrow": "₫"
    },
    "VNN": {
        "displayName": "Vietnamesischer Dong (1978-1985)",
        "displayName-count-one": "Vietnamesischer Dong (1978-1985)",
        "displayName-count-other": "Vietnamesische Dong (1978-1985)",
        "symbol": "VNN"
    },
    "VUV": {
        "displayName": "Vanuatu-Vatu",
        "displayName-count-one": "Vanuatu-Vatu",
        "displayName-count-other": "Vanuatu-Vatu",
        "symbol": "VUV"
    },
    "WST": {

```

```

        "displayName": "Samoanischer Tala",
        "displayName-count-one": "Samoanischer Tala",
        "displayName-count-other": "Samoanische Tala",
        "symbol": "WST"
    },
    "XAF": {
        "displayName": "CFA-Franc (BEAC)",
        "displayName-count-one": "CFA-Franc (BEAC)",
        "displayName-count-other": "CFA-Franc (BEAC)",
        "symbol": "FCFA"
    },
    "XAG": {
        "displayName": "Unze Silber",
        "displayName-count-one": "Unze Silber",
        "displayName-count-other": "Unzen Silber",
        "symbol": "XAG"
    },
    "XAU": {
        "displayName": "Unze Gold",
        "displayName-count-one": "Unze Gold",
        "displayName-count-other": "Unzen Gold",
        "symbol": "XAU"
    },
    "XBA": {
        "displayName": "Europäische Rechnungseinheit",
        "displayName-count-one": "Europäische Rechnungseinheiten",
        "displayName-count-other": "Europäische Rechnungseinheiten",
        "symbol": "XBA"
    },
    "XBB": {
        "displayName": "Europäische Währungseinheit (XBB)",
        "displayName-count-one": "Europäische Währungseinheiten (XBB)",
        "displayName-count-other": "Europäische Währungseinheiten
(XBB) ",
        "symbol": "XBB"
    },
    "XBC": {
        "displayName": "Europäische Rechnungseinheit (XBC)",
        "displayName-count-one": "Europäische Rechnungseinheiten (XBC)",
        "displayName-count-other": "Europäische Rechnungseinheiten
(XBC) ",
        "symbol": "XBC"
    },
    "XBD": {
        "displayName": "Europäische Rechnungseinheit (XBD)",
        "displayName-count-one": "Europäische Rechnungseinheiten (XBD)",
        "displayName-count-other": "Europäische Rechnungseinheiten
(XBD) ",
        "symbol": "XBD"
    },
    "XCD": {
        "displayName": "Ostkaribischer Dollar",
        "displayName-count-one": "Ostkaribischer Dollar",
        "displayName-count-other": "Ostkaribische Dollar",
        "symbol": "EC$",
        "symbol-alt-narrow": "$"
    },
    },

```



```

    "XDR": {
      "displayName": "Sonderziehungsrechte",
      "displayName-count-one": "Sonderziehungsrechte",
      "displayName-count-other": "Sonderziehungsrechte",
      "symbol": "XDR"
    },
    "XEU": {
      "displayName": "Europäische Währungseinheit (XEU)",
      "displayName-count-one": "Europäische Währungseinheiten (XEU)",
      "displayName-count-other": "Europäische Währungseinheiten
(XEU) ",
      "symbol": "XEU"
    },
    "XFO": {
      "displayName": "Französischer Gold-Franc",
      "displayName-count-one": "Französische Gold-Franc",
      "displayName-count-other": "Französische Gold-Franc",
      "symbol": "XFO"
    },
    "XFU": {
      "displayName": "Französischer UIC-Franc",
      "displayName-count-one": "Französische UIC-Franc",
      "displayName-count-other": "Französische UIC-Franc",
      "symbol": "XFU"
    },
    "XOF": {
      "displayName": "CFA-Franc (BCEAO)",
      "displayName-count-one": "CFA-Franc (BCEAO)",
      "displayName-count-other": "CFA-Francs (BCEAO)",
      "symbol": "CFA"
    },
    "XPD": {
      "displayName": "Unze Palladium",
      "displayName-count-one": "Unze Palladium",
      "displayName-count-other": "Unzen Palladium",
      "symbol": "XPD"
    },
    "XPF": {
      "displayName": "CFP-Franc",
      "displayName-count-one": "CFP-Franc",
      "displayName-count-other": "CFP-Franc",
      "symbol": "CFPF"
    },
    "XPT": {
      "displayName": "Unze Platin",
      "displayName-count-one": "Unze Platin",
      "displayName-count-other": "Unzen Platin",
      "symbol": "XPT"
    },
    "XRE": {
      "displayName": "RINET Funds",
      "displayName-count-one": "RINET Funds",
      "displayName-count-other": "RINET Funds",
      "symbol": "XRE"
    },
    "XSU": {
      "displayName": "SUCRE",

```

```

        "displayName-count-one": "SUCRE",
        "displayName-count-other": "SUCRE",
        "symbol": "XSU"
    },
    "XTS": {
        "displayName": "Testwährung",
        "displayName-count-one": "Testwährung",
        "displayName-count-other": "Testwährung",
        "symbol": "XTS"
    },
    "XUA": {
        "displayName": "Rechnungseinheit der AfEB",
        "displayName-count-one": "Rechnungseinheit der AfEB",
        "displayName-count-other": "Rechnungseinheiten der AfEB",
        "symbol": "XUA"
    },
    "XXX": {
        "displayName": "Unbekannte Währung",
        "displayName-count-one": "(unbekannte Währung)",
        "displayName-count-other": "(unbekannte Währung)",
        "symbol": "XXX"
    },
    "YDD": {
        "displayName": "Jemen-Dinar",
        "displayName-count-one": "Jemen-Dinar",
        "displayName-count-other": "Jemen-Dinar",
        "symbol": "YDD"
    },
    "YER": {
        "displayName": "Jemen-Rial",
        "displayName-count-one": "Jemen-Rial",
        "displayName-count-other": "Jemen-Rial",
        "symbol": "YER"
    },
    "YUD": {
        "displayName": "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-one": "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-other": "Jugoslawische Dinar (1966-1990)",
        "symbol": "YUD"
    },
    "YUM": {
        "displayName": "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-one": "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-other": "Jugoslawische Neue Dinar (1994-2002)",
        "symbol": "YUM"
    },
    "YUN": {
        "displayName": "Jugoslawischer Dinar (konvertibel)",
        "displayName-count-one": "Jugoslawische Dinar (konvertibel)",
        "displayName-count-other": "Jugoslawische Dinar (konvertibel)",
        "symbol": "YUN"
    },
    "YUR": {
        "displayName": "Jugoslawischer reformierter Dinar (1992-1993)",

```

```

        "displayName-count-one": "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-other": "Jugoslawische reformierte Dinar (1992-1993)",
        "symbol": "YUR"
    },
    "ZAL": {
        "displayName": "Südafrikanischer Rand (Finanz)",
        "displayName-count-one": "Südafrikanischer Rand (Finanz)",
        "displayName-count-other": "Südafrikanischer Rand (Finanz)",
        "symbol": "ZAL"
    },
    "ZAR": {
        "displayName": "Südafrikanischer Rand",
        "displayName-count-one": "Südafrikanischer Rand",
        "displayName-count-other": "Südafrikanische Rand",
        "symbol": "ZAR",
        "symbol-alt-narrow": "R"
    },
    "ZMK": {
        "displayName": "Kwacha (1968-2012)",
        "displayName-count-one": "Kwacha (1968-2012)",
        "displayName-count-other": "Kwacha (1968-2012)",
        "symbol": "ZMK"
    },
    "ZMW": {
        "displayName": "Kwacha",
        "displayName-count-one": "Kwacha",
        "displayName-count-other": "Kwacha",
        "symbol": "ZMW",
        "symbol-alt-narrow": "K"
    },
    "ZRN": {
        "displayName": "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-one": "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-other": "Zaire-Neue Zaïre (1993-1998)",
        "symbol": "ZRN"
    },
    "ZRZ": {
        "displayName": "Zaire-Zaïre (1971-1993)",
        "displayName-count-one": "Zaire-Zaïre (1971-1993)",
        "displayName-count-other": "Zaire-Zaïre (1971-1993)",
        "symbol": "ZRZ"
    },
    "ZWD": {
        "displayName": "Simbabwe-Dollar (1980-2008)",
        "displayName-count-one": "Simbabwe-Dollar (1980-2008)",
        "displayName-count-other": "Simbabwe-Dollar (1980-2008)",
        "symbol": "ZWD"
    },
    "ZWL": {
        "displayName": "Simbabwe-Dollar (2009)",
        "displayName-count-one": "Simbabwe-Dollar (2009)",
        "displayName-count-other": "Simbabwe-Dollar (2009)",
        "symbol": "ZWL"
    },
    "ZWR": {

```

```

        "displayName": "Simbabwe-Dollar (2008)",
        "displayName-count-one": "Simbabwe-Dollar (2008)",
        "displayName-count-other": "Simbabwe-Dollar (2008)",
        "symbol": "ZWR"
      }
    }
  }
}

```

CURRENCIES.JSX

```

{
  "main";
  {
    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13259 $",
          "_cldrVersion";
          "31";
        }
        "language";
        "de";
      }
      "numbers";
      {
        "currencies";
        {
          "ADP";
          {
            "displayName";
            "Andorranische Pesete",
            "displayName-count-one";
            "Andorranische Pesete",
            "displayName-count-other";
            "Andorranische Peseten",
            "symbol";
            "ADP";
          }
          "AED";
          {
            "displayName";
            "VAE-Dirham",
            "displayName-count-one";
            "VAE-Dirham",
            "displayName-count-other";
            "VAE-Dirham",
            "symbol";
            "AED";
          }
        }
      }
    }
  }
}

```

```
"AFA";
{
  "displayName";
  "Afghanische Afghani (1927-2002)",
    "displayName-count-one";
  "Afghanische Afghani (1927-2002)",
    "displayName-count-other";
  "Afghanische Afghani (1927-2002)",
    "symbol";
  "AFA";
}
"AFN";
{
  "displayName";
  "Afghanischer Afghani",
    "displayName-count-one";
  "Afghanischer Afghani",
    "displayName-count-other";
  "Afghanische Afghani",
    "symbol";
  "AFN";
}
"ALK";
{
  "displayName";
  "Albanischer Lek (1946-1965)",
    "displayName-count-one";
  "Albanischer Lek (1946-1965)",
    "displayName-count-other";
  "Albanische Lek (1946-1965)";
}
"ALL";
{
  "displayName";
  "Albanischer Lek",
    "displayName-count-one";
  "Albanischer Lek",
    "displayName-count-other";
  "Albanische Lek",
    "symbol";
  "ALL";
}
"AMD";
{
  "displayName";
  "Armenischer Dram",
    "displayName-count-one";
  "Armenischer Dram",
    "displayName-count-other";
  "Armenische Dram",
    "symbol";
  "AMD";
}
"ANG";
{
  "displayName";
  "Niederländische-Antillen-Gulden",
```

```

        "displayName-count-one";
        "Niederländische-Antillen-Gulden",
        "displayName-count-other";
        "Niederländische-Antillen-Gulden",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "Angolanischer Kwanza",
        "displayName-count-one";
        "Angolanischer Kwanza",
        "displayName-count-other";
        "Angolanische Kwanza",
        "symbol";
        "AOA",
        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other";
        "Angolanische Kwanza (1977-1990)",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-other";
        "Angolanische Neue Kwanza (1990-2000)",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-other";
        "Angolanische Kwanza Reajustado (1995-1999)",
        "symbol";
        "AOR";
    }
    "ARA";
    {
        "displayName";
        "Argentinischer Austral",

```

```

        "displayName-count-one";
        "Argentinischer Austral",
        "displayName-count-other";
        "Argentinische Austral",
        "symbol";
        "ARA";
    }
    "ARL";
    {
        "displayName";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other";
        "Argentinische Pesos Ley (1970-1983)",
        "symbol";
        "ARL";
    }
    "ARM";
    {
        "displayName";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-one";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-other";
        "Argentinische Pesos (1881-1970)",
        "symbol";
        "ARM";
    }
    "ARP";
    {
        "displayName";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-one";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-other";
        "Argentinische Peso (1983-1985)",
        "symbol";
        "ARP";
    }
    "ARS";
    {
        "displayName";
        "Argentinischer Peso",
        "displayName-count-one";
        "Argentinischer Peso",
        "displayName-count-other";
        "Argentinische Pesos",
        "symbol";
        "ARS",
        "symbol-alt-narrow";
        "$";
    }
    "ATS";
    {
        "displayName";
        "Österreichischer Schilling",

```

```

        "displayName-count-one";
        "Österreichischer Schilling",
        "displayName-count-other";
        "Österreichische Schilling",
        "symbol";
        "öS";
    }
    "AUD";
    {
        "displayName";
        "Australischer Dollar",
        "displayName-count-one";
        "Australischer Dollar",
        "displayName-count-other";
        "Australische Dollar",
        "symbol";
        "AU$",
        "symbol-alt-narrow";
        "$";
    }
    "AWG";
    {
        "displayName";
        "Aruba-Florin",
        "displayName-count-one";
        "Aruba-Florin",
        "displayName-count-other";
        "Aruba-Florin",
        "symbol";
        "AWG";
    }
    "AZM";
    {
        "displayName";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-one";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-other";
        "Aserbaidtschan-Manat (1993-2006)",
        "symbol";
        "AZM";
    }
    "AZN";
    {
        "displayName";
        "Aserbaidtschan-Manat",
        "displayName-count-one";
        "Aserbaidtschan-Manat",
        "displayName-count-other";
        "Aserbaidtschan-Manat",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "Bosnien und Herzegowina Dinar (1992-1994)",

```



```

        "displayName-count-one";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-other";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-other";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "symbol";
        "BAM",
        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-one";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-other";
        "Bosnien und Herzegowina Neue Dinar (1994-1997)",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "Barbados-Dollar",
        "displayName-count-one";
        "Barbados-Dollar",
        "displayName-count-other";
        "Barbados-Dollar",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "$";
    }
    "BDT";
    {
        "displayName";
        "Bangladesch-Taka",
        "displayName-count-one";
        "Bangladesch-Taka",
        "displayName-count-other";
        "Bangladesch-Taka",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }

```

```

    }
    "BEC";
    {
        "displayName";
        "Belgischer Franc (konvertibel)",
        "displayName-count-one";
        "Belgischer Franc (konvertibel)",
        "displayName-count-other";
        "Belgische Franc (konvertibel)",
        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "Belgischer Franc",
        "displayName-count-one";
        "Belgischer Franc",
        "displayName-count-other";
        "Belgische Franc",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "Belgischer Finanz-Franc",
        "displayName-count-one";
        "Belgischer Finanz-Franc",
        "displayName-count-other";
        "Belgische Finanz-Franc",
        "symbol";
        "BEL";
    }
    "BGL";
    {
        "displayName";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-one";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-other";
        "Bulgarische Lew (1962-1999)",
        "symbol";
        "BGL";
    }
    "BGM";
    {
        "displayName";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-one";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-other";
        "Bulgarische Lew (1952-1962)",
        "symbol";
        "BGK";
    }
    "BGN";

```

```

    {
      "displayName";
      "Bulgarischer Lew",
      "displayName-count-one";
      "Bulgarischer Lew",
      "displayName-count-other";
      "Bulgarische Lew",
      "symbol";
      "BGN";
    }
    "BGO";
    {
      "displayName";
      "Bulgarischer Lew (1879-1952)",
      "displayName-count-one";
      "Bulgarischer Lew (1879-1952)",
      "displayName-count-other";
      "Bulgarische Lew (1879-1952)",
      "symbol";
      "BGJ";
    }
    "BHD";
    {
      "displayName";
      "Bahrain-Dinar",
      "displayName-count-one";
      "Bahrain-Dinar",
      "displayName-count-other";
      "Bahrain-Dinar",
      "symbol";
      "BHD";
    }
    "BIF";
    {
      "displayName";
      "Burundi-Franc",
      "displayName-count-one";
      "Burundi-Franc",
      "displayName-count-other";
      "Burundi-Francs",
      "symbol";
      "BIF";
    }
    "BMD";
    {
      "displayName";
      "Bermuda-Dollar",
      "displayName-count-one";
      "Bermuda-Dollar",
      "displayName-count-other";
      "Bermuda-Dollar",
      "symbol";
      "BMD",
      "symbol-alt-narrow";
      "$";
    }
    "BND";

```

```

{
  "displayName";
  "Brunei-Dollar",
    "displayName-count-one";
  "Brunei-Dollar",
    "displayName-count-other";
  "Brunei-Dollar",
    "symbol";
  "BND",
    "symbol-alt-narrow";
  "$";
}
"BOB";
{
  "displayName";
  "Bolivanischer Boliviano",
    "displayName-count-one";
  "Bolivanischer Boliviano",
    "displayName-count-other";
  "Bolivianische Bolivianos",
    "symbol";
  "BOB",
    "symbol-alt-narrow";
  "Bs";
}
"BOL";
{
  "displayName";
  "Bolivianischer Boliviano (1863-1963)",
    "displayName-count-one";
  "Bolivianischer Boliviano (1863-1963)",
    "displayName-count-other";
  "Bolivianische Bolivianos (1863-1963)",
    "symbol";
  "BOL";
}
"BOP";
{
  "displayName";
  "Bolivianischer Peso",
    "displayName-count-one";
  "Bolivianischer Peso",
    "displayName-count-other";
  "Bolivianische Peso",
    "symbol";
  "BOP";
}
"BOV";
{
  "displayName";
  "Boliviansiche Mvdol",
    "displayName-count-one";
  "Boliviansiche Mvdol",
    "displayName-count-other";
  "Bolivianische Mvdol",
    "symbol";
  "BOV";
}

```

```

    }
    "BRB";
    {
      "displayName";
      "Brazilianischer Cruzeiro Novo (1967-1986)",
      "displayName-count-one";
      "Brazilianischer Cruzeiro Novo (1967-1986)",
      "displayName-count-other";
      "Brazilianische Cruzeiro Novo (1967-1986)",
      "symbol";
      "BRB";
    }
    "BRC";
    {
      "displayName";
      "Brazilianischer Cruzado (1986-1989)",
      "displayName-count-one";
      "Brazilianischer Cruzado (1986-1989)",
      "displayName-count-other";
      "Brazilianische Cruzado (1986-1989)",
      "symbol";
      "BRC";
    }
    "BRE";
    {
      "displayName";
      "Brazilianischer Cruzeiro (1990-1993)",
      "displayName-count-one";
      "Brazilianischer Cruzeiro (1990-1993)",
      "displayName-count-other";
      "Brazilianische Cruzeiro (1990-1993)",
      "symbol";
      "BRE";
    }
    "BRL";
    {
      "displayName";
      "Brazilianischer Real",
      "displayName-count-one";
      "Brazilianischer Real",
      "displayName-count-other";
      "Brazilianische Real",
      "symbol";
      "R$";
      "symbol-alt-narrow";
      "R$";
    }
    "BRN";
    {
      "displayName";
      "Brazilianischer Cruzado Novo (1989-1990)",
      "displayName-count-one";
      "Brazilianischer Cruzado Novo (1989-1990)",
      "displayName-count-other";
      "Brazilianische Cruzado Novo (1989-1990)",
      "symbol";
      "BRN";
    }

```

```

    }
    "BRR";
    {
      "displayName";
      "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-one";
      "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-other";
      "Brasilianische Cruzeiro (1993-1994)",
      "symbol";
      "BRR";
    }
    "BRZ";
    {
      "displayName";
      "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-one";
      "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-other";
      "Brasilianischer Cruzeiro (1942-1967)",
      "symbol";
      "BRZ";
    }
    "BSD";
    {
      "displayName";
      "Bahamas-Dollar",
      "displayName-count-one";
      "Bahamas-Dollar",
      "displayName-count-other";
      "Bahamas-Dollar",
      "symbol";
      "BSD",
      "symbol-alt-narrow";
      "$";
    }
    "BTN";
    {
      "displayName";
      "Bhutan-Ngultrum",
      "displayName-count-one";
      "Bhutan-Ngultrum",
      "displayName-count-other";
      "Bhutan-Ngultrum",
      "symbol";
      "BTN";
    }
    "BUK";
    {
      "displayName";
      "Birmanischer Kyat",
      "displayName-count-one";
      "Birmanischer Kyat",
      "displayName-count-other";
      "Birmanische Kyat",
      "symbol";
      "BUK";
    }

```

```

    }
    "BWP";
    {
      "displayName";
      "Botswanischer Pula",
      "displayName-count-one";
      "Botswanischer Pula",
      "displayName-count-other";
      "Botswanische Pula",
      "symbol";
      "BWP",
      "symbol-alt-narrow";
      "P";
    }
    "BYB";
    {
      "displayName";
      "Belarus-Rubel (1994-1999)",
      "displayName-count-one";
      "Belarus-Rubel (1994-1999)",
      "displayName-count-other";
      "Belarus-Rubel (1994-1999)",
      "symbol";
      "BYB";
    }
    "BYN";
    {
      "displayName";
      "Weißrussischer Rubel",
      "displayName-count-one";
      "Weißrussischer Rubel",
      "displayName-count-other";
      "Weißrussische Rubel",
      "symbol";
      "BYN",
      "symbol-alt-narrow";
      "p.";
    }
    "BYR";
    {
      "displayName";
      "Weißrussischer Rubel (2000-2016)",
      "displayName-count-one";
      "Weißrussischer Rubel (2000-2016)",
      "displayName-count-other";
      "Weißrussische Rubel (2000-2016)",
      "symbol";
      "BYR";
    }
    "BZD";
    {
      "displayName";
      "Belize-Dollar",
      "displayName-count-one";
      "Belize-Dollar",
      "displayName-count-other";
      "Belize-Dollar",

```

```

        "symbol";
        "BZD",
        "symbol-alt-narrow";
        "$";
    }
    "CAD";
    {
        "displayName";
        "Kanadischer Dollar",
        "displayName-count-one";
        "Kanadischer Dollar",
        "displayName-count-other";
        "Kanadische Dollar",
        "symbol";
        "CA$",
        "symbol-alt-narrow";
        "$";
    }
    "CDF";
    {
        "displayName";
        "Kongo-Franc",
        "displayName-count-one";
        "Kongo-Franc",
        "displayName-count-other";
        "Kongo-Francs",
        "symbol";
        "CDF";
    }
    "CHE";
    {
        "displayName";
        "WIR-Euro",
        "displayName-count-one";
        "WIR-Euro",
        "displayName-count-other";
        "WIR-Euro",
        "symbol";
        "CHE";
    }
    "CHF";
    {
        "displayName";
        "Schweizer Franken",
        "displayName-count-one";
        "Schweizer Franken",
        "displayName-count-other";
        "Schweizer Franken",
        "symbol";
        "CHF";
    }
    "CHW";
    {
        "displayName";
        "WIR Franken",
        "displayName-count-one";
        "WIR Franken",

```



```

        "displayName-count-other";
        "WIR Franken",
        "symbol";
        "CHW";
    }
    "CLE";
    {
        "displayName";
        "Chilenischer Escudo",
        "displayName-count-one";
        "Chilenischer Escudo",
        "displayName-count-other";
        "Chilenische Escudo",
        "symbol";
        "CLE";
    }
    "CLF";
    {
        "displayName";
        "Chilenische Unidades de Fomento",
        "displayName-count-one";
        "Chilenische Unidades de Fomento",
        "displayName-count-other";
        "Chilenische Unidades de Fomento",
        "symbol";
        "CLF";
    }
    "CLP";
    {
        "displayName";
        "Chilenischer Peso",
        "displayName-count-one";
        "Chilenischer Peso",
        "displayName-count-other";
        "Chilenische Pesos",
        "symbol";
        "CLP",
        "symbol-alt-narrow";
        "$";
    }
    "CNX";
    {
        "displayName";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-one";
        "Dollar der Chinesischen Volksbank",
        "displayName-count-other";
        "Dollar der Chinesischen Volksbank",
        "symbol";
        "CNX";
    }
    "CNY";
    {
        "displayName";
        "Renminbi Yuan",
        "displayName-count-one";
        "Chinesischer Yuan",

```

```

        "displayName-count-other";
        "Renminbi Yuan",
        "symbol";
        "CN¥",
        "symbol-alt-narrow";
        "¥";
    }
    "COP";
    {
        "displayName";
        "Kolumbianischer Peso",
        "displayName-count-one";
        "Kolumbianischer Peso",
        "displayName-count-other";
        "Kolumbianische Pesos",
        "symbol";
        "COP",
        "symbol-alt-narrow";
        "$";
    }
    "COU";
    {
        "displayName";
        "Kolumbianische Unidades de valor real",
        "displayName-count-one";
        "Kolumbianische Unidad de valor real",
        "displayName-count-other";
        "Kolumbianische Unidades de valor real",
        "symbol";
        "COU";
    }
    "CRC";
    {
        "displayName";
        "Costa-Rica-Colón",
        "displayName-count-one";
        "Costa-Rica-Colón",
        "displayName-count-other";
        "Costa-Rica-Colón",
        "symbol";
        "CRC",
        "symbol-alt-narrow";
        "₡";
    }
    "CSD";
    {
        "displayName";
        "Serbischer Dinar (2002-2006)",
        "displayName-count-one";
        "Serbischer Dinar (2002-2006)",
        "displayName-count-other";
        "Serbische Dinar (2002-2006)",
        "symbol";
        "CSD";
    }
    "CSK";
    {

```

```

        "displayName";
        "Tschechoslowakische Krone",
        "displayName-count-one";
        "Tschechoslowakische Kronen",
        "displayName-count-other";
        "Tschechoslowakische Kronen",
        "symbol";
        "CSK";
    }
    "CUC";
    {
        "displayName";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-one";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-other";
        "Kubanische Pesos (konvertibel)",
        "symbol";
        "CUC",
        "symbol-alt-narrow";
        "Cub$";
    }
    "CUP";
    {
        "displayName";
        "Kubanischer Peso",
        "displayName-count-one";
        "Kubanischer Peso",
        "displayName-count-other";
        "Kubanische Pesos",
        "symbol";
        "CUP",
        "symbol-alt-narrow";
        "$";
    }
    "CVE";
    {
        "displayName";
        "Cabo-Verde-Escudo",
        "displayName-count-one";
        "Cabo-Verde-Escudo",
        "displayName-count-other";
        "Cabo-Verde-Escudos",
        "symbol";
        "CVE";
    }
    "CYP";
    {
        "displayName";
        "Zypern-Pfund",
        "displayName-count-one";
        "Zypern Pfund",
        "displayName-count-other";
        "Zypern Pfund",
        "symbol";
        "CYP";
    }
}

```

```
"CZK";
{
  "displayName";
  "Tschechische Krone",
    "displayName-count-one";
  "Tschechische Krone",
    "displayName-count-other";
  "Tschechische Kronen",
    "symbol";
  "CZK",
    "symbol-alt-narrow";
  "Kč";
}
"DDM";
{
  "displayName";
  "Mark der DDR",
    "displayName-count-one";
  "Mark der DDR",
    "displayName-count-other";
  "Mark der DDR",
    "symbol";
  "DDM";
}
"DEM";
{
  "displayName";
  "Deutsche Mark",
    "displayName-count-one";
  "Deutsche Mark",
    "displayName-count-other";
  "Deutsche Mark",
    "symbol";
  "DM";
}
"DJF";
{
  "displayName";
  "Dschibuti-Franc",
    "displayName-count-one";
  "Dschibuti-Franc",
    "displayName-count-other";
  "Dschibuti-Franc",
    "symbol";
  "DJF";
}
"DKK";
{
  "displayName";
  "Dänische Krone",
    "displayName-count-one";
  "Dänische Krone",
    "displayName-count-other";
  "Dänische Kronen",
    "symbol";
  "DKK",
    "symbol-alt-narrow";
}
```

```

        "kr";
    }
    "DOP";
    {
        "displayName";
        "Dominikanischer Peso",
        "displayName-count-one";
        "Dominikanischer Peso",
        "displayName-count-other";
        "Dominikanische Pesos",
        "symbol";
        "DOP",
        "symbol-alt-narrow";
        "$";
    }
    "DZD";
    {
        "displayName";
        "Algerischer Dinar",
        "displayName-count-one";
        "Algerischer Dinar",
        "displayName-count-other";
        "Algerische Dinar",
        "symbol";
        "DZD";
    }
    "ECS";
    {
        "displayName";
        "Ecuadorianischer Sucre",
        "displayName-count-one";
        "Ecuadorianischer Sucre",
        "displayName-count-other";
        "Ecuadorianische Sucre",
        "symbol";
        "ECS";
    }
    "ECV";
    {
        "displayName";
        "Verrechnungseinheit für Ecuador",
        "displayName-count-one";
        "Verrechnungseinheiten für Ecuador",
        "displayName-count-other";
        "Verrechnungseinheiten für Ecuador",
        "symbol";
        "ECV";
    }
    "EEK";
    {
        "displayName";
        "Estnische Krone",
        "displayName-count-one";
        "Estnische Krone",
        "displayName-count-other";
        "Estnische Kronen",
        "symbol";
    }

```

```

        "EEK";
    }
    "EGP";
    {
        "displayName";
        "Ägyptisches Pfund",
        "displayName-count-one";
        "Ägyptisches Pfund",
        "displayName-count-other";
        "Ägyptische Pfund",
        "symbol";
        "EGP",
        "symbol-alt-narrow";
        "E£";
    }
    "ERN";
    {
        "displayName";
        "Eritreischer Nakfa",
        "displayName-count-one";
        "Eritreischer Nakfa",
        "displayName-count-other";
        "Eritreische Nakfa",
        "symbol";
        "ERN";
    }
    "ESA";
    {
        "displayName";
        "Spanische Peseta (A-Konten)",
        "displayName-count-one";
        "Spanische Peseta (A-Konten)",
        "displayName-count-other";
        "Spanische Peseten (A-Konten)",
        "symbol";
        "ESA";
    }
    "ESB";
    {
        "displayName";
        "Spanische Peseta (konvertibel)",
        "displayName-count-one";
        "Spanische Peseta (konvertibel)",
        "displayName-count-other";
        "Spanische Peseten (konvertibel)",
        "symbol";
        "ESB";
    }
    "ESP";
    {
        "displayName";
        "Spanische Peseta",
        "displayName-count-one";
        "Spanische Peseta",
        "displayName-count-other";
        "Spanische Peseten",
        "symbol";
    }

```

```

        "ESP",
        "symbol-alt-narrow";
        "₱";
    }
    "ETB";
    {
        "displayName";
        "Äthiopischer Birr",
        "displayName-count-one";
        "Äthiopischer Birr",
        "displayName-count-other";
        "Äthiopische Birr",
        "symbol";
        "ETB";
    }
    "EUR";
    {
        "displayName";
        "Euro",
        "displayName-count-one";
        "Euro",
        "displayName-count-other";
        "Euro",
        "symbol";
        "€",
        "symbol-alt-narrow";
        "€";
    }
    "FIM";
    {
        "displayName";
        "Finnische Mark",
        "displayName-count-one";
        "Finnische Mark",
        "displayName-count-other";
        "Finnische Mark",
        "symbol";
        "FIM";
    }
    "FJD";
    {
        "displayName";
        "Fidschi-Dollar",
        "displayName-count-one";
        "Fidschi-Dollar",
        "displayName-count-other";
        "Fidschi-Dollar",
        "symbol";
        "FJD",
        "symbol-alt-narrow";
        "$";
    }
    "FKP";
    {
        "displayName";
        "Falkland-Pfund",
        "displayName-count-one";

```

```

        "Falkland-Pfund",
        "displayName-count-other";
        "Falkland-Pfund",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "Fl£";
    }
    "FRF";
    {
        "displayName";
        "Französischer Franc",
        "displayName-count-one";
        "Französischer Franc",
        "displayName-count-other";
        "Französische Franc",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "Britisches Pfund",
        "displayName-count-one";
        "Britisches Pfund",
        "displayName-count-other";
        "Britische Pfund",
        "symbol";
        "£",
        "symbol-alt-narrow";
        "£";
    }
    "GEK";
    {
        "displayName";
        "Georgischer Kupon Larit",
        "displayName-count-one";
        "Georgischer Kupon Larit",
        "displayName-count-other";
        "Georgische Kupon Larit",
        "symbol";
        "GEK";
    }
    "GEL";
    {
        "displayName";
        "Georgischer Lari",
        "displayName-count-one";
        "Georgischer Lari",
        "displayName-count-other";
        "Georgische Lari",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }

```



```

    }
    "GHC";
    {
        "displayName";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-one";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-other";
        "Ghanaische Cedi (1979-2007)",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "Ghanaischer Cedi",
        "displayName-count-one";
        "Ghanaischer Cedi",
        "displayName-count-other";
        "Ghanaische Cedi",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "Gibraltar-Pfund",
        "displayName-count-one";
        "Gibraltar-Pfund",
        "displayName-count-other";
        "Gibraltar Pfund",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
    "GMD";
    {
        "displayName";
        "Gambia-Dalasi",
        "displayName-count-one";
        "Gambia-Dalasi",
        "displayName-count-other";
        "Gambia-Dalasi",
        "symbol";
        "GMD";
    }
    "GNF";
    {
        "displayName";
        "Guinea-Franc",
        "displayName-count-one";
        "Guinea-Franc",
        "displayName-count-other";
        "Guinea-Franc",
        "symbol";
        "GNF",

```

```

        "symbol-alt-narrow";
        "F.G.";
    }
    "GNS";
    {
        "displayName";
        "Guineischer Syli",
        "displayName-count-one";
        "Guineischer Syli",
        "displayName-count-other";
        "Guineische Syli",
        "symbol";
        "GNS";
    }
    "GQE";
    {
        "displayName";
        "Äquatorialguinea-Ekwele",
        "displayName-count-one";
        "Äquatorialguinea-Ekwele",
        "displayName-count-other";
        "Äquatorialguinea-Ekwele",
        "symbol";
        "GQE";
    }
    "GRD";
    {
        "displayName";
        "Griechische Drachme",
        "displayName-count-one";
        "Griechische Drachme",
        "displayName-count-other";
        "Griechische Drachmen",
        "symbol";
        "GRD";
    }
    "GTQ";
    {
        "displayName";
        "Guatemaltekinscher Quetzal",
        "displayName-count-one";
        "Guatemaltekinscher Quetzal",
        "displayName-count-other";
        "Guatemaltekinsche Quetzales",
        "symbol";
        "GTQ",
        "symbol-alt-narrow";
        "Q";
    }
    "GWE";
    {
        "displayName";
        "Portugiesisch Guinea Escudo",
        "displayName-count-one";
        "Portugiesisch Guinea Escudo",
        "displayName-count-other";
        "Portugiesisch Guinea Escudo",

```

```

        "symbol";
        "GWE";
    }
    "GWP";
    {
        "displayName";
        "Guinea-Bissau Peso",
        "displayName-count-one";
        "Guinea-Bissau Peso",
        "displayName-count-other";
        "Guinea-Bissau Pesos",
        "symbol";
        "GWP";
    }
    "GYD";
    {
        "displayName";
        "Guyana-Dollar",
        "displayName-count-one";
        "Guyana-Dollar",
        "displayName-count-other";
        "Guyana-Dollar",
        "symbol";
        "GYD",
        "symbol-alt-narrow";
        "$";
    }
    "HKD";
    {
        "displayName";
        "Hongkong-Dollar",
        "displayName-count-one";
        "Hongkong-Dollar",
        "displayName-count-other";
        "Hongkong-Dollar",
        "symbol";
        "HK$",
        "symbol-alt-narrow";
        "$";
    }
    "HNL";
    {
        "displayName";
        "Honduras-Lempira",
        "displayName-count-one";
        "Honduras-Lempira",
        "displayName-count-other";
        "Honduras-Lempira",
        "symbol";
        "HNL",
        "symbol-alt-narrow";
        "L";
    }
    "HRD";
    {
        "displayName";
        "Kroatischer Dinar",

```

```

        "displayName-count-one";
        "Kroatischer Dinar",
        "displayName-count-other";
        "Kroatische Dinar",
        "symbol";
        "HRD";
    }
    "HRK";
    {
        "displayName";
        "Kroatischer Kuna",
        "displayName-count-one";
        "Kroatischer Kuna",
        "displayName-count-other";
        "Kroatische Kuna",
        "symbol";
        "HRK",
        "symbol-alt-narrow";
        "kn";
    }
    "HTG";
    {
        "displayName";
        "Haitianische Gourde",
        "displayName-count-one";
        "Haitianische Gourde",
        "displayName-count-other";
        "Haitianische Gourdes",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "Ungarischer Forint",
        "displayName-count-one";
        "Ungarischer Forint",
        "displayName-count-other";
        "Ungarische Forint",
        "symbol";
        "HUF",
        "symbol-alt-narrow";
        "Ft";
    }
    "IDR";
    {
        "displayName";
        "Indonesische Rupiah",
        "displayName-count-one";
        "Indonesische Rupiah",
        "displayName-count-other";
        "Indonesische Rupiah",
        "symbol";
        "IDR",
        "symbol-alt-narrow";
        "Rp";
    }
}

```

```

    "IEP";
    {
        "displayName";
        "Irishes Pfund",
        "displayName-count-one";
        "Irishes Pfund",
        "displayName-count-other";
        "Irische Pfund",
        "symbol";
        "IEP";
    }
    "ILP";
    {
        "displayName";
        "Israelisches Pfund",
        "displayName-count-one";
        "Israelisches Pfund",
        "displayName-count-other";
        "Israelische Pfund",
        "symbol";
        "ILP";
    }
    "ILR";
    {
        "displayName";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-one";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-other";
        "Israelische Schekel (1980-1985)";
    }
    "ILS";
    {
        "displayName";
        "Israelischer Neuer Schekel",
        "displayName-count-one";
        "Israelischer Neuer Schekel",
        "displayName-count-other";
        "Israelische Neue Schekel",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "Indische Rupie",
        "displayName-count-one";
        "Indische Rupie",
        "displayName-count-other";
        "Indische Rupien",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
}

```

```

"IQD";
{
  "displayName";
  "Irakischer Dinar",
    "displayName-count-one";
  "Irakischer Dinar",
    "displayName-count-other";
  "Irakische Dinar",
    "symbol";
  "IQD";
}
"IRR";
{
  "displayName";
  "Iranischer Rial",
    "displayName-count-one";
  "Iranischer Rial",
    "displayName-count-other";
  "Iranische Rial",
    "symbol";
  "IRR";
}
"ISJ";
{
  "displayName";
  "Isländische Krone (1918-1981)",
    "displayName-count-one";
  "Isländische Krone (1918-1981)",
    "displayName-count-other";
  "Isländische Kronen (1918-1981)";
}
"ISK";
{
  "displayName";
  "Isländische Krone",
    "displayName-count-one";
  "Isländische Krone",
    "displayName-count-other";
  "Isländische Kronen",
    "symbol";
  "ISK",
    "symbol-alt-narrow";
  "kr";
}
"ITL";
{
  "displayName";
  "Italienische Lira",
    "displayName-count-one";
  "Italienische Lira",
    "displayName-count-other";
  "Italienische Lire",
    "symbol";
  "ITL";
}
"JMD";
{

```

```

        "displayName";
        "Jamaika-Dollar",
        "displayName-count-one";
        "Jamaika-Dollar",
        "displayName-count-other";
        "Jamaika-Dollar",
        "symbol";
        "JMD",
        "symbol-alt-narrow";
        "$";
    }
    "JOD";
    {
        "displayName";
        "Jordanischer Dinar",
        "displayName-count-one";
        "Jordanischer Dinar",
        "displayName-count-other";
        "Jordanische Dinar",
        "symbol";
        "JOD";
    }
    "JPY";
    {
        "displayName";
        "Japanischer Yen",
        "displayName-count-one";
        "Japanischer Yen",
        "displayName-count-other";
        "Japanische Yen",
        "symbol";
        "¥",
        "symbol-alt-narrow";
        "¥";
    }
    "KES";
    {
        "displayName";
        "Kenia-Schilling",
        "displayName-count-one";
        "Kenia-Schilling",
        "displayName-count-other";
        "Kenia-Schilling",
        "symbol";
        "KES";
    }
    "KGS";
    {
        "displayName";
        "Kirgisischer Som",
        "displayName-count-one";
        "Kirgisischer Som",
        "displayName-count-other";
        "Kirgisische Som",
        "symbol";
        "KGS";
    }
}

```

```

    "KHR";
    {
      "displayName";
      "Kambodschanischer Riel",
      "displayName-count-one";
      "Kambodschanischer Riel",
      "displayName-count-other";
      "Kambodschanische Riel",
      "symbol";
      "KHR",
      "symbol-alt-narrow";
      "៛";
    }
    "KMF";
    {
      "displayName";
      "Komoren-Franc",
      "displayName-count-one";
      "Komoren-Franc",
      "displayName-count-other";
      "Komoren-Francs",
      "symbol";
      "KMF",
      "symbol-alt-narrow";
      "FC";
    }
    "KPW";
    {
      "displayName";
      "Nordkoreanischer Won",
      "displayName-count-one";
      "Nordkoreanischer Won",
      "displayName-count-other";
      "Nordkoreanische Won",
      "symbol";
      "KPW",
      "symbol-alt-narrow";
      "₩";
    }
    "KRH";
    {
      "displayName";
      "Südkoreanischer Hwan (1953-1962)",
      "displayName-count-one";
      "Südkoreanischer Hwan (1953-1962)",
      "displayName-count-other";
      "Südkoreanischer Hwan (1953-1962)",
      "symbol";
      "KRH";
    }
    "KRO";
    {
      "displayName";
      "Südkoreanischer Won (1945-1953)",
      "displayName-count-one";
      "Südkoreanischer Won (1945-1953)",

```



```

        "displayName-count-other";
        "Südkoreanischer Won (1945-1953)",
        "symbol";
        "KRO";
    }
    "KRW";
    {
        "displayName";
        "Südkoreanischer Won",
        "displayName-count-one";
        "Südkoreanischer Won",
        "displayName-count-other";
        "Südkoreanische Won",
        "symbol";
        "₩",
        "symbol-alt-narrow";
        "₩";
    }
    "KWD";
    {
        "displayName";
        "Kuwait-Dinar",
        "displayName-count-one";
        "Kuwait-Dinar",
        "displayName-count-other";
        "Kuwait-Dinar",
        "symbol";
        "KWD";
    }
    "KYD";
    {
        "displayName";
        "Kaiman-Dollar",
        "displayName-count-one";
        "Kaiman-Dollar",
        "displayName-count-other";
        "Kaiman-Dollar",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
        "$";
    }
    "KZT";
    {
        "displayName";
        "Kasachischer Tenge",
        "displayName-count-one";
        "Kasachischer Tenge",
        "displayName-count-other";
        "Kasachische Tenge",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "T";
    }
    "LAK";
    {

```

```

        "displayName";
        "Laotischer Kip",
        "displayName-count-one";
        "Laotischer Kip",
        "displayName-count-other";
        "Laotische Kip",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "Libanesisches Pfund",
        "displayName-count-one";
        "Libanesisches Pfund",
        "displayName-count-other";
        "Libanesische Pfund",
        "symbol";
        "LBP",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";
        "Sri-Lanka-Rupie",
        "displayName-count-one";
        "Sri-Lanka-Rupie",
        "displayName-count-other";
        "Sri-Lanka-Rupien",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "LRD";
    {
        "displayName";
        "Liberianischer Dollar",
        "displayName-count-one";
        "Liberianischer Dollar",
        "displayName-count-other";
        "Liberianische Dollar",
        "symbol";
        "LRD",
        "symbol-alt-narrow";
        "$";
    }
    "LSL";
    {
        "displayName";
        "Loti",
        "displayName-count-one";
        "Loti",
        "displayName-count-other";
    }

```

```

        "Loti",
        "symbol";
        "LSL";
    }
    "LTL";
    {
        "displayName";
        "Litauischer Litas",
        "displayName-count-one";
        "Litauischer Litas",
        "displayName-count-other";
        "Litauische Litas",
        "symbol";
        "LTL",
        "symbol-alt-narrow";
        "Lt";
    }
    "LTT";
    {
        "displayName";
        "Litauischer Talonas",
        "displayName-count-one";
        "Litauische Talonas",
        "displayName-count-other";
        "Litauische Talonas",
        "symbol";
        "LTT";
    }
    "LUC";
    {
        "displayName";
        "Luxemburgischer Franc (konvertibel)",
        "displayName-count-one";
        "Luxemburgische Franc (konvertibel)",
        "displayName-count-other";
        "Luxemburgische Franc (konvertibel)",
        "symbol";
        "LUC";
    }
    "LUF";
    {
        "displayName";
        "Luxemburgischer Franc",
        "displayName-count-one";
        "Luxemburgische Franc",
        "displayName-count-other";
        "Luxemburgische Franc",
        "symbol";
        "LUF";
    }
    "LUL";
    {
        "displayName";
        "Luxemburgischer Finanz-Franc",
        "displayName-count-one";
        "Luxemburgische Finanz-Franc",
        "displayName-count-other";
    }

```

```

        "Luxemburgische Finanz-Franc",
        "symbol";
        "LUL";
    }
    "LVL";
    {
        "displayName";
        "Lettischer Lats",
        "displayName-count-one";
        "Lettischer Lats",
        "displayName-count-other";
        "Lettische Lats",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "Lettischer Rubel",
        "displayName-count-one";
        "Lettische Rubel",
        "displayName-count-other";
        "Lettische Rubel",
        "symbol";
        "LVR";
    }
    "LYD";
    {
        "displayName";
        "Libyscher Dinar",
        "displayName-count-one";
        "Libyscher Dinar",
        "displayName-count-other";
        "Libysche Dinar",
        "symbol";
        "LYD";
    }
    "MAD";
    {
        "displayName";
        "Marokkanischer Dirham",
        "displayName-count-one";
        "Marokkanischer Dirham",
        "displayName-count-other";
        "Marokkanische Dirham",
        "symbol";
        "MAD";
    }
    "MAF";
    {
        "displayName";
        "Marokkanischer Franc",
        "displayName-count-one";
        "Marokkanische Franc",
        "displayName-count-other";
    }

```

```

        "Marokkanische Franc",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "Monegassischer Franc",
        "displayName-count-one";
        "Monegassischer Franc",
        "displayName-count-other";
        "Monegassische Franc",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "Moldau-Cupon",
        "displayName-count-one";
        "Moldau-Cupon",
        "displayName-count-other";
        "Moldau-Cupon",
        "symbol";
        "MDC";
    }
    "MDL";
    {
        "displayName";
        "Moldau-Leu",
        "displayName-count-one";
        "Moldau-Leu",
        "displayName-count-other";
        "Moldau-Leu",
        "symbol";
        "MDL";
    }
    "MGA";
    {
        "displayName";
        "Madagaskar-Ariary",
        "displayName-count-one";
        "Madagaskar-Ariary",
        "displayName-count-other";
        "Madagaskar-Ariary",
        "symbol";
        "MGA",
        "symbol-alt-narrow";
        "Ar";
    }
    "MGF";
    {
        "displayName";
        "Madagaskar-Franc",
        "displayName-count-one";
        "Madagaskar-Franc",
        "displayName-count-other";
    }

```

```

        "Madagaskar-Franc",
        "symbol";
        "MGF";
    }
    "MKD";
    {
        "displayName";
        "Mazedonischer Denar",
        "displayName-count-one";
        "Mazedonischer Denar",
        "displayName-count-other";
        "Mazedonische Denari",
        "symbol";
        "MKD";
    }
    "MKN";
    {
        "displayName";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-one";
        "Mazedonischer Denar (1992-1993)",
        "displayName-count-other";
        "Mazedonische Denar (1992-1993)",
        "symbol";
        "MKN";
    }
    "MLF";
    {
        "displayName";
        "Malischer Franc",
        "displayName-count-one";
        "Malische Franc",
        "displayName-count-other";
        "Malische Franc",
        "symbol";
        "MLF";
    }
    "MMK";
    {
        "displayName";
        "Myanmarischer Kyat",
        "displayName-count-one";
        "Myanmarischer Kyat",
        "displayName-count-other";
        "Myanmarische Kyat",
        "symbol";
        "MMK",
        "symbol-alt-narrow";
        "K";
    }
    "MNT";
    {
        "displayName";
        "Mongolischer Tögrög",
        "displayName-count-one";
        "Mongolischer Tögrög",
        "displayName-count-other";
    }

```

```

        "Mongolische Tögrög",
        "symbol";
    "MNT",
        "symbol-alt-narrow";
    "₮";
}
"MOP";
{
    "displayName";
    "Macao-Pataca",
        "displayName-count-one";
    "Macao-Pataca",
        "displayName-count-other";
    "Macao-Pataca",
        "symbol";
    "MOP";
}
"MRO";
{
    "displayName";
    "Mauretanischer Ouguiya",
        "displayName-count-one";
    "Mauretanischer Ouguiya",
        "displayName-count-other";
    "Mauretanische Ouguiya",
        "symbol";
    "MRO";
}
"MTL";
{
    "displayName";
    "Maltesische Lira",
        "displayName-count-one";
    "Maltesische Lira",
        "displayName-count-other";
    "Maltesische Lira",
        "symbol";
    "MTL";
}
"MTP";
{
    "displayName";
    "Maltesisches Pfund",
        "displayName-count-one";
    "Maltesische Pfund",
        "displayName-count-other";
    "Maltesische Pfund",
        "symbol";
    "MTP";
}
"MUR";
{
    "displayName";
    "Mauritius-Rupie",
        "displayName-count-one";
    "Mauritius-Rupie",
        "displayName-count-other";

```

```

        "Mauritius-Rupien",
        "symbol";
    "MUR",
        "symbol-alt-narrow";
    "Rs";
}
"MVP";
{
    "displayName";
    "Malediven-Rupie (alt)",
        "displayName-count-one";
    "Malediven-Rupie (alt)",
        "displayName-count-other";
    "Malediven-Rupien (alt)";
}
"MVR";
{
    "displayName";
    "Malediven-Rufiyaa",
        "displayName-count-one";
    "Malediven-Rufiyaa",
        "displayName-count-other";
    "Malediven-Rupien",
        "symbol";
    "MVR";
}
"MWK";
{
    "displayName";
    "Malawi-Kwacha",
        "displayName-count-one";
    "Malawi-Kwacha",
        "displayName-count-other";
    "Malawi-Kwacha",
        "symbol";
    "MWK";
}
"MXN";
{
    "displayName";
    "Mexikanischer Peso",
        "displayName-count-one";
    "Mexikanischer Peso",
        "displayName-count-other";
    "Mexikanische Pesos",
        "symbol";
    "MX$",
        "symbol-alt-narrow";
    "$";
}
"MXP";
{
    "displayName";
    "Mexikanischer Silber-Peso (1861-1992)",
        "displayName-count-one";
    "Mexikanische Silber-Peso (1861-1992)",
        "displayName-count-other";
}

```



```

        "Mexikanische Silber-Pesos (1861-1992)",
        "symbol";
        "MXP";
    }
    "MXV";
    {
        "displayName";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-one";
        "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-other";
        "Mexikanische Unidad de Inversion (UDI)",
        "symbol";
        "MXV";
    }
    "MYR";
    {
        "displayName";
        "Malaysischer Ringgit",
        "displayName-count-one";
        "Malaysischer Ringgit",
        "displayName-count-other";
        "Malaysische Ringgit",
        "symbol";
        "MYR",
        "symbol-alt-narrow";
        "RM";
    }
    "MZE";
    {
        "displayName";
        "Mosambikanischer Escudo",
        "displayName-count-one";
        "Mozambikanische Escudo",
        "displayName-count-other";
        "Mozambikanische Escudo",
        "symbol";
        "MZE";
    }
    "MZM";
    {
        "displayName";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other";
        "Mosambikanische Meticais (1980-2006)",
        "symbol";
        "MZM";
    }
    "MZN";
    {
        "displayName";
        "Mosambikanischer Metical",
        "displayName-count-one";
        "Mosambikanischer Metical",
        "displayName-count-other";
    }

```

```

        "Mosambikanische Meticaïs",
        "symbol";
        "MZN";
    }
    "NAD";
    {
        "displayName";
        "Namibia-Dollar",
        "displayName-count-one";
        "Namibia-Dollar",
        "displayName-count-other";
        "Namibia-Dollar",
        "symbol";
        "NAD",
        "symbol-alt-narrow";
        "$";
    }
    "NGN";
    {
        "displayName";
        "Nigerianischer Naira",
        "displayName-count-one";
        "Nigerianischer Naira",
        "displayName-count-other";
        "Nigerianische Naira",
        "symbol";
        "NGN",
        "symbol-alt-narrow";
        "₦";
    }
    "NIC";
    {
        "displayName";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-one";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-other";
        "Nicaraguanische Córdoba (1988-1991)",
        "symbol";
        "NIC";
    }
    "NIO";
    {
        "displayName";
        "Nicaragua-Córdoba",
        "displayName-count-one";
        "Nicaragua-Córdoba",
        "displayName-count-other";
        "Nicaragua-Córdobas",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";

```

```

        "Niederländischer Gulden",
        "displayName-count-one";
        "Niederländischer Gulden",
        "displayName-count-other";
        "Niederländische Gulden",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "Norwegische Krone",
        "displayName-count-one";
        "Norwegische Krone",
        "displayName-count-other";
        "Norwegische Kronen",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "Nepalesische Rupie",
        "displayName-count-one";
        "Nepalesische Rupie",
        "displayName-count-other";
        "Nepalesische Rupien",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {
        "displayName";
        "Neuseeland-Dollar",
        "displayName-count-one";
        "Neuseeland-Dollar",
        "displayName-count-other";
        "Neuseeland-Dollar",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
        "$";
    }
    "OMR";
    {
        "displayName";
        "Omanischer Rial",
        "displayName-count-one";
        "Omanischer Rial",
        "displayName-count-other";
        "Omanische Rials",
        "symbol";
        "OMR";
    }

```

```

    }
    "PAB";
    {
        "displayName";
        "Panamaischer Balboa",
        "displayName-count-one";
        "Panamaischer Balboa",
        "displayName-count-other";
        "Panamaische Balboas",
        "symbol";
        "PAB";
    }
    "PEI";
    {
        "displayName";
        "Peruanischer Inti",
        "displayName-count-one";
        "Peruanische Inti",
        "displayName-count-other";
        "Peruanische Inti",
        "symbol";
        "PEI";
    }
    "PEN";
    {
        "displayName";
        "Peruanischer Sol",
        "displayName-count-one";
        "Peruanischer Sol",
        "displayName-count-other";
        "Peruanische Sol",
        "symbol";
        "PEN";
    }
    "PES";
    {
        "displayName";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-one";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-other";
        "Peruanische Sol (1863-1965)",
        "symbol";
        "PES";
    }
    "PGK";
    {
        "displayName";
        "Papua-Neuguineischer Kina",
        "displayName-count-one";
        "Papua-Neuguineischer Kina",
        "displayName-count-other";
        "Papua-Neuguineische Kina",
        "symbol";
        "PGK";
    }
    "PHP";

```

```

    {
      "displayName";
      "Philippinischer Peso",
      "displayName-count-one";
      "Philippinischer Peso",
      "displayName-count-other";
      "Philippinische Pesos",
      "symbol";
      "PHP",
      "symbol-alt-narrow";
      "₱";
    }
    "PKR";
    {
      "displayName";
      "Pakistanische Rupie",
      "displayName-count-one";
      "Pakistanische Rupie",
      "displayName-count-other";
      "Pakistanische Rupien",
      "symbol";
      "PKR",
      "symbol-alt-narrow";
      "Rs";
    }
    "PLN";
    {
      "displayName";
      "Polnischer Złoty",
      "displayName-count-one";
      "Polnischer Złoty",
      "displayName-count-other";
      "Polnische Złoty",
      "symbol";
      "PLN",
      "symbol-alt-narrow";
      "zł";
    }
    "PLZ";
    {
      "displayName";
      "Polnischer Zloty (1950-1995)",
      "displayName-count-one";
      "Polnischer Zloty (1950-1995)",
      "displayName-count-other";
      "Polnische Zloty (1950-1995)",
      "symbol";
      "PLZ";
    }
    "PTE";
    {
      "displayName";
      "Portugiesischer Escudo",
      "displayName-count-one";
      "Portugiesische Escudo",
      "displayName-count-other";
      "Portugiesische Escudo",

```

```

        "symbol";
        "PTE";
    }
    "PYG";
    {
        "displayName";
        "Paraguayischer Guaraní",
        "displayName-count-one";
        "Paraguayischer Guaraní",
        "displayName-count-other";
        "Paraguayische Guaraníes",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "Katar-Riyal",
        "displayName-count-one";
        "Katar-Riyal",
        "displayName-count-other";
        "Katar-Riyal",
        "symbol";
        "QAR";
    }
    "RHD";
    {
        "displayName";
        "Rhodesischer Dollar",
        "displayName-count-one";
        "Rhodesische Dollar",
        "displayName-count-other";
        "Rhodesische Dollar",
        "symbol";
        "RHD";
    }
    "ROL";
    {
        "displayName";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-one";
        "Rumänischer Leu (1952-2006)",
        "displayName-count-other";
        "Rumänische Leu (1952-2006)",
        "symbol";
        "ROL";
    }
    "RON";
    {
        "displayName";
        "Rumänischer Leu",
        "displayName-count-one";
        "Rumänischer Leu",
        "displayName-count-other";
        "Rumänische Leu",

```

```

        "symbol";
        "RON",
        "symbol-alt-narrow";
        "L";
    }
    "RSD";
    {
        "displayName";
        "Serbischer Dinar",
        "displayName-count-one";
        "Serbischer Dinar",
        "displayName-count-other";
        "Serbische Dinaren",
        "symbol";
        "RSD";
    }
    "RUB";
    {
        "displayName";
        "Russischer Rubel",
        "displayName-count-one";
        "Russischer Rubel",
        "displayName-count-other";
        "Russische Rubel",
        "symbol";
        "RUB",
        "symbol-alt-narrow";
        "₽";
    }
    "RUR";
    {
        "displayName";
        "Russischer Rubel (1991-1998)",
        "displayName-count-one";
        "Russischer Rubel (1991-1998)",
        "displayName-count-other";
        "Russische Rubel (1991-1998)",
        "symbol";
        "RUR",
        "symbol-alt-narrow";
        "p.";
    }
    "RWF";
    {
        "displayName";
        "Ruanda-Franc",
        "displayName-count-one";
        "Ruanda-Franc",
        "displayName-count-other";
        "Ruanda-Francis",
        "symbol";
        "RWF",
        "symbol-alt-narrow";
        "F.Rw";
    }
    "SAR";
    {

```

```

        "displayName";
        "Saudi-Rial",
        "displayName-count-one";
        "Saudi-Rial",
        "displayName-count-other";
        "Saudi-Rial",
        "symbol";
        "SAR";
    }
    "SBD";
    {
        "displayName";
        "Salomonen-Dollar",
        "displayName-count-one";
        "Salomonen-Dollar",
        "displayName-count-other";
        "Salomonen-Dollar",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "$";
    }
    "SCR";
    {
        "displayName";
        "Seychellen-Rupie",
        "displayName-count-one";
        "Seychellen-Rupie",
        "displayName-count-other";
        "Seychellen-Rupien",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other";
        "Sudanesische Dinar (1992-2007)",
        "symbol";
        "SDD";
    }
    "SDG";
    {
        "displayName";
        "Sudanesisches Pfund",
        "displayName-count-one";
        "Sudanesisches Pfund",
        "displayName-count-other";
        "Sudanesische Pfund",
        "symbol";
        "SDG";
    }
    "SDP";
    {

```



```

        "displayName";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other";
        "Sudanesische Pfund (1957-1998)",
        "symbol";
        "SDP";
    }
    "SEK";
    {
        "displayName";
        "Schwedische Krone",
        "displayName-count-one";
        "Schwedische Krone",
        "displayName-count-other";
        "Schwedische Kronen",
        "symbol";
        "SEK",
        "symbol-alt-narrow";
        "kr";
    }
    "SGD";
    {
        "displayName";
        "Singapur-Dollar",
        "displayName-count-one";
        "Singapur-Dollar",
        "displayName-count-other";
        "Singapur-Dollar",
        "symbol";
        "SGD",
        "symbol-alt-narrow";
        "$";
    }
    "SHP";
    {
        "displayName";
        "St. Helena-Pfund",
        "displayName-count-one";
        "St. Helena-Pfund",
        "displayName-count-other";
        "St. Helena-Pfund",
        "symbol";
        "SHP",
        "symbol-alt-narrow";
        "£";
    }
    "SIT";
    {
        "displayName";
        "Slowenischer Tolar",
        "displayName-count-one";
        "Slowenischer Tolar",
        "displayName-count-other";
        "Slowenische Tolar",
        "symbol";
    }

```

```

        "SIT";
    }
    "SKK";
    {
        "displayName";
        "Slowakische Krone",
        "displayName-count-one";
        "Slowakische Kronen",
        "displayName-count-other";
        "Slowakische Kronen",
        "symbol";
        "SKK";
    }
    "SLL";
    {
        "displayName";
        "Sierra-leonischer Leone",
        "displayName-count-one";
        "Sierra-leonischer Leone",
        "displayName-count-other";
        "Sierra-leonische Leones",
        "symbol";
        "SLL";
    }
    "SOS";
    {
        "displayName";
        "Somalia-Schilling",
        "displayName-count-one";
        "Somalia-Schilling",
        "displayName-count-other";
        "Somalia-Schilling",
        "symbol";
        "SOS";
    }
    "SRD";
    {
        "displayName";
        "Suriname-Dollar",
        "displayName-count-one";
        "Suriname-Dollar",
        "displayName-count-other";
        "Suriname-Dollar",
        "symbol";
        "SRD",
        "symbol-alt-narrow";
        "$";
    }
    "SRG";
    {
        "displayName";
        "Suriname Gulden",
        "displayName-count-one";
        "Suriname-Gulden",
        "displayName-count-other";
        "Suriname-Gulden",
        "symbol";
    }

```

```

        "SRG";
    }
    "SSP";
    {
        "displayName";
        "Südsudanesisches Pfund",
        "displayName-count-one";
        "Südsudanesisches Pfund",
        "displayName-count-other";
        "Südsudanesische Pfund",
        "symbol";
        "SSP",
        "symbol-alt-narrow";
        "£";
    }
    "STD";
    {
        "displayName";
        "São-toméischer Dobra",
        "displayName-count-one";
        "São-toméischer Dobra",
        "displayName-count-other";
        "São-toméische Dobra",
        "symbol";
        "STD",
        "symbol-alt-narrow";
        "Db";
    }
    "SUR";
    {
        "displayName";
        "Sowjetischer Rubel",
        "displayName-count-one";
        "Sowjetische Rubel",
        "displayName-count-other";
        "Sowjetische Rubel",
        "symbol";
        "SUR";
    }
    "SVC";
    {
        "displayName";
        "El Salvador Colon",
        "displayName-count-one";
        "El Salvador-Colon",
        "displayName-count-other";
        "El Salvador-Colon",
        "symbol";
        "SVC";
    }
    "SYP";
    {
        "displayName";
        "Syrisches Pfund",
        "displayName-count-one";
        "Syrisches Pfund",
        "displayName-count-other";
    }

```

```

        "Syrische Pfund",
        "symbol";
        "SYP",
        "symbol-alt-narrow";
        "SYP";
    }
    "SZL";
    {
        "displayName";
        "Swasiländischer Lilangeni",
        "displayName-count-one";
        "Swasiländischer Lilangeni",
        "displayName-count-other";
        "Swasiländische Emalangeni",
        "symbol";
        "SZL";
    }
    "THB";
    {
        "displayName";
        "Thailändischer Baht",
        "displayName-count-one";
        "Thailändischer Baht",
        "displayName-count-other";
        "Thailändische Baht",
        "symbol";
        "฿",
        "symbol-alt-narrow";
        "฿";
    }
    "TJR";
    {
        "displayName";
        "Tadschikistan Rubel",
        "displayName-count-one";
        "Tadschikistan-Rubel",
        "displayName-count-other";
        "Tadschikistan-Rubel",
        "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "Tadschikistan-Somoni",
        "displayName-count-one";
        "Tadschikistan-Somoni",
        "displayName-count-other";
        "Tadschikistan-Somoni",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-one";

```

```

        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-other";
        "Turkmenistan-Manat (1993-2009)",
        "symbol";
        "TMM";
    }
    "TMT";
    {
        "displayName";
        "Turkmenistan-Manat",
        "displayName-count-one";
        "Turkmenistan-Manat",
        "displayName-count-other";
        "Turkmenistan-Manat",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "Tunesischer Dinar",
        "displayName-count-one";
        "Tunesischer Dinar",
        "displayName-count-other";
        "Tunesische Dinar",
        "symbol";
        "TND";
    }
    "TOP";
    {
        "displayName";
        "Tongaischer Pa'anga",
        "displayName-count-one";
        "Tongaischer Pa'anga",
        "displayName-count-other";
        "Tongaische Pa'anga",
        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
    "TPE";
    {
        "displayName";
        "Timor-Escudo",
        "displayName-count-one";
        "Timor-Escudo",
        "displayName-count-other";
        "Timor-Escudo",
        "symbol";
        "TPE";
    }
    "TRL";
    {
        "displayName";
        "Türkische Lira (1922-2005)",
        "displayName-count-one";

```

```

        "Türkische Lira (1922-2005)",
        "displayName-count-other";
        "Türkische Lira (1922-2005)",
        "symbol";
        "TRL";
    }
    "TRY";
    {
        "displayName";
        "Türkische Lira",
        "displayName-count-one";
        "Türkische Lira",
        "displayName-count-other";
        "Türkische Lira",
        "symbol";
        "TRY",
        "symbol-alt-narrow";
        "₺",
        "symbol-alt-variant";
        "TL";
    }
    "TTD";
    {
        "displayName";
        "Trinidad und Tobago-Dollar",
        "displayName-count-one";
        "Trinidad und Tobago-Dollar",
        "displayName-count-other";
        "Trinidad und Tobago-Dollar",
        "symbol";
        "TTD",
        "symbol-alt-narrow";
        "$";
    }
    "TWD";
    {
        "displayName";
        "Neuer Taiwan-Dollar",
        "displayName-count-one";
        "Neuer Taiwan-Dollar",
        "displayName-count-other";
        "Neue Taiwan-Dollar",
        "symbol";
        "NT$",
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";
    {
        "displayName";
        "Tansania-Schilling",
        "displayName-count-one";
        "Tansania-Schilling",
        "displayName-count-other";
        "Tansania-Schilling",
        "symbol";
        "TZS";
    }

```

```

    }
    "UAH";
    {
        "displayName";
        "Ukrainische Hrywnja",
        "displayName-count-one";
        "Ukrainische Hrywnja",
        "displayName-count-other";
        "Ukrainische Hrywen",
        "symbol";
        "UAH",
        "symbol-alt-narrow";
        "₴";
    }
    "UAK";
    {
        "displayName";
        "Ukrainischer Karbovanetz",
        "displayName-count-one";
        "Ukrainische Karbovanetz",
        "displayName-count-other";
        "Ukrainische Karbovanetz",
        "symbol";
        "UAK";
    }
    "UGS";
    {
        "displayName";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-one";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-other";
        "Uganda-Schilling (1966-1987)",
        "symbol";
        "UGS";
    }
    "UGX";
    {
        "displayName";
        "Uganda-Schilling",
        "displayName-count-one";
        "Uganda-Schilling",
        "displayName-count-other";
        "Uganda-Schilling",
        "symbol";
        "UGX";
    }
    "USD";
    {
        "displayName";
        "US-Dollar",
        "displayName-count-one";
        "US-Dollar",
        "displayName-count-other";
        "US-Dollar",
        "symbol";
        "$",
    }

```

```

        "symbol-alt-narrow";
        "$";
    }
    "USN";
    {
        "displayName";
        "US Dollar (Nächster Tag)",
        "displayName-count-one";
        "US-Dollar (Nächster Tag)",
        "displayName-count-other";
        "US-Dollar (Nächster Tag)",
        "symbol";
        "USN";
    }
    "USS";
    {
        "displayName";
        "US Dollar (Gleicher Tag)",
        "displayName-count-one";
        "US-Dollar (Gleicher Tag)",
        "displayName-count-other";
        "US-Dollar (Gleicher Tag)",
        "symbol";
        "USS";
    }
    "UYI";
    {
        "displayName";
        "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
        "displayName-count-one";
        "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
        "displayName-count-other";
        "Uruguayische Pesos (Indexierte
Rechnungseinheiten)",
        "symbol";
        "UYI";
    }
    "UYP";
    {
        "displayName";
        "Uruguayischer Peso (1975-1993)",
        "displayName-count-one";
        "Uruguayischer Peso (1975-1993)",
        "displayName-count-other";
        "Uruguayische Pesos (1975-1993)",
        "symbol";
        "UYP";
    }
    "UYU";
    {
        "displayName";
        "Uruguayischer Peso",
        "displayName-count-one";
        "Uruguayischer Peso",
        "displayName-count-other";
    }

```



```

        "Uruguayische Pesos",
        "symbol";
        "UYU",
        "symbol-alt-narrow";
        "$";
    }
    "UZS";
    {
        "displayName";
        "Usbekistan-Sum",
        "displayName-count-one";
        "Usbekistan-Sum",
        "displayName-count-other";
        "Usbekistan-Sum",
        "symbol";
        "UZS";
    }
    "VEB";
    {
        "displayName";
        "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-one";
        "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-other";
        "Venezolanische Bolívares (1871-2008)",
        "symbol";
        "VEB";
    }
    "VEF";
    {
        "displayName";
        "Venezolanischer Bolívar",
        "displayName-count-one";
        "Venezolanischer Bolívar",
        "displayName-count-other";
        "Venezolanische Bolívares",
        "symbol";
        "VEF",
        "symbol-alt-narrow";
        "Bs";
    }
    "VND";
    {
        "displayName";
        "Vietnamesischer Dong",
        "displayName-count-one";
        "Vietnamesischer Dong",
        "displayName-count-other";
        "Vietnamesische Dong",
        "symbol";
        "₫",
        "symbol-alt-narrow";
        "₫";
    }
    "VNN";
    {
        "displayName";

```

```

        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-one";
        "Vietnamesischer Dong (1978-1985) ",
        "displayName-count-other";
        "Vietnamesische Dong (1978-1985) ",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "Vanuatu-Vatu",
        "displayName-count-one";
        "Vanuatu-Vatu",
        "displayName-count-other";
        "Vanuatu-Vatu",
        "symbol";
        "VUV";
    }
    "WST";
    {
        "displayName";
        "Samoanischer Tala",
        "displayName-count-one";
        "Samoanischer Tala",
        "displayName-count-other";
        "Samoanische Tala",
        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "CFA-Franc (BEAC) ",
        "displayName-count-one";
        "CFA-Franc (BEAC) ",
        "displayName-count-other";
        "CFA-Franc (BEAC) ",
        "symbol";
        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "Unze Silber",
        "displayName-count-one";
        "Unze Silber",
        "displayName-count-other";
        "Unzen Silber",
        "symbol";
        "XAG";
    }
    "XAU";
    {
        "displayName";
        "Unze Gold",
        "displayName-count-one";

```

```

        "Unze Gold",
        "displayName-count-other";
    "Unzen Gold",
        "symbol";
    "XAU";
}
"XBA";
{
    "displayName";
    "Europäische Rechnungseinheit",
        "displayName-count-one";
    "Europäische Rechnungseinheiten",
        "displayName-count-other";
    "Europäische Rechnungseinheiten",
        "symbol";
    "XBA";
}
"XBB";
{
    "displayName";
    "Europäische Währungseinheit (XBB)",
        "displayName-count-one";
    "Europäische Währungseinheiten (XBB)",
        "displayName-count-other";
    "Europäische Währungseinheiten (XBB)",
        "symbol";
    "XBB";
}
"XBC";
{
    "displayName";
    "Europäische Rechnungseinheit (XBC)",
        "displayName-count-one";
    "Europäische Rechnungseinheiten (XBC)",
        "displayName-count-other";
    "Europäische Rechnungseinheiten (XBC)",
        "symbol";
    "XBC";
}
"XBD";
{
    "displayName";
    "Europäische Rechnungseinheit (XBD)",
        "displayName-count-one";
    "Europäische Rechnungseinheiten (XBD)",
        "displayName-count-other";
    "Europäische Rechnungseinheiten (XBD)",
        "symbol";
    "XBD";
}
"XCD";
{
    "displayName";
    "Ostkaribischer Dollar",
        "displayName-count-one";
    "Ostkaribischer Dollar",
        "displayName-count-other";

```

```

        "Ostkaribische Dollar",
        "symbol";
    "EC$",
    "symbol-alt-narrow";
    "$";
}
"XDR";
{
    "displayName";
    "Sonderziehungsrechte",
    "displayName-count-one";
    "Sonderziehungsrechte",
    "displayName-count-other";
    "Sonderziehungsrechte",
    "symbol";
    "XDR";
}
"XEU";
{
    "displayName";
    "Europäische Währungseinheit (XEU)",
    "displayName-count-one";
    "Europäische Währungseinheiten (XEU)",
    "displayName-count-other";
    "Europäische Währungseinheiten (XEU)",
    "symbol";
    "XEU";
}
"XFO";
{
    "displayName";
    "Französischer Gold-Franc",
    "displayName-count-one";
    "Französische Gold-Franc",
    "displayName-count-other";
    "Französische Gold-Franc",
    "symbol";
    "XFO";
}
"XFU";
{
    "displayName";
    "Französischer UIC-Franc",
    "displayName-count-one";
    "Französische UIC-Franc",
    "displayName-count-other";
    "Französische UIC-Franc",
    "symbol";
    "XFU";
}
"XOF";
{
    "displayName";
    "CFA-Franc (BCEAO)",
    "displayName-count-one";
    "CFA-Franc (BCEAO)",
    "displayName-count-other";

```

```

        "CFA-Francs (BCEAO)",
        "symbol";
        "CFA";
    }
    "XPD";
    {
        "displayName";
        "Unze Palladium",
        "displayName-count-one";
        "Unze Palladium",
        "displayName-count-other";
        "Unzen Palladium",
        "symbol";
        "XPD";
    }
    "XPF";
    {
        "displayName";
        "CFP-Franc",
        "displayName-count-one";
        "CFP-Franc",
        "displayName-count-other";
        "CFP-Franc",
        "symbol";
        "CFPF";
    }
    "XPT";
    {
        "displayName";
        "Unze Platin",
        "displayName-count-one";
        "Unze Platin",
        "displayName-count-other";
        "Unzen Platin",
        "symbol";
        "XPT";
    }
    "XRE";
    {
        "displayName";
        "RINET Funds",
        "displayName-count-one";
        "RINET Funds",
        "displayName-count-other";
        "RINET Funds",
        "symbol";
        "XRE";
    }
    "XSU";
    {
        "displayName";
        "SUCRE",
        "displayName-count-one";
        "SUCRE",
        "displayName-count-other";
        "SUCRE",
        "symbol";
    }

```

```

        "XSU";
    }
    "XTS";
    {
        "displayName";
        "Testwährung",
            "displayName-count-one";
        "Testwährung",
            "displayName-count-other";
        "Testwährung",
            "symbol";
        "XTS";
    }
    "XUA";
    {
        "displayName";
        "Rechnungseinheit der AfEB",
            "displayName-count-one";
        "Rechnungseinheit der AfEB",
            "displayName-count-other";
        "Rechnungseinheiten der AfEB",
            "symbol";
        "XUA";
    }
    "XXX";
    {
        "displayName";
        "Unbekannte Währung",
            "displayName-count-one";
        "(unbekannte Währung)",
            "displayName-count-other";
        "(unbekannte Währung)",
            "symbol";
        "XXX";
    }
    "YDD";
    {
        "displayName";
        "Jemen-Dinar",
            "displayName-count-one";
        "Jemen-Dinar",
            "displayName-count-other";
        "Jemen-Dinar",
            "symbol";
        "YDD";
    }
    "YER";
    {
        "displayName";
        "Jemen-Rial",
            "displayName-count-one";
        "Jemen-Rial",
            "displayName-count-other";
        "Jemen-Rial",
            "symbol";
        "YER";
    }
}

```

```

"YUD";
{
  "displayName";
  "Jugoslawischer Dinar (1966-1990)",
  "displayName-count-one";
  "Jugoslawischer Dinar (1966-1990)",
  "displayName-count-other";
  "Jugoslawische Dinar (1966-1990)",
  "symbol";
  "YUD";
}
"YUM";
{
  "displayName";
  "Jugoslawischer Neuer Dinar (1994-2002)",
  "displayName-count-one";
  "Jugoslawischer Neuer Dinar (1994-2002)",
  "displayName-count-other";
  "Jugoslawische Neue Dinar (1994-2002)",
  "symbol";
  "YUM";
}
"YUN";
{
  "displayName";
  "Jugoslawischer Dinar (konvertibel)",
  "displayName-count-one";
  "Jugoslawische Dinar (konvertibel)",
  "displayName-count-other";
  "Jugoslawische Dinar (konvertibel)",
  "symbol";
  "YUN";
}
"YUR";
{
  "displayName";
  "Jugoslawischer reformierter Dinar (1992-1993)",
  "displayName-count-one";
  "Jugoslawischer reformierter Dinar (1992-1993)",
  "displayName-count-other";
  "Jugoslawische reformierte Dinar (1992-1993)",
  "symbol";
  "YUR";
}
"ZAL";
{
  "displayName";
  "Südafrikanischer Rand (Finanz)",
  "displayName-count-one";
  "Südafrikanischer Rand (Finanz)",
  "displayName-count-other";
  "Südafrikanischer Rand (Finanz)",
  "symbol";
  "ZAL";
}
"ZAR";
{

```

```

        "displayName";
        "Südafrikanischer Rand",
        "displayName-count-one";
        "Südafrikanischer Rand",
        "displayName-count-other";
        "Südafrikanische Rand",
        "symbol";
        "ZAR",
        "symbol-alt-narrow";
        "R";
    }
    "ZMK";
    {
        "displayName";
        "Kwacha (1968-2012)",
        "displayName-count-one";
        "Kwacha (1968-2012)",
        "displayName-count-other";
        "Kwacha (1968-2012)",
        "symbol";
        "ZMK";
    }
    "ZMW";
    {
        "displayName";
        "Kwacha",
        "displayName-count-one";
        "Kwacha",
        "displayName-count-other";
        "Kwacha",
        "symbol";
        "ZMW",
        "symbol-alt-narrow";
        "K";
    }
    "ZRN";
    {
        "displayName";
        "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-one";
        "Zaire-Neuer Zaïre (1993-1998)",
        "displayName-count-other";
        "Zaire-Neue Zaïre (1993-1998)",
        "symbol";
        "ZRN";
    }
    "ZRZ";
    {
        "displayName";
        "Zaire-Zaïre (1971-1993)",
        "displayName-count-one";
        "Zaire-Zaïre (1971-1993)",
        "displayName-count-other";
        "Zaire-Zaïre (1971-1993)",
        "symbol";
        "ZRZ";
    }
}

```



```

        "ZWD";
    {
        "displayName";
        "Simbabwe-Dollar (1980-2008)",
            "displayName-count-one";
        "Simbabwe-Dollar (1980-2008)",
            "displayName-count-other";
        "Simbabwe-Dollar (1980-2008)",
            "symbol";
        "ZWD";
    }
    "ZWL";
    {
        "displayName";
        "Simbabwe-Dollar (2009)",
            "displayName-count-one";
        "Simbabwe-Dollar (2009)",
            "displayName-count-other";
        "Simbabwe-Dollar (2009)",
            "symbol";
        "ZWL";
    }
    "ZWR";
    {
        "displayName";
        "Simbabwe-Dollar (2008)",
            "displayName-count-one";
        "Simbabwe-Dollar (2008)",
            "displayName-count-other";
        "Simbabwe-Dollar (2008)",
            "symbol";
        "ZWR";
    }
}

}

}

}

```

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
    'de': {
        'datetimepicker': {
```

```

        placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
        today: 'heute'
    }
}
});
class App extends React.Component {
    dateValue = new Date("12/11/2017 1:00 AM");
    render() {
        return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} locale='de' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
    'de': {
        'datetimepicker': {
            placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
            today: 'heute'
        }
    }
});
class App extends React.Component<{}, {}> {
    private dateValue: Date = new Date("12/11/2017 1:00 AM");
    render() {
        return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} locale='de' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

NUMBERINGSYSTEMS.JSON

```

{
    "supplemental": {
        "version": {
            "_number": "$Revision: 12732 $",
            "_unicodeVersion": "9.0.0",
            "_cldrVersion": "31"
        },
        "numberingSystems": {
            "adlm": {

```

```

    "_digits": "୧୪୪୫୬୭୮୯୦",
    "_type": "numeric"
  },
  "ahom": {
    "_digits": "ᱠᱡᱣᱤᱨᱰᱤᱣᱤ",
    "_type": "numeric"
  },
  "arab": {
    "_digits": "٠١٢٣٤٥٦٧٨٩",
    "_type": "numeric"
  },
  "arabext": {
    "_digits": "٠١٢٣٤٥٦٧٨٩",
    "_type": "numeric"
  },
  "armn": {
    "_rules": "armenian-upper",
    "_type": "algorithmic"
  },
  "armnlow": {
    "_rules": "armenian-lower",
    "_type": "algorithmic"
  },
  "bali": {
    "_digits": "᭐᭄ᭅᭆᭇᭈᭉᭊᭋᭌ᭍᭎᭏",
    "_type": "numeric"
  },
  "beng": {
    "_digits": "০১২৩৪৫৬৭৮৯",
    "_type": "numeric"
  },
  "bhks": {
    "_digits": "ᱠᱡᱣᱤᱨᱰᱤᱣᱤ",
    "_type": "numeric"
  },
  "brah": {
    "_digits": "ᱠᱡᱣᱤᱨᱰᱤᱣᱤ",
    "_type": "numeric"
  },
  "cakm": {
    "_digits": "ᱠᱡᱣᱤᱨᱰᱤᱣᱤ",
    "_type": "numeric"
  },
  "cham": {
    "_digits": "ᱠᱡᱣᱤᱨᱰᱤᱣᱤ",
    "_type": "numeric"
  },
  "cyr1": {
    "_rules": "cyrillic-lower",
    "_type": "algorithmic"
  },
  "deva": {
    "_digits": "०१२३४५६७८९",
    "_type": "numeric"
  },
  "ethi": {

```

```

    "_rules": "ethiopic",
    "_type": "algorithmic"
  },
  "fullwide": {
    "_digits": "0 1 2 3 4 5 6 7 8 9",
    "_type": "numeric"
  },
  "geor": {
    "_rules": "georgian",
    "_type": "algorithmic"
  },
  "grek": {
    "_rules": "greek-upper",
    "_type": "algorithmic"
  },
  "greklow": {
    "_rules": "greek-lower",
    "_type": "algorithmic"
  },
  "gujr": {
    "_digits": "૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯",
    "_type": "numeric"
  },
  "guru": {
    "_digits": "੦ ੧ ੨ ੩ ੪ ੫ ੬ ੭ ੮ ੯",
    "_type": "numeric"
  },
  "hanidays": {
    "_rules": "zh/SpelloutRules/spellout-numbering-days",
    "_type": "algorithmic"
  },
  "hanidec": {
    "_digits": "〇 一 二 三 四 五 六 七 八 九",
    "_type": "numeric"
  },
  "hans": {
    "_rules": "zh/SpelloutRules/spellout-cardinal",
    "_type": "algorithmic"
  },
  "hansfin": {
    "_rules": "zh/SpelloutRules/spellout-cardinal-financial",
    "_type": "algorithmic"
  },
  "hant": {
    "_rules": "zh_Hant/SpelloutRules/spellout-cardinal",
    "_type": "algorithmic"
  },
  "hantfin": {
    "_rules": "zh_Hant/SpelloutRules/spellout-cardinal-financial",
    "_type": "algorithmic"
  },
  "hebr": {
    "_rules": "hebrew",
    "_type": "algorithmic"
  },
  "hmng": {

```

```

    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "java": {
    "_digits": "၀၁၂၃၄၅၆၇၈၉၀၁၂၃၄၅၆၇၈၉",
    "_type": "numeric"
  },
  "jpan": {
    "_rules": "ja/SpelloutRules/spellout-cardinal",
    "_type": "algorithmic"
  },
  "jpanfin": {
    "_rules": "ja/SpelloutRules/spellout-cardinal-financial",
    "_type": "algorithmic"
  },
  "kali": {
    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "khmr": {
    "_digits": "០១២៣៤៥៦៧៨៩",
    "_type": "numeric"
  },
  "knda": {
    "_digits": "೦೧೨೩೪೫೬೭೮೯",
    "_type": "numeric"
  },
  "lana": {
    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "lanatham": {
    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "laoo": {
    "_digits": "໐໑໒໓໔໕໖໗໘໑",
    "_type": "numeric"
  },
  "latn": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "lepc": {
    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "limb": {
    "_digits": "□□□□□□□□",
    "_type": "numeric"
  },
  "mathbold": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  },

```

```

"mathdbl": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"mathmono": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"mathsanb": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"mathsans": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"mlym": {
  "_digits": "ഘറനർദ്രണവുൻ",
  "_type": "numeric"
},
"modi": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"mong": {
  "_digits": "᠐᠑᠒᠐ᠠᠨᠨᠠᠭᠤᠯᠤᠰ",
  "_type": "numeric"
},
"mroo": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"mtei": {
  "_digits": "᱀ᱟᱨᱢᱟᱝᱞᱟᱹ",
  "_type": "numeric"
},
"mymr": {
  "_digits": "၀၁၂၃၄၅၆၇၈",
  "_type": "numeric"
},
"mymrshan": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"mymrtlng": {
  "_digits": "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
  "_type": "numeric"
},
"newa": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"nkoo": {
  "_digits": "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",

```

```

    "_type": "numeric"
  },
  "olck": {
    "_digits": "᠐ᠠᠨᠯᠠᠭᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "orya": {
    "_digits": "୦୧୨୩୪୫୬୭୮୯",
    "_type": "numeric"
  },
  "osma": {
    "_digits": "᠐᠑᠎ᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "roman": {
    "_rules": "roman-upper",
    "_type": "algorithmic"
  },
  "romanlow": {
    "_rules": "roman-lower",
    "_type": "algorithmic"
  },
  "saur": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "shrd": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "sind": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "sinh": {
    "_digits": "⁂ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "sora": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "sund": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "takr": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "talu": {
    "_digits": "᠐ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
    "_type": "numeric"
  },
  "taml": {

```

```

    "_rules": "tamil",
    "_type": "algorithmic"
  },
  "tamldec": {
    "_digits": "0௧௨௩௪௫௬௭௮௯",
    "_type": "numeric"
  },
  "telu": {
    "_digits": "౦౧౨౩౪౫౬౭౮౯",
    "_type": "numeric"
  },
  "thai": {
    "_digits": "๐๑๒๓๔๕๖๗๘๙",
    "_type": "numeric"
  },
  "tibb": {
    "_digits": "༠༡༢༣༤༥༦༧༨༩",
    "_type": "numeric"
  },
  "tirh": {
    "_digits": "୦୧୨୩୪୫୬୭୮୯",
    "_type": "numeric"
  },
  "vaih": {
    "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱹᱺᱻᱼᱽ᱾᱿",
    "_type": "numeric"
  },
  "wara": {
    "_digits": "ᳵᳶ᳷᳸᳹ᳺ᳻᳼᳽᳾᳿",
    "_type": "numeric"
  }
}

```

NUMBERINGSYSTEMS.JSX

```

{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
    "numberingSystems";
    {
      "adlm";
      {
        "_digits";
        "ᲀᲁᲂᲃᲄᲅᲆᲇᲈᲉᲊ᲋᲌᲍᲎᲏ᲐᲑᲒᲓᲔᲕᲖᲗᲘᲙᲚᲛᲜᲝᲞᲟᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩᲪᲫᲬᲭᲮᲯᲰᲱᲲᲳᲴᲵᲶᲷᲸᲹᲺ᲻᲼ᲽᲾᲿ",

```



```

        "_type";
        "numeric";
    }
    "ahom";
    {
        "_digits";
        "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
        "_type";
        "numeric";
    }
    "arab";
    {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
    }
    "arabext";
    {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
    }
    "armn";
    {
        "_rules";
        "armenian-upper",
        "_type";
        "algorithmic";
    }
    "armnlow";
    {
        "_rules";
        "armenian-lower",
        "_type";
        "algorithmic";
    }
    "bali";
    {
        "_digits";
        "᭐ᭀᭁᭂᭃ᭄ᭅᭆᭇᭈᭉ",
        "_type";
        "numeric";
    }
    "beng";
    {
        "_digits";
        "০১২৩৪৫৬৭৮৯",
        "_type";
        "numeric";
    }
    "bhks";
    {
        "_digits";
        "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",

```

```

        "_type";
        "numeric";
    }
    "brah";
    {
        "_digits";
        "ᱠᱡᱢᱯᱤᱨᱫᱽᱜᱟᱲ",
        "_type";
        "numeric";
    }
    "cakm";
    {
        "_digits";
        "ᱵᱚᱠᱫᱽᱨᱫᱽᱯᱩᱨ",
        "_type";
        "numeric";
    }
    "cham";
    {
        "_digits";
        "ᱠᱤᱢᱤᱰᱤᱨᱫᱽᱜᱟᱲ",
        "_type";
        "numeric";
    }
    "cyr1";
    {
        "_rules";
        "cyrillic-lower",
        "_type";
        "algorithmic";
    }
    "deva";
    {
        "_digits";
        "ᱠᱤᱢᱤᱰᱤᱨᱫᱽᱜᱟᱲ",
        "_type";
        "numeric";
    }
    "ethi";
    {
        "_rules";
        "ethiopic",
        "_type";
        "algorithmic";
    }
    "fullwide";
    {
        "_digits";
        "ᱠᱤᱢᱤᱰᱤᱨᱫᱽᱜᱟᱲ",
        "_type";
        "numeric";
    }
    "geor";
    {
        "_rules";
        "georgian",

```

```

        "_type";
        "algorithmic";
    }
    "grek";
    {
        "_rules";
        "greek-upper",
        "_type";
        "algorithmic";
    }
    "greklow";
    {
        "_rules";
        "greek-lower",
        "_type";
        "algorithmic";
    }
    "gujr";
    {
        "_digits";
        "૦૧૨૩૪૫૬૭૮૯",
        "_type";
        "numeric";
    }
    "guru";
    {
        "_digits";
        "੦੧੨੩੪੫੬੭੮੯",
        "_type";
        "numeric";
    }
    "hanidays";
    {
        "_rules";
        "zh/SpelloutRules/spellout-numbering-days",
        "_type";
        "algorithmic";
    }
    "hanidec";
    {
        "_digits";
        "〇一二三四五六七八九",
        "_type";
        "numeric";
    }
    "hans";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hansfin";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal-financial",

```

```

        "_type";
        "algorithmic";
    }
    "hant";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hantfin";
    {
        "_rules";
        "zh_Hant/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "hebr";
    {
        "_rules";
        "hebrew",
        "_type";
        "algorithmic";
    }
    "hmng";
    {
        "_digits";
        "ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
        "_type";
        "numeric";
    }
    "java";
    {
        "_digits";
        "ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",
        "_type";
        "numeric";
    }
    "jpan";
    {
        "_rules";
        "ja/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "jpanfin";
    {
        "_rules";
        "ja/SpelloutRules/spellout-cardinal-financial",
        "_type";
        "algorithmic";
    }
    "kali";
    {
        "_digits";
        "ᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨᠠᠨ",

```

```

        "_type";
        "numeric";
    }
    "khmr";
    {
        "_digits";
        "០១២៣៤៥៦៧៨៩",
        "_type";
        "numeric";
    }
    "knda";
    {
        "_digits";
        "೦೧೨೩೪೫೬೭೮೯",
        "_type";
        "numeric";
    }
    "lana";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "lanatham";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "laoo";
    {
        "_digits";
        "໐໑໒໓໔໕໖໗໘໑",
        "_type";
        "numeric";
    }
    "latn";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "lepc";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "limb";
    {
        "_digits";
        "□□□□□□□□",

```

```

        "_type";
        "numeric";
    }
    "mathbold";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathdbl";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathmono";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsanb";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsans";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mlym";
    {
        "_digits";
        "൦൧൨൩൪൫൬൭൮൯",
        "_type";
        "numeric";
    }
    "modi";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "mong";
    {
        "_digits";
        "᠐᠑᠒᠓᠐ᠠᠨᠳᠤᠭᠤᠯᠤᠰ",

```

```

        "_type";
        "numeric";
    }
    "mroo";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mtei";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mymr";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mymrshan";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈",
        "_type";
        "numeric";
    }
    "mymrtlng";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "newa";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "nkoo";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "olck";
    {

```

```

    "_digits";
    "0022662266",
    "_type";
    "numeric";
}
"orya";
{
    "_digits";
    "0099889988",
    "_type";
    "numeric";
}
"osma";
{
    "_digits";
    "05588843CU",
    "_type";
    "numeric";
}
"roman";
{
    "_rules";
    "roman-upper",
    "_type";
    "algorithmic";
}
"romanlow";
{
    "_rules";
    "roman-lower",
    "_type";
    "algorithmic";
}
"saur";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"shrd";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"sind";
{
    "_digits";
    "□□□□□□□□□□",
    "_type";
    "numeric";
}
"sinh";
{

```



```

        "_digits";
        "൧൨൩൪൫൬൭൮൯",
        "_type";
        "numeric";
    }
    "sora";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "sund";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "takr";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "talu";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "taml";
    {
        "_rules";
        "tamil",
        "_type";
        "algorithmic";
    }
    "tamldec";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "telu";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "thai";
    {

```

```

        "_digits";
        "༠༡༢༣༤༥༦༧༨༩",
        "_type";
        "numeric";
    }
    "tibet";
    {
        "_digits";
        "༠༡༢༣༤༥༦༧༨༩",
        "_type";
        "numeric";
    }
    "tirh";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "vair";
    {
        "_digits";
        "འ| ༢ ༣ ༤ ༥ ༦ ༧ ༨ ༩",
        "_type";
        "numeric";
    }
    "wara";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
}
}
}

```

NUMBERS.JSON

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 13259 $",
          "_cldrVersion": "31"
        },
        "language": "de"
      },
      "numbers": {
        "defaultNumberingSystem": "latn",
        "otherNumberingSystems": {
          "native": "latn"
        },
        "minimumGroupingDigits": "1",
        "symbols-numberSystem-latn": {
```

```

    "decimal": ",",
    "group": ".",
    "list": ";",
    "percentSign": "%",
    "plusSign": "+",
    "minusSign": "-",
    "exponential": "E",
    "superscriptingExponent": "·",
    "perMille": "‰",
    "infinity": "∞",
    "nan": "NaN",
    "timeSeparator": ":"
  },
  "decimalFormats-numberSystem-latn": {
    "standard": "#,##0.###",
    "long": {
      "decimalFormat": {
        "1000-count-one": "0 Tausend",
        "1000-count-other": "0 Tausend",
        "10000-count-one": "00 Tausend",
        "10000-count-other": "00 Tausend",
        "100000-count-one": "000 Tausend",
        "100000-count-other": "000 Tausend",
        "1000000-count-one": "0 Million",
        "1000000-count-other": "0 Millionen",
        "10000000-count-one": "00 Millionen",
        "10000000-count-other": "00 Millionen",
        "100000000-count-one": "000 Millionen",
        "100000000-count-other": "000 Millionen",
        "1000000000-count-one": "0 Milliarde",
        "1000000000-count-other": "0 Milliarden",
        "10000000000-count-one": "00 Milliarden",
        "10000000000-count-other": "00 Milliarden",
        "100000000000-count-one": "000 Milliarden",
        "100000000000-count-other": "000 Milliarden",
        "1000000000000-count-one": "0 Billion",
        "1000000000000-count-other": "0 Billionen",
        "10000000000000-count-one": "00 Billionen",
        "10000000000000-count-other": "00 Billionen",
        "100000000000000-count-one": "000 Billionen",
        "100000000000000-count-other": "000 Billionen"
      }
    },
    "short": {
      "decimalFormat": {
        "1000-count-one": "0",
        "1000-count-other": "0",
        "10000-count-one": "0",
        "10000-count-other": "0",
        "100000-count-one": "0",
        "100000-count-other": "0",
        "1000000-count-one": "0 Mio'."',
        "1000000-count-other": "0 Mio'."',
        "10000000-count-one": "00 Mio'."',
        "10000000-count-other": "00 Mio'."',
        "100000000-count-one": "000 Mio'."',
        "100000000-count-other": "000 Mio'."',

```

```

        "1000000000-count-one": "0 Mrd'.'",
        "1000000000-count-other": "0 Mrd'.'",
        "1000000000-count-one": "00 Mrd'.'",
        "1000000000-count-other": "00 Mrd'.'",
        "10000000000-count-one": "000 Mrd'.'",
        "10000000000-count-other": "000 Mrd'.'",
        "100000000000-count-one": "0 Bio'.'",
        "100000000000-count-other": "0 Bio'.'",
        "1000000000000-count-one": "00 Bio'.'",
        "1000000000000-count-other": "00 Bio'.'",
        "10000000000000-count-one": "000 Bio'.'",
        "10000000000000-count-other": "000 Bio'.'"
    }
  },
  "scientificFormats-numberSystem-latn": {
    "standard": "#E0"
  },
  "percentFormats-numberSystem-latn": {
    "standard": "#,##0 %"
  },
  "currencyFormats-numberSystem-latn": {
    "currencySpacing": {
      "beforeCurrency": {
        "currencyMatch": "[:^S:]",
        "surroundingMatch": "[:digit:]",
        "insertBetween": " "
      },
      "afterCurrency": {
        "currencyMatch": "[:^S:]",
        "surroundingMatch": "[:digit:]",
        "insertBetween": " "
      }
    },
    "standard": "#,##0.00 ¤",
    "accounting": "#,##0.00 ¤",
    "short": {
      "standard": {
        "1000-count-one": "0 Tsd'.' ¤",
        "1000-count-other": "0 Tsd'.' ¤",
        "10000-count-one": "00 Tsd'.' ¤",
        "10000-count-other": "00 Tsd'.' ¤",
        "100000-count-one": "000 Tsd'.' ¤",
        "100000-count-other": "000 Tsd'.' ¤",
        "1000000-count-one": "0 Mio'.' ¤",
        "1000000-count-other": "0 Mio'.' ¤",
        "10000000-count-one": "00 Mio'.' ¤",
        "10000000-count-other": "00 Mio'.' ¤",
        "100000000-count-one": "000 Mio'.' ¤",
        "100000000-count-other": "000 Mio'.' ¤",
        "1000000000-count-one": "0 Mrd'.' ¤",
        "1000000000-count-other": "0 Mrd'.' ¤",
        "10000000000-count-one": "00 Mrd'.' ¤",
        "10000000000-count-other": "00 Mrd'.' ¤",
        "100000000000-count-one": "000 Mrd'.' ¤",
        "100000000000-count-other": "000 Mrd'.' ¤",
        "1000000000000-count-one": "0 Bio'.' ¤",

```

```

        "10000000000000-count-other": "0 Bio'.' ¤",
        "10000000000000-count-one": "00 Bio'.' ¤",
        "10000000000000-count-other": "00 Bio'.' ¤",
        "10000000000000-count-one": "000 Bio'.' ¤",
        "10000000000000-count-other": "000 Bio'.' ¤"
    },
    },
    "unitPattern-count-one": "{0} {1}",
    "unitPattern-count-other": "{0} {1}"
},
"miscPatterns-numberSystem-latn": {
    "atLeast": "{0}+",
    "range": "{0}-{1}"
},
"minimalPairs": {
    "pluralMinimalPairs": "{0} Tag",
    "pluralMinimalPairs": "{0} Tage",
    "other": "{0}. Abzweigung nach rechts nehmen"
}
}
}
}
}
}

```

NUMBERS.JSX

```

{
    "main";
    {
        "de";
        {
            "identity";
            {
                "version";
                {
                    "_number";
                    "$Revision: 13259 $",
                    "_cldrVersion";
                    "31";
                }
                "language";
                "de";
            }
            "numbers";
            {
                "defaultNumberingSystem";
                "latn",
                "otherNumberingSystems";
                {
                    "native";
                    "latn";
                }
                "minimumGroupingDigits";
                "1",
                "symbols-numberSystem-latn";
                {

```

```

        "decimal";
        ",",
        "group";
        ".",
        "list";
        ";",
        "percentSign";
        "%",
        "plusSign";
        "+",
        "minusSign";
        "-",
        "exponential";
        "E",
        "superscriptingExponent";
        ". ",
        "perMille";
        "‰",
        "infinity";
        "∞",
        "nan";
        "NaN",
        "timeSeparator";
        ":";
    }
    "decimalFormats-numberSystem-latn";
    {
        "standard";
        "#,##0.###",
        "long";
        {
            "decimalFormat";
            {
                "1000-count-one";
                "0 Tausend",
                "1000-count-other";
                "0 Tausend",
                "10000-count-one";
                "00 Tausend",
                "10000-count-other";
                "00 Tausend",
                "100000-count-one";
                "000 Tausend",
                "100000-count-other";
                "000 Tausend",
                "1000000-count-one";
                "0 Million",
                "1000000-count-other";
                "0 Millionen",
                "10000000-count-one";
                "00 Millionen",
                "10000000-count-other";
                "00 Millionen",
                "100000000-count-one";
                "000 Millionen",
                "100000000-count-other";
                "000 Millionen",
            }
        }
    }

```

```

        "1000000000-count-one";
    "0 Milliarde",
        "1000000000-count-other";
    "0 Milliarden",
        "10000000000-count-one";
    "00 Milliarden",
        "10000000000-count-other";
    "00 Milliarden",
        "100000000000-count-one";
    "000 Milliarden",
        "100000000000-count-other";
    "000 Milliarden",
        "1000000000000-count-one";
    "0 Billion",
        "1000000000000-count-other";
    "0 Billionen",
        "10000000000000-count-one";
    "00 Billionen",
        "10000000000000-count-other";
    "00 Billionen",
        "100000000000000-count-one";
    "000 Billionen",
        "100000000000000-count-other";
    "000 Billionen";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-one";
        "0",
            "1000-count-other";
        "0",
            "10000-count-one";
        "0",
            "10000-count-other";
        "0",
            "100000-count-one";
        "0",
            "100000-count-other";
        "0",
            "1000000-count-one";
        "0 Mio'.'",
            "1000000-count-other";
        "0 Mio'.'",
            "10000000-count-one";
        "00 Mio'.'",
            "10000000-count-other";
        "00 Mio'.'",
            "100000000-count-one";
        "000 Mio'.'",
            "100000000-count-other";
        "000 Mio'.'",
            "1000000000-count-one";
        "0 Mrd'.'",
            "1000000000-count-other";
    }
}

```

```

        "0 Mrd'.'",
        "10000000000-count-one";
        "00 Mrd'.'",
        "10000000000-count-other";
        "00 Mrd'.'",
        "10000000000-count-one";
        "000 Mrd'.'",
        "10000000000-count-other";
        "000 Mrd'.'",
        "10000000000-count-one";
        "0 Bio'.'",
        "10000000000-count-other";
        "0 Bio'.'",
        "10000000000-count-one";
        "00 Bio'.'",
        "10000000000-count-other";
        "00 Bio'.'",
        "10000000000-count-one";
        "000 Bio'.'",
        "10000000000-count-other";
        "000 Bio'.'";
    }
}
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}
"percentFormats-numberSystem-latn";
{
    "standard";
    "#,##0 %";
}
"currencyFormats-numberSystem-latn";
{
    "currencySpacing";
    {
        "beforeCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
        "afterCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
    }
}
}

```



```

"standard";
"#,##0.00 ¤",
    "accounting";
"#,##0.00 ¤",
    "short";
{
    "standard";
    {
        "1000-count-one";
        "0 Tsd'.' ¤",
            "1000-count-other";
        "0 Tsd'.' ¤",
            "10000-count-one";
        "00 Tsd'.' ¤",
            "10000-count-other";
        "00 Tsd'.' ¤",
            "100000-count-one";
        "000 Tsd'.' ¤",
            "100000-count-other";
        "000 Tsd'.' ¤",
            "1000000-count-one";
        "0 Mio'.' ¤",
            "1000000-count-other";
        "0 Mio'.' ¤",
            "10000000-count-one";
        "00 Mio'.' ¤",
            "10000000-count-other";
        "00 Mio'.' ¤",
            "100000000-count-one";
        "000 Mio'.' ¤",
            "100000000-count-other";
        "000 Mio'.' ¤",
            "1000000000-count-one";
        "0 Mrd'.' ¤",
            "1000000000-count-other";
        "0 Mrd'.' ¤",
            "10000000000-count-one";
        "00 Mrd'.' ¤",
            "10000000000-count-other";
        "00 Mrd'.' ¤",
            "100000000000-count-one";
        "000 Mrd'.' ¤",
            "100000000000-count-other";
        "000 Mrd'.' ¤",
            "1000000000000-count-one";
        "0 Bio'.' ¤",
            "1000000000000-count-other";
        "0 Bio'.' ¤",
            "10000000000000-count-one";
        "00 Bio'.' ¤",
            "10000000000000-count-other";
        "00 Bio'.' ¤",
            "100000000000000-count-one";
        "000 Bio'.' ¤",
            "100000000000000-count-other";
        "000 Bio'.' ¤";
    }
}

```

```

    }
    "unitPattern-count-one";
    "{0} {1}",
        "unitPattern-count-other";
    "{0} {1}";
}
"miscPatterns-numberSystem-latn";
{
    "atLeast";
    "{0}+",
        "range";
    "{0}-{1}";
}
"minimalPairs";
{
    "pluralMinimalPairs";
    "{0} Tag",
        "pluralMinimalPairs";
    "{0} Tage",
        "other";
    "{0}. Abzweigung nach rechts nehmen";
}
}
}
}

```

TIMEZONENAMES.JSON

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 12879 $",
          "_cldrVersion": "30.0.3"
        },
        "language": "de"
      },
      "dates": {
        "timeZoneNames": {
          "hourFormat": "+HH:mm;-HH:mm",
          "gmtFormat": "GMT{0}",
          "gmtZeroFormat": "GMT",
          "regionFormat": "{0} Zeit",
          "regionFormat-type-daylight": "{0} Sommerzeit",
          "regionFormat-type-standard": "{0} Normalzeit",
          "fallbackFormat": "{1} ({0})",
          "zone": {
            "America": {
              "Adak": {
                "exemplarCity": "Adak"
              },
              "Anchorage": {
                "exemplarCity": "Anchorage"
              }
            }
          }
        }
      }
    }
  }
}
```

```
"Anguilla": {
  "exemplarCity": "Anguilla"
},
"Antigua": {
  "exemplarCity": "Antigua"
},
"Araguaina": {
  "exemplarCity": "Araguaina"
},
"Argentina": {
  "Rio_Gallegos": {
    "exemplarCity": "Rio Gallegos"
  },
  "San_Juan": {
    "exemplarCity": "San Juan"
  },
  "Ushuaia": {
    "exemplarCity": "Ushuaia"
  },
  "La_Rioja": {
    "exemplarCity": "La Rioja"
  },
  "San_Luis": {
    "exemplarCity": "San Luis"
  },
  "Salta": {
    "exemplarCity": "Salta"
  },
  "Tucuman": {
    "exemplarCity": "Tucuman"
  }
},
"Aruba": {
  "exemplarCity": "Aruba"
},
"Asuncion": {
  "exemplarCity": "Asunción"
},
"Bahia": {
  "exemplarCity": "Bahia"
},
"Bahia_Banderas": {
  "exemplarCity": "Bahia Banderas"
},
"Barbados": {
  "exemplarCity": "Barbados"
},
"Belem": {
  "exemplarCity": "Belem"
},
"Belize": {
  "exemplarCity": "Belize"
},
"Blanc-Sablon": {
  "exemplarCity": "Blanc-Sablon"
},
"Boa_Vista": {
```

```
        "exemplarCity": "Boa Vista"
      },
      "Bogota": {
        "exemplarCity": "Bogotá"
      },
      "Boise": {
        "exemplarCity": "Boise"
      },
      "Buenos_Aires": {
        "exemplarCity": "Buenos Aires"
      },
      "Cambridge_Bay": {
        "exemplarCity": "Cambridge Bay"
      },
      "Campo_Grande": {
        "exemplarCity": "Campo Grande"
      },
      "Cancun": {
        "exemplarCity": "Cancún"
      },
      "Caracas": {
        "exemplarCity": "Caracas"
      },
      "Catamarca": {
        "exemplarCity": "Catamarca"
      },
      "Cayenne": {
        "exemplarCity": "Cayenne"
      },
      "Cayman": {
        "exemplarCity": "Kaimaninseln"
      },
      "Chicago": {
        "exemplarCity": "Chicago"
      },
      "Chihuahua": {
        "exemplarCity": "Chihuahua"
      },
      "Coral_Harbour": {
        "exemplarCity": "Atikokan"
      },
      "Cordoba": {
        "exemplarCity": "Córdoba"
      },
      "Costa_Rica": {
        "exemplarCity": "Costa Rica"
      },
      "Creston": {
        "exemplarCity": "Creston"
      },
      "Cuiaba": {
        "exemplarCity": "Cuiaba"
      },
      "Curacao": {
        "exemplarCity": "Curaçao"
      },
      "Danmarkshavn": {
```

```
    "exemplarCity": "Danmarkshavn"
  },
  "Dawson": {
    "exemplarCity": "Dawson"
  },
  "Dawson_Creek": {
    "exemplarCity": "Dawson Creek"
  },
  "Denver": {
    "exemplarCity": "Denver"
  },
  "Detroit": {
    "exemplarCity": "Detroit"
  },
  "Dominica": {
    "exemplarCity": "Dominica"
  },
  "Edmonton": {
    "exemplarCity": "Edmonton"
  },
  "Eirunepe": {
    "exemplarCity": "Eirunepe"
  },
  "El_Salvador": {
    "exemplarCity": "El Salvador"
  },
  "Fort_Nelson": {
    "exemplarCity": "Fort Nelson"
  },
  "Fortaleza": {
    "exemplarCity": "Fortaleza"
  },
  "Glace_Bay": {
    "exemplarCity": "Glace Bay"
  },
  "Godthab": {
    "exemplarCity": "Nuuk"
  },
  "Goose_Bay": {
    "exemplarCity": "Goose Bay"
  },
  "Grand_Turk": {
    "exemplarCity": "Grand Turk"
  },
  "Grenada": {
    "exemplarCity": "Grenada"
  },
  "Guadeloupe": {
    "exemplarCity": "Guadeloupe"
  },
  "Guatemala": {
    "exemplarCity": "Guatemala"
  },
  "Guayaquil": {
    "exemplarCity": "Guayaquil"
  },
  "Guyana": {
```

```
        "exemplarCity": "Guyana"
      },
      "Halifax": {
        "exemplarCity": "Halifax"
      },
      "Havana": {
        "exemplarCity": "Havanna"
      },
      "Hermosillo": {
        "exemplarCity": "Hermosillo"
      },
      "Indiana": {
        "Vincennes": {
          "exemplarCity": "Vincennes, Indiana"
        },
        "Petersburg": {
          "exemplarCity": "Petersburg, Indiana"
        },
        "Tell_City": {
          "exemplarCity": "Tell City, Indiana"
        },
        "Knox": {
          "exemplarCity": "Knox, Indiana"
        },
        "Winamac": {
          "exemplarCity": "Winamac, Indiana"
        },
        "Marengo": {
          "exemplarCity": "Marengo, Indiana"
        },
        "Vevay": {
          "exemplarCity": "Vevay, Indiana"
        }
      },
      "Indianapolis": {
        "exemplarCity": "Indianapolis"
      },
      "Inuvik": {
        "exemplarCity": "Inuvik"
      },
      "Iqaluit": {
        "exemplarCity": "Iqaluit"
      },
      "Jamaica": {
        "exemplarCity": "Jamaika"
      },
      "Jujuy": {
        "exemplarCity": "Jujuy"
      },
      "Juneau": {
        "exemplarCity": "Juneau"
      },
      "Kentucky": {
        "Monticello": {
          "exemplarCity": "Monticello, Kentucky"
        }
      },
    },
  ],
```

```
"Kralendijk": {
  "exemplarCity": "Kralendijk"
},
"La_Paz": {
  "exemplarCity": "La Paz"
},
"Lima": {
  "exemplarCity": "Lima"
},
"Los_Angeles": {
  "exemplarCity": "Los Angeles"
},
"Louisville": {
  "exemplarCity": "Louisville"
},
"Lower_Princes": {
  "exemplarCity": "Lower Prince's Quarter"
},
"Maceio": {
  "exemplarCity": "Maceio"
},
"Managua": {
  "exemplarCity": "Managua"
},
"Manaus": {
  "exemplarCity": "Manaus"
},
"Marigot": {
  "exemplarCity": "Marigot"
},
"Martinique": {
  "exemplarCity": "Martinique"
},
"Matamoros": {
  "exemplarCity": "Matamoros"
},
"Mazatlan": {
  "exemplarCity": "Mazatlan"
},
"Mendoza": {
  "exemplarCity": "Mendoza"
},
"Menominee": {
  "exemplarCity": "Menominee"
},
"Merida": {
  "exemplarCity": "Merida"
},
"Metlakatla": {
  "exemplarCity": "Metlakatla"
},
"Mexico_City": {
  "exemplarCity": "Mexiko-Stadt"
},
"Miquelon": {
  "exemplarCity": "Miquelon"
},
```

```
"Moncton": {
  "exemplarCity": "Moncton"
},
"Monterrey": {
  "exemplarCity": "Monterrey"
},
"Montevideo": {
  "exemplarCity": "Montevideo"
},
"Montserrat": {
  "exemplarCity": "Montserrat"
},
"Nassau": {
  "exemplarCity": "Nassau"
},
"New_York": {
  "exemplarCity": "New York"
},
"Nipigon": {
  "exemplarCity": "Nipigon"
},
"Nome": {
  "exemplarCity": "Nome"
},
"Noronha": {
  "exemplarCity": "Noronha"
},
"North_Dakota": {
  "Beulah": {
    "exemplarCity": "Beulah, North Dakota"
  },
  "New_Salem": {
    "exemplarCity": "New Salem, North Dakota"
  },
  "Center": {
    "exemplarCity": "Center, North Dakota"
  }
},
"Ojinaga": {
  "exemplarCity": "Ojinaga"
},
"Panama": {
  "exemplarCity": "Panama"
},
"Pangnirtung": {
  "exemplarCity": "Pangnirtung"
},
"Paramaribo": {
  "exemplarCity": "Paramaribo"
},
"Phoenix": {
  "exemplarCity": "Phoenix"
},
"Port-au-Prince": {
  "exemplarCity": "Port-au-Prince"
},
"Port_of_Spain": {
```



```
        "exemplarCity": "Port of Spain"
    },
    "Porto_Velho": {
        "exemplarCity": "Porto Velho"
    },
    "Puerto_Rico": {
        "exemplarCity": "Puerto Rico"
    },
    "Rainy_River": {
        "exemplarCity": "Rainy River"
    },
    "Rankin_Inlet": {
        "exemplarCity": "Rankin Inlet"
    },
    "Recife": {
        "exemplarCity": "Recife"
    },
    "Regina": {
        "exemplarCity": "Regina"
    },
    "Resolute": {
        "exemplarCity": "Resolute"
    },
    "Rio_Branco": {
        "exemplarCity": "Rio Branco"
    },
    "Santa_Isabel": {
        "exemplarCity": "Santa Isabel"
    },
    "Santarem": {
        "exemplarCity": "Santarem"
    },
    "Santiago": {
        "exemplarCity": "Santiago"
    },
    "Santo_Domingo": {
        "exemplarCity": "Santo Domingo"
    },
    "Sao_Paulo": {
        "exemplarCity": "São Paulo"
    },
    "Scoresbysund": {
        "exemplarCity": "Ittoqqortoormiit"
    },
    "Sitka": {
        "exemplarCity": "Sitka"
    },
    "St_Barthelemy": {
        "exemplarCity": "Saint-Barthélemy"
    },
    "St_Johns": {
        "exemplarCity": "St. John's"
    },
    "St_Kitts": {
        "exemplarCity": "St. Kitts"
    },
    "St_Lucia": {
```

```
        "exemplarCity": "St. Lucia"
      },
      "St_Thomas": {
        "exemplarCity": "St. Thomas"
      },
      "St_Vincent": {
        "exemplarCity": "St. Vincent"
      },
      "Swift_Current": {
        "exemplarCity": "Swift Current"
      },
      "Tegucigalpa": {
        "exemplarCity": "Tegucigalpa"
      },
      "Thule": {
        "exemplarCity": "Thule"
      },
      "Thunder_Bay": {
        "exemplarCity": "Thunder Bay"
      },
      "Tijuana": {
        "exemplarCity": "Tijuana"
      },
      "Toronto": {
        "exemplarCity": "Toronto"
      },
      "Tortola": {
        "exemplarCity": "Tortola"
      },
      "Vancouver": {
        "exemplarCity": "Vancouver"
      },
      "Whitehorse": {
        "exemplarCity": "Whitehorse"
      },
      "Winnipeg": {
        "exemplarCity": "Winnipeg"
      },
      "Yakutat": {
        "exemplarCity": "Yakutat"
      },
      "Yellowknife": {
        "exemplarCity": "Yellowknife"
      }
    },
    "Atlantic": {
      "Azores": {
        "exemplarCity": "Azoren"
      },
      "Bermuda": {
        "exemplarCity": "Bermudas"
      },
      "Canary": {
        "exemplarCity": "Kanaren"
      },
      "Cape_Verde": {
        "exemplarCity": "Cabo Verde"
      }
    }
  }
}
```

```
    },
    "Faeroe": {
      "exemplarCity": "Färöer"
    },
    "Madeira": {
      "exemplarCity": "Madeira"
    },
    "Reykjavik": {
      "exemplarCity": "Reykjavík"
    },
    "South_Georgia": {
      "exemplarCity": "Südgeorgien"
    },
    "St_Helena": {
      "exemplarCity": "St. Helena"
    },
    "Stanley": {
      "exemplarCity": "Stanley"
    }
  },
  "Europe": {
    "Amsterdam": {
      "exemplarCity": "Amsterdam"
    },
    "Andorra": {
      "exemplarCity": "Andorra"
    },
    "Astrakhan": {
      "exemplarCity": "Astrachan"
    },
    "Athens": {
      "exemplarCity": "Athen"
    },
    "Belgrade": {
      "exemplarCity": "Belgrad"
    },
    "Berlin": {
      "exemplarCity": "Berlin"
    },
    "Bratislava": {
      "exemplarCity": "Bratislava"
    },
    "Brussels": {
      "exemplarCity": "Brüssel"
    },
    "Bucharest": {
      "exemplarCity": "Bukarest"
    },
    "Budapest": {
      "exemplarCity": "Budapest"
    },
    "Busingen": {
      "exemplarCity": "Büsingen"
    },
    "Chisinau": {
      "exemplarCity": "Kischinau"
    }
  },
```

```
"Copenhagen": {
  "exemplarCity": "Kopenhagen"
},
"Dublin": {
  "long": {
    "daylight": "Irische Sommerzeit"
  },
  "exemplarCity": "Dublin"
},
"Gibraltar": {
  "exemplarCity": "Gibraltar"
},
"Guernsey": {
  "exemplarCity": "Guernsey"
},
"Helsinki": {
  "exemplarCity": "Helsinki"
},
"Isle_of_Man": {
  "exemplarCity": "Isle of Man"
},
"Istanbul": {
  "exemplarCity": "Istanbul"
},
"Jersey": {
  "exemplarCity": "Jersey"
},
"Kaliningrad": {
  "exemplarCity": "Kaliningrad"
},
"Kiev": {
  "exemplarCity": "Kiew"
},
"Kirov": {
  "exemplarCity": "Kirow"
},
"Lisbon": {
  "exemplarCity": "Lissabon"
},
"Ljubljana": {
  "exemplarCity": "Ljubljana"
},
"London": {
  "long": {
    "daylight": "Britische Sommerzeit"
  },
  "exemplarCity": "London"
},
"Luxembourg": {
  "exemplarCity": "Luxemburg"
},
"Madrid": {
  "exemplarCity": "Madrid"
},
"Malta": {
  "exemplarCity": "Malta"
},
```

```
"Mariehamn": {
  "exemplarCity": "Mariehamn"
},
"Minsk": {
  "exemplarCity": "Minsk"
},
"Monaco": {
  "exemplarCity": "Monaco"
},
"Moscow": {
  "exemplarCity": "Moskau"
},
"Oslo": {
  "exemplarCity": "Oslo"
},
"Paris": {
  "exemplarCity": "Paris"
},
"Podgorica": {
  "exemplarCity": "Podgorica"
},
"Prague": {
  "exemplarCity": "Prag"
},
"Riga": {
  "exemplarCity": "Riga"
},
"Rome": {
  "exemplarCity": "Rom"
},
"Samara": {
  "exemplarCity": "Samara"
},
"San_Marino": {
  "exemplarCity": "San Marino"
},
"Sarajevo": {
  "exemplarCity": "Sarajevo"
},
"Simferopol": {
  "exemplarCity": "Simferopol"
},
"Skopje": {
  "exemplarCity": "Skopje"
},
"Sofia": {
  "exemplarCity": "Sofia"
},
"Stockholm": {
  "exemplarCity": "Stockholm"
},
"Tallinn": {
  "exemplarCity": "Tallinn"
},
"Tirane": {
  "exemplarCity": "Tirana"
},
},
```

```
"Ulyanovsk": {
  "exemplarCity": "Uljanowsk"
},
"Uzhgorod": {
  "exemplarCity": "Uschgorod"
},
"Vaduz": {
  "exemplarCity": "Vaduz"
},
"Vatican": {
  "exemplarCity": "Vatikan"
},
"Vienna": {
  "exemplarCity": "Wien"
},
"Vilnius": {
  "exemplarCity": "Vilnius"
},
"Volgograd": {
  "exemplarCity": "Wolgograd"
},
"Warsaw": {
  "exemplarCity": "Warschau"
},
"Zagreb": {
  "exemplarCity": "Zagreb"
},
"Zaporozhye": {
  "exemplarCity": "Saporischja"
},
"Zurich": {
  "exemplarCity": "Zürich"
}
},
"Africa": {
  "Abidjan": {
    "exemplarCity": "Abidjan"
  },
  "Accra": {
    "exemplarCity": "Accra"
  },
  "Addis_Ababa": {
    "exemplarCity": "Addis Abeba"
  },
  "Algiers": {
    "exemplarCity": "Algier"
  },
  "Asmera": {
    "exemplarCity": "Asmara"
  },
  "Bamako": {
    "exemplarCity": "Bamako"
  },
  "Bangui": {
    "exemplarCity": "Bangui"
  },
  "Banjul": {
```

```
        "exemplarCity": "Banjul"
    },
    "Bissau": {
        "exemplarCity": "Bissau"
    },
    "Blantyre": {
        "exemplarCity": "Blantyre"
    },
    "Brazzaville": {
        "exemplarCity": "Brazzaville"
    },
    "Bujumbura": {
        "exemplarCity": "Bujumbura"
    },
    "Cairo": {
        "exemplarCity": "Kairo"
    },
    "Casablanca": {
        "exemplarCity": "Casablanca"
    },
    "Ceuta": {
        "exemplarCity": "Ceuta"
    },
    "Conakry": {
        "exemplarCity": "Conakry"
    },
    "Dakar": {
        "exemplarCity": "Dakar"
    },
    "Dar_es_Salaam": {
        "exemplarCity": "Daressalam"
    },
    "Djibouti": {
        "exemplarCity": "Dschibuti"
    },
    "Douala": {
        "exemplarCity": "Douala"
    },
    "El_Aaiun": {
        "exemplarCity": "El Aaiún"
    },
    "Freetown": {
        "exemplarCity": "Freetown"
    },
    "Gaborone": {
        "exemplarCity": "Gaborone"
    },
    "Harare": {
        "exemplarCity": "Harare"
    },
    "Johannesburg": {
        "exemplarCity": "Johannesburg"
    },
    "Juba": {
        "exemplarCity": "Juba"
    },
    "Kampala": {
```

```
    "exemplarCity": "Kampala"
  },
  "Khartoum": {
    "exemplarCity": "Khartum"
  },
  "Kigali": {
    "exemplarCity": "Kigali"
  },
  "Kinshasa": {
    "exemplarCity": "Kinshasa"
  },
  "Lagos": {
    "exemplarCity": "Lagos"
  },
  "Libreville": {
    "exemplarCity": "Libreville"
  },
  "Lome": {
    "exemplarCity": "Lomé"
  },
  "Luanda": {
    "exemplarCity": "Luanda"
  },
  "Lubumbashi": {
    "exemplarCity": "Lubumbashi"
  },
  "Lusaka": {
    "exemplarCity": "Lusaka"
  },
  "Malabo": {
    "exemplarCity": "Malabo"
  },
  "Maputo": {
    "exemplarCity": "Maputo"
  },
  "Maseru": {
    "exemplarCity": "Maseru"
  },
  "Mbabane": {
    "exemplarCity": "Mbabane"
  },
  "Mogadishu": {
    "exemplarCity": "Mogadischu"
  },
  "Monrovia": {
    "exemplarCity": "Monrovia"
  },
  "Nairobi": {
    "exemplarCity": "Nairobi"
  },
  "Ndjamena": {
    "exemplarCity": "N'Djamena"
  },
  "Niamey": {
    "exemplarCity": "Niamey"
  },
  "Nouakchott": {
```



```
        "exemplarCity": "Nouakchott"
      },
      "Ouagadougou": {
        "exemplarCity": "Ouagadougou"
      },
      "Porto-Novo": {
        "exemplarCity": "Porto Novo"
      },
      "Sao_Tome": {
        "exemplarCity": "São Tomé"
      },
      "Tripoli": {
        "exemplarCity": "Tripolis"
      },
      "Tunis": {
        "exemplarCity": "Tunis"
      },
      "Windhoek": {
        "exemplarCity": "Windhoek"
      }
    },
    "Asia": {
      "Aden": {
        "exemplarCity": "Aden"
      },
      "Almaty": {
        "exemplarCity": "Almaty"
      },
      "Amman": {
        "exemplarCity": "Amman"
      },
      "Anadyr": {
        "exemplarCity": "Anadyr"
      },
      "Aqtai": {
        "exemplarCity": "Aqtai"
      },
      "Aqtobe": {
        "exemplarCity": "Aktobe"
      },
      "Ashgabat": {
        "exemplarCity": "Aşgabat"
      },
      "Baghdad": {
        "exemplarCity": "Bagdad"
      },
      "Bahrain": {
        "exemplarCity": "Bahrain"
      },
      "Baku": {
        "exemplarCity": "Baku"
      },
      "Bangkok": {
        "exemplarCity": "Bangkok"
      },
      "Barnaul": {
        "exemplarCity": "Barnaul"
      }
    }
  }
}
```

```
    },
    "Beirut": {
      "exemplarCity": "Beirut"
    },
    "Bishkek": {
      "exemplarCity": "Bischkek"
    },
    "Brunei": {
      "exemplarCity": "Brunei"
    },
    "Calcutta": {
      "exemplarCity": "Kalkutta"
    },
    "Chita": {
      "exemplarCity": "Tschita"
    },
    "Choibalsan": {
      "exemplarCity": "Tschoibalsan"
    },
    "Colombo": {
      "exemplarCity": "Colombo"
    },
    "Damascus": {
      "exemplarCity": "Damaskus"
    },
    "Dhaka": {
      "exemplarCity": "Dhaka"
    },
    "Dili": {
      "exemplarCity": "Dili"
    },
    "Dubai": {
      "exemplarCity": "Dubai"
    },
    "Dushanbe": {
      "exemplarCity": "Duschanbe"
    },
    "Gaza": {
      "exemplarCity": "Gaza"
    },
    "Hebron": {
      "exemplarCity": "Hebron"
    },
    "Hong_Kong": {
      "exemplarCity": "Hongkong"
    },
    "Hovd": {
      "exemplarCity": "Chowd"
    },
    "Irkutsk": {
      "exemplarCity": "Irkutsk"
    },
    "Jakarta": {
      "exemplarCity": "Jakarta"
    },
    "Jayapura": {
      "exemplarCity": "Jayapura"
    }
```

```
    },  
    "Jerusalem": {  
      "exemplarCity": "Jerusalem"  
    },  
    "Kabul": {  
      "exemplarCity": "Kabul"  
    },  
    "Kamchatka": {  
      "exemplarCity": "Kamtschatka"  
    },  
    "Karachi": {  
      "exemplarCity": "Karatschi"  
    },  
    "Katmandu": {  
      "exemplarCity": "Kathmandu"  
    },  
    "Khandyga": {  
      "exemplarCity": "Chandyga"  
    },  
    "Krasnoyarsk": {  
      "exemplarCity": "Krasnojarsk"  
    },  
    "Kuala_Lumpur": {  
      "exemplarCity": "Kuala Lumpur"  
    },  
    "Kuching": {  
      "exemplarCity": "Kuching"  
    },  
    "Kuwait": {  
      "exemplarCity": "Kuwait"  
    },  
    "Macau": {  
      "exemplarCity": "Macao"  
    },  
    "Magadan": {  
      "exemplarCity": "Magadan"  
    },  
    "Makassar": {  
      "exemplarCity": "Makassar"  
    },  
    "Manila": {  
      "exemplarCity": "Manila"  
    },  
    "Muscat": {  
      "exemplarCity": "Maskat"  
    },  
    "Nicosia": {  
      "exemplarCity": "Nikosia"  
    },  
    "Novokuznetsk": {  
      "exemplarCity": "Nowokuznetsk"  
    },  
    "Novosibirsk": {  
      "exemplarCity": "Nowosibirsk"  
    },  
    "Omsk": {  
      "exemplarCity": "Omsk"  
    }  
  }  
}
```

```
    },
    "Oral": {
      "exemplarCity": "Oral"
    },
    "Phnom_Penh": {
      "exemplarCity": "Phnom Penh"
    },
    "Pontianak": {
      "exemplarCity": "Pontianak"
    },
    "Pyongyang": {
      "exemplarCity": "Pjöngjang"
    },
    "Qatar": {
      "exemplarCity": "Katar"
    },
    "Qyzylorda": {
      "exemplarCity": "Qysylorda"
    },
    "Rangoon": {
      "exemplarCity": "Rangun"
    },
    "Riyadh": {
      "exemplarCity": "Riad"
    },
    "Saigon": {
      "exemplarCity": "Ho-Chi-Minh-Stadt"
    },
    "Sakhalin": {
      "exemplarCity": "Sachalin"
    },
    "Samarkand": {
      "exemplarCity": "Samarkand"
    },
    "Seoul": {
      "exemplarCity": "Seoul"
    },
    "Shanghai": {
      "exemplarCity": "Shanghai"
    },
    "Singapore": {
      "exemplarCity": "Singapur"
    },
    "Srednekolymsk": {
      "exemplarCity": "Srednekolymsk"
    },
    "Taipei": {
      "exemplarCity": "Taipeh"
    },
    "Tashkent": {
      "exemplarCity": "Taschkent"
    },
    "Tbilisi": {
      "exemplarCity": "Tiflis"
    },
    "Tehran": {
      "exemplarCity": "Teheran"
    }
  }
```

```
    },
    "Thimphu": {
      "exemplarCity": "Thimphu"
    },
    "Tokyo": {
      "exemplarCity": "Tokio"
    },
    "Tomsk": {
      "exemplarCity": "Tomsk"
    },
    "Ulaanbaatar": {
      "exemplarCity": "Ulaanbaatar"
    },
    "Urumqi": {
      "exemplarCity": "Ürümqi"
    },
    "Ust-Nera": {
      "exemplarCity": "Ust-Nera"
    },
    "Vientiane": {
      "exemplarCity": "Vientiane"
    },
    "Vladivostok": {
      "exemplarCity": "Wladiwostok"
    },
    "Yakutsk": {
      "exemplarCity": "Jakutsk"
    },
    "Yekaterinburg": {
      "exemplarCity": "Jekaterinburg"
    },
    "Yerevan": {
      "exemplarCity": "Eriwan"
    }
  },
  "Indian": {
    "Antananarivo": {
      "exemplarCity": "Antananarivo"
    },
    "Chagos": {
      "exemplarCity": "Chagos"
    },
    "Christmas": {
      "exemplarCity": "Weihnachtsinsel"
    },
    "Cocos": {
      "exemplarCity": "Cocos"
    },
    "Comoro": {
      "exemplarCity": "Komoren"
    },
    "Kerguelen": {
      "exemplarCity": "Kerguelen"
    },
    "Mahe": {
      "exemplarCity": "Mahe"
    }
  },
}
```

```
"Maldives": {
  "exemplarCity": "Malediven"
},
"Mauritius": {
  "exemplarCity": "Mauritius"
},
"Mayotte": {
  "exemplarCity": "Mayotte"
},
"Reunion": {
  "exemplarCity": "Réunion"
}
},
"Australia": {
  "Adelaide": {
    "exemplarCity": "Adelaide"
  },
  "Brisbane": {
    "exemplarCity": "Brisbane"
  },
  "Broken_Hill": {
    "exemplarCity": "Broken Hill"
  },
  "Currie": {
    "exemplarCity": "Currie"
  },
  "Darwin": {
    "exemplarCity": "Darwin"
  },
  "Eucla": {
    "exemplarCity": "Eucla"
  },
  "Hobart": {
    "exemplarCity": "Hobart"
  },
  "Lindeman": {
    "exemplarCity": "Lindeman"
  },
  "Lord_Howe": {
    "exemplarCity": "Lord Howe"
  },
  "Melbourne": {
    "exemplarCity": "Melbourne"
  },
  "Perth": {
    "exemplarCity": "Perth"
  },
  "Sydney": {
    "exemplarCity": "Sydney"
  }
},
"Pacific": {
  "Apia": {
    "exemplarCity": "Apia"
  },
  "Auckland": {
    "exemplarCity": "Auckland"
  }
}
```

```
    },  
    "Bougainville": {  
      "exemplarCity": "Bougainville"  
    },  
    "Chatham": {  
      "exemplarCity": "Chatham"  
    },  
    "Easter": {  
      "exemplarCity": "Osterinsel"  
    },  
    "Efate": {  
      "exemplarCity": "Efate"  
    },  
    "Enderbury": {  
      "exemplarCity": "Enderbury"  
    },  
    "Fakaofu": {  
      "exemplarCity": "Fakaofu"  
    },  
    "Fiji": {  
      "exemplarCity": "Fidschi"  
    },  
    "Funafuti": {  
      "exemplarCity": "Funafuti"  
    },  
    "Galapagos": {  
      "exemplarCity": "Galapagos"  
    },  
    "Gambier": {  
      "exemplarCity": "Gambier"  
    },  
    "Guadalcanal": {  
      "exemplarCity": "Guadalcanal"  
    },  
    "Guam": {  
      "exemplarCity": "Guam"  
    },  
    "Honolulu": {  
      "exemplarCity": "Honolulu"  
    },  
    "Johnston": {  
      "exemplarCity": "Johnston"  
    },  
    "Kiritimati": {  
      "exemplarCity": "Kiritimati"  
    },  
    "Kosrae": {  
      "exemplarCity": "Kosrae"  
    },  
    "Kwajalein": {  
      "exemplarCity": "Kwajalein"  
    },  
    "Majuro": {  
      "exemplarCity": "Majuro"  
    },  
    "Marquesas": {  
      "exemplarCity": "Marquesas"
```

```
    },
    "Midway": {
      "exemplarCity": "Midway"
    },
    "Nauru": {
      "exemplarCity": "Nauru"
    },
    "Niue": {
      "exemplarCity": "Niue"
    },
    "Norfolk": {
      "exemplarCity": "Norfolk"
    },
    "Noumea": {
      "exemplarCity": "Noumea"
    },
    "Pago_Pago": {
      "exemplarCity": "Pago Pago"
    },
    "Palau": {
      "exemplarCity": "Palau"
    },
    "Pitcairn": {
      "exemplarCity": "Pitcairn"
    },
    "Ponape": {
      "exemplarCity": "Pohnpei"
    },
    "Port_Moresby": {
      "exemplarCity": "Port Moresby"
    },
    "Rarotonga": {
      "exemplarCity": "Rarotonga"
    },
    "Saipan": {
      "exemplarCity": "Saipan"
    },
    "Tahiti": {
      "exemplarCity": "Tahiti"
    },
    "Tarawa": {
      "exemplarCity": "Tarawa"
    },
    "Tongatapu": {
      "exemplarCity": "Tongatapu"
    },
    "Truk": {
      "exemplarCity": "Chuuk"
    },
    "Wake": {
      "exemplarCity": "Wake"
    },
    "Wallis": {
      "exemplarCity": "Wallis"
    }
  },
  "Arctic": {
```



```
    "Longyearbyen": {
      "exemplarCity": "Longyearbyen"
    },
    "Antarctica": {
      "Casey": {
        "exemplarCity": "Casey"
      },
      "Davis": {
        "exemplarCity": "Davis"
      },
      "DumontDUrville": {
        "exemplarCity": "Dumont d'Urville"
      },
      "Macquarie": {
        "exemplarCity": "Macquarie"
      },
      "Mawson": {
        "exemplarCity": "Mawson"
      },
      "McMurdo": {
        "exemplarCity": "McMurdo"
      },
      "Palmer": {
        "exemplarCity": "Palmer"
      },
      "Rothera": {
        "exemplarCity": "Rothera"
      },
      "Syowa": {
        "exemplarCity": "Syowa"
      },
      "Troll": {
        "exemplarCity": "Troll"
      },
      "Vostok": {
        "exemplarCity": "Wostok"
      }
    },
    "Etc": {
      "GMT": {
        "exemplarCity": "GMT"
      },
      "GMT1": {
        "exemplarCity": "GMT+1"
      },
      "GMT10": {
        "exemplarCity": "GMT+10"
      },
      "GMT11": {
        "exemplarCity": "GMT+11"
      },
      "GMT12": {
        "exemplarCity": "GMT+12"
      },
      "GMT2": {
        "exemplarCity": "GMT+2"
      }
    }
  }
}
```

```
    },  
    "GMT3": {  
      "exemplarCity": "GMT+3"  
    },  
    "GMT4": {  
      "exemplarCity": "GMT+4"  
    },  
    "GMT5": {  
      "exemplarCity": "GMT+5"  
    },  
    "GMT6": {  
      "exemplarCity": "GMT+6"  
    },  
    "GMT7": {  
      "exemplarCity": "GMT+7"  
    },  
    "GMT8": {  
      "exemplarCity": "GMT+8"  
    },  
    "GMT9": {  
      "exemplarCity": "GMT+9"  
    },  
    "GMT-1": {  
      "exemplarCity": "GMT-1"  
    },  
    "GMT-10": {  
      "exemplarCity": "GMT-10"  
    },  
    "GMT-11": {  
      "exemplarCity": "GMT-11"  
    },  
    "GMT-12": {  
      "exemplarCity": "GMT-12"  
    },  
    "GMT-13": {  
      "exemplarCity": "GMT-13"  
    },  
    "GMT-14": {  
      "exemplarCity": "GMT-14"  
    },  
    "GMT-2": {  
      "exemplarCity": "GMT-2"  
    },  
    "GMT-3": {  
      "exemplarCity": "GMT-3"  
    },  
    "GMT-4": {  
      "exemplarCity": "GMT-4"  
    },  
    "GMT-5": {  
      "exemplarCity": "GMT-5"  
    },  
    "GMT-6": {  
      "exemplarCity": "GMT-6"  
    },  
    "GMT-7": {  
      "exemplarCity": "GMT-7"
```

```

    },
    "GMT-8": {
      "exemplarCity": "GMT-8"
    },
    "GMT-9": {
      "exemplarCity": "GMT-9"
    },
    "Unknown": {
      "exemplarCity": "Unbekannt"
    }
  },
  "metazone": {
    "Acre": {
      "long": {
        "generic": "Acre-Zeit",
        "standard": "Acre-Normalzeit",
        "daylight": "Acre-Sommerzeit"
      }
    },
    "Afghanistan": {
      "long": {
        "standard": "Afghanistan-Zeit"
      }
    },
    "Africa_Central": {
      "long": {
        "standard": "Zentralafrikanische Zeit"
      }
    },
    "Africa_Eastern": {
      "long": {
        "standard": "Ostafrikanische Zeit"
      }
    },
    "Africa_Southern": {
      "long": {
        "standard": "Südafrikanische Zeit"
      }
    },
    "Africa_Western": {
      "long": {
        "generic": "Westafrikanische Zeit",
        "standard": "Westafrikanische Normalzeit",
        "daylight": "Westafrikanische Sommerzeit"
      }
    },
    "Alaska": {
      "long": {
        "generic": "Alaska-Zeit",
        "standard": "Alaska-Normalzeit",
        "daylight": "Alaska-Sommerzeit"
      }
    },
    "Almaty": {
      "long": {
        "generic": "Almaty-Zeit",

```

```

        "standard": "Almaty-Normalzeit",
        "daylight": "Almaty-Sommerzeit"
    },
    },
    "Amazon": {
        "long": {
            "generic": "Amazonas-Zeit",
            "standard": "Amazonas-Normalzeit",
            "daylight": "Amazonas-Sommerzeit"
        }
    },
    "America_Central": {
        "long": {
            "generic": "Nordamerikanische Inlandzeit",
            "standard": "Nordamerikanische Inland-Normalzeit",
            "daylight": "Nordamerikanische Inland-Sommerzeit"
        }
    },
    "America_Eastern": {
        "long": {
            "generic": "Nordamerikanische Ostküstenzeit",
            "standard": "Nordamerikanische Ostküsten-Normalzeit",
            "daylight": "Nordamerikanische Ostküsten-Sommerzeit"
        }
    },
    "America_Mountain": {
        "long": {
            "generic": "Rocky-Mountain-Zeit",
            "standard": "Rocky Mountain-Normalzeit",
            "daylight": "Rocky-Mountain-Sommerzeit"
        }
    },
    "America_Pacific": {
        "long": {
            "generic": "Nordamerikanische Westküstenzeit",
            "standard": "Nordamerikanische Westküsten-Normalzeit",
            "daylight": "Nordamerikanische Westküsten-Sommerzeit"
        }
    },
    "Anadyr": {
        "long": {
            "generic": "Anadyr Zeit",
            "standard": "Anadyr Normalzeit",
            "daylight": "Anadyr Sommerzeit"
        }
    },
    "Apia": {
        "long": {
            "generic": "Apia-Zeit",
            "standard": "Apia-Normalzeit",
            "daylight": "Apia-Sommerzeit"
        }
    },
    "Aqttau": {
        "long": {
            "generic": "Aqttau-Zeit",
            "standard": "Aqttau-Normalzeit",

```

```

        "daylight": "Aqtau-Sommerzeit"
    },
    },
    "Aqtobe": {
        "long": {
            "generic": "Aqtöbe-Zeit",
            "standard": "Aqtöbe-Normalzeit",
            "daylight": "Aqtöbe-Sommerzeit"
        }
    },
    "Arabian": {
        "long": {
            "generic": "Arabische Zeit",
            "standard": "Arabische Normalzeit",
            "daylight": "Arabische Sommerzeit"
        }
    },
    "Argentina": {
        "long": {
            "generic": "Argentinische Zeit",
            "standard": "Argentinische Normalzeit",
            "daylight": "Argentinische Sommerzeit"
        }
    },
    "Argentina_Western": {
        "long": {
            "generic": "Westargentinische Zeit",
            "standard": "Westargentinische Normalzeit",
            "daylight": "Westargentinische Sommerzeit"
        }
    },
    "Armenia": {
        "long": {
            "generic": "Armenische Zeit",
            "standard": "Armenische Normalzeit",
            "daylight": "Armenische Sommerzeit"
        }
    },
    "Atlantic": {
        "long": {
            "generic": "Atlantik-Zeit",
            "standard": "Atlantik-Normalzeit",
            "daylight": "Atlantik-Sommerzeit"
        }
    },
    "Australia_Central": {
        "long": {
            "generic": "Zentralaustralische Zeit",
            "standard": "Zentralaustralische Normalzeit",
            "daylight": "Zentralaustralische Sommerzeit"
        }
    },
    "Australia_CentralWestern": {
        "long": {
            "generic": "Zentral-/Westaustralische Zeit",
            "standard": "Zentral-/Westaustralische Normalzeit",
            "daylight": "Zentral-/Westaustralische Sommerzeit"
        }
    }
}

```

```

    }
  },
  "Australia_Eastern": {
    "long": {
      "generic": "Ostaustralische Zeit",
      "standard": "Ostaustralische Normalzeit",
      "daylight": "Ostaustralische Sommerzeit"
    }
  },
  "Australia_Western": {
    "long": {
      "generic": "Westaustralische Zeit",
      "standard": "Westaustralische Normalzeit",
      "daylight": "Westaustralische Sommerzeit"
    }
  },
  "Azerbaijan": {
    "long": {
      "generic": "Aserbaidtschanische Zeit",
      "standard": "Aserbeidschanische Normalzeit",
      "daylight": "Aserbaidtschanische Sommerzeit"
    }
  },
  "Azores": {
    "long": {
      "generic": "Azoren-Zeit",
      "standard": "Azoren-Normalzeit",
      "daylight": "Azoren-Sommerzeit"
    }
  },
  "Bangladesh": {
    "long": {
      "generic": "Bangladesch-Zeit",
      "standard": "Bangladesch-Normalzeit",
      "daylight": "Bangladesch-Sommerzeit"
    }
  },
  "Bhutan": {
    "long": {
      "standard": "Bhutan-Zeit"
    }
  },
  "Bolivia": {
    "long": {
      "standard": "Bolivianische Zeit"
    }
  },
  "Brasilia": {
    "long": {
      "generic": "Brasília-Zeit",
      "standard": "Brasília-Normalzeit",
      "daylight": "Brasília-Sommerzeit"
    }
  },
  "Brunei": {
    "long": {
      "standard": "Brunei-Zeit"
    }
  }
}

```

```

    }
  },
  "Cape_Verde": {
    "long": {
      "generic": "Cabo-Verde-Zeit",
      "standard": "Cabo-Verde-Normalzeit",
      "daylight": "Cabo-Verde-Sommerzeit"
    }
  },
  "Casey": {
    "long": {
      "standard": "Casey-Zeit"
    }
  },
  "Chamorro": {
    "long": {
      "standard": "Chamorro-Zeit"
    }
  },
  "Chatham": {
    "long": {
      "generic": "Chatham-Zeit",
      "standard": "Chatham-Normalzeit",
      "daylight": "Chatham-Sommerzeit"
    }
  },
  "Chile": {
    "long": {
      "generic": "Chilenische Zeit",
      "standard": "Chilenische Normalzeit",
      "daylight": "Chilenische Sommerzeit"
    }
  },
  "China": {
    "long": {
      "generic": "Chinesische Zeit",
      "standard": "Chinesische Normalzeit",
      "daylight": "Chinesische Sommerzeit"
    }
  },
  "Choibalsan": {
    "long": {
      "generic": "Tschoibalsan-Zeit",
      "standard": "Tschoibalsan-Normalzeit",
      "daylight": "Tschoibalsan-Sommerzeit"
    }
  },
  "Christmas": {
    "long": {
      "standard": "Weihnachtsinsel-Zeit"
    }
  },
  "Cocos": {
    "long": {
      "standard": "Kokosinseln-Zeit"
    }
  },
  },

```

```

"Colombia": {
  "long": {
    "generic": "Kolumbianische Zeit",
    "standard": "Kolumbianische Normalzeit",
    "daylight": "Kolumbianische Sommerzeit"
  }
},
"Cook": {
  "long": {
    "generic": "Cookinseln-Zeit",
    "standard": "Cookinseln-Normalzeit",
    "daylight": "Cookinseln-Sommerzeit"
  }
},
"Cuba": {
  "long": {
    "generic": "Kubanische Zeit",
    "standard": "Kubanische Normalzeit",
    "daylight": "Kubanische Sommerzeit"
  }
},
"Davis": {
  "long": {
    "standard": "Davis-Zeit"
  }
},
"DumontDUrville": {
  "long": {
    "standard": "Dumont-d'Urville-Zeit"
  }
},
"East_Timor": {
  "long": {
    "standard": "Osttimor-Zeit"
  }
},
"Easter": {
  "long": {
    "generic": "Osterinsel-Zeit",
    "standard": "Osterinsel-Normalzeit",
    "daylight": "Osterinsel-Sommerzeit"
  }
},
"Ecuador": {
  "long": {
    "standard": "Ecuadorianische Zeit"
  }
},
"Europe_Central": {
  "long": {
    "generic": "Mitteleuropäische Zeit",
    "standard": "Mitteleuropäische Normalzeit",
    "daylight": "Mitteleuropäische Sommerzeit"
  },
  "short": {
    "generic": "MEZ",
    "standard": "MEZ",

```



```

        "daylight": "MESZ"
    },
    },
    "Europe_Eastern": {
        "long": {
            "generic": "Osteuropäische Zeit",
            "standard": "Osteuropäische Normalzeit",
            "daylight": "Osteuropäische Sommerzeit"
        },
        "short": {
            "generic": "OEZ",
            "standard": "OEZ",
            "daylight": "OESZ"
        }
    },
    "Europe_Further_Eastern": {
        "long": {
            "standard": "Kaliningrader Zeit"
        }
    },
    "Europe_Western": {
        "long": {
            "generic": "Westeuropäische Zeit",
            "standard": "Westeuropäische Normalzeit",
            "daylight": "Westeuropäische Sommerzeit"
        },
        "short": {
            "generic": "WEZ",
            "standard": "WEZ",
            "daylight": "WESZ"
        }
    },
    "Falkland": {
        "long": {
            "generic": "Falklandinseln-Zeit",
            "standard": "Falklandinseln-Normalzeit",
            "daylight": "Falklandinseln-Sommerzeit"
        }
    },
    "Fiji": {
        "long": {
            "generic": "Fidschi-Zeit",
            "standard": "Fidschi-Normalzeit",
            "daylight": "Fidschi-Sommerzeit"
        }
    },
    "French_Guiana": {
        "long": {
            "standard": "Französisch-Guayana-Zeit"
        }
    },
    "French_Southern": {
        "long": {
            "standard": "Französische Süd- und Antarktisgebiete-Zeit"
        }
    },
    "Galapagos": {

```

```
        "long": {
          "standard": "Galapagos-Zeit"
        }
      },
      "Gambier": {
        "long": {
          "standard": "Gambier-Zeit"
        }
      },
      "Georgia": {
        "long": {
          "generic": "Georgische Zeit",
          "standard": "Georgische Normalzeit",
          "daylight": "Georgische Sommerzeit"
        }
      },
      "Gilbert_Islands": {
        "long": {
          "standard": "Gilbert-Inseln-Zeit"
        }
      },
      "GMT": {
        "long": {
          "standard": "Mittlere Greenwich-Zeit"
        }
      },
      "Greenland_Eastern": {
        "long": {
          "generic": "Ostgrönland-Zeit",
          "standard": "Ostgrönland-Normalzeit",
          "daylight": "Ostgrönland-Sommerzeit"
        }
      },
      "Greenland_Western": {
        "long": {
          "generic": "Westgrönland-Zeit",
          "standard": "Westgrönland-Normalzeit",
          "daylight": "Westgrönland-Sommerzeit"
        }
      },
      "Guam": {
        "long": {
          "standard": "Guam-Zeit"
        }
      },
      "Gulf": {
        "long": {
          "standard": "Golf-Zeit"
        }
      },
      "Guyana": {
        "long": {
          "standard": "Guyana-Zeit"
        }
      },
      "Hawaii_Aleutian": {
        "long": {
```

```

        "generic": "Hawaii-Aleuten-Zeit",
        "standard": "Hawaii-Aleuten-Normalzeit",
        "daylight": "Hawaii-Aleuten-Sommerzeit"
    }
},
"Hong_Kong": {
    "long": {
        "generic": "Hongkong-Zeit",
        "standard": "Hongkong-Normalzeit",
        "daylight": "Hongkong-Sommerzeit"
    }
},
"Hovd": {
    "long": {
        "generic": "Chowd-Zeit",
        "standard": "Chowd-Normalzeit",
        "daylight": "Chowd-Sommerzeit"
    }
},
"India": {
    "long": {
        "standard": "Indische Zeit"
    }
},
"Indian_Ocean": {
    "long": {
        "standard": "Indischer Ozean-Zeit"
    }
},
"Indochina": {
    "long": {
        "standard": "Indochina-Zeit"
    }
},
"Indonesia_Central": {
    "long": {
        "standard": "Zentralindonesische Zeit"
    }
},
"Indonesia_Eastern": {
    "long": {
        "standard": "Ostindonesische Zeit"
    }
},
"Indonesia_Western": {
    "long": {
        "standard": "Westindonesische Zeit"
    }
},
"Iran": {
    "long": {
        "generic": "Iranische Zeit",
        "standard": "Iranische Normalzeit",
        "daylight": "Iranische Sommerzeit"
    }
},
"Irkutsk": {

```

```
    "long": {
      "generic": "Irkutsk-Zeit",
      "standard": "Irkutsk-Normalzeit",
      "daylight": "Irkutsk-Sommerzeit"
    }
  },
  "Israel": {
    "long": {
      "generic": "Israelische Zeit",
      "standard": "Israelische Normalzeit",
      "daylight": "Israelische Sommerzeit"
    }
  },
  "Japan": {
    "long": {
      "generic": "Japanische Zeit",
      "standard": "Japanische Normalzeit",
      "daylight": "Japanische Sommerzeit"
    }
  },
  "Kamchatka": {
    "long": {
      "generic": "Kamtschatka-Zeit",
      "standard": "Kamtschatka-Normalzeit",
      "daylight": "Kamtschatka-Sommerzeit"
    }
  },
  "Kazakhstan_Eastern": {
    "long": {
      "standard": "Ostkasachische Zeit"
    }
  },
  "Kazakhstan_Western": {
    "long": {
      "standard": "Westkasachische Zeit"
    }
  },
  "Korea": {
    "long": {
      "generic": "Koreanische Zeit",
      "standard": "Koreanische Normalzeit",
      "daylight": "Koreanische Sommerzeit"
    }
  },
  "Kosrae": {
    "long": {
      "standard": "Kosrae-Zeit"
    }
  },
  "Krasnoyarsk": {
    "long": {
      "generic": "Krasnojarsk-Zeit",
      "standard": "Krasnojarsk-Normalzeit",
      "daylight": "Krasnojarsk-Sommerzeit"
    }
  },
  "Kyrgystan": {
```

```

        "long": {
          "standard": "Kirgisistan-Zeit"
        }
      },
      "Lanka": {
        "long": {
          "standard": "Sri-Lanka-Zeit"
        }
      },
      "Line_Islands": {
        "long": {
          "standard": "Linieninseln-Zeit"
        }
      },
      "Lord_Howe": {
        "long": {
          "generic": "Lord-Howe-Zeit",
          "standard": "Lord-Howe-Normalzeit",
          "daylight": "Lord-Howe-Sommerzeit"
        }
      },
      "Macau": {
        "long": {
          "generic": "Macau-Zeit",
          "standard": "Macau-Normalzeit",
          "daylight": "Macau-Sommerzeit"
        }
      },
      "Macquarie": {
        "long": {
          "standard": "Macquarieinsel-Zeit"
        }
      },
      "Magadan": {
        "long": {
          "generic": "Magadan-Zeit",
          "standard": "Magadan-Normalzeit",
          "daylight": "Magadan-Sommerzeit"
        }
      },
      "Malaysia": {
        "long": {
          "standard": "Malaysische Zeit"
        }
      },
      "Maldives": {
        "long": {
          "standard": "Malediven-Zeit"
        }
      },
      "Marquesas": {
        "long": {
          "standard": "Marquesas-Zeit"
        }
      },
      "Marshall_Islands": {
        "long": {

```

```

        "standard": "Marshallinseln-Zeit"
    },
    },
    "Mauritius": {
        "long": {
            "generic": "Mauritius-Zeit",
            "standard": "Mauritius-Normalzeit",
            "daylight": "Mauritius-Sommerzeit"
        }
    },
    "Mawson": {
        "long": {
            "standard": "Mawson-Zeit"
        }
    },
    },
    "Mexico_Northwest": {
        "long": {
            "generic": "Mexiko Nordwestliche Zone-Zeit",
            "standard": "Mexiko Nordwestliche Zone-Normalzeit",
            "daylight": "Mexiko Nordwestliche Zone-Sommerzeit"
        }
    },
    },
    "Mexico_Pacific": {
        "long": {
            "generic": "Mexiko Pazifikzone-Zeit",
            "standard": "Mexiko Pazifikzone-Normalzeit",
            "daylight": "Mexiko Pazifikzone-Sommerzeit"
        }
    },
    },
    "Mongolia": {
        "long": {
            "generic": "Ulaanbaatar-Zeit",
            "standard": "Ulaanbaatar-Normalzeit",
            "daylight": "Ulaanbaatar-Sommerzeit"
        }
    },
    },
    "Moscow": {
        "long": {
            "generic": "Moskauer Zeit",
            "standard": "Moskauer Normalzeit",
            "daylight": "Moskauer Sommerzeit"
        }
    },
    },
    "Myanmar": {
        "long": {
            "standard": "Myanmar-Zeit"
        }
    },
    },
    "Nauru": {
        "long": {
            "standard": "Nauru-Zeit"
        }
    },
    },
    "Nepal": {
        "long": {
            "standard": "Nepalesische Zeit"
        }
    }
}

```

```

    },
    "New_Caledonia": {
      "long": {
        "generic": "Neukaledonische Zeit",
        "standard": "Neukaledonische Normalzeit",
        "daylight": "Neukaledonische Sommerzeit"
      }
    },
    "New_Zealand": {
      "long": {
        "generic": "Neuseeland-Zeit",
        "standard": "Neuseeland-Normalzeit",
        "daylight": "Neuseeland-Sommerzeit"
      }
    },
    "Newfoundland": {
      "long": {
        "generic": "Neufundland-Zeit",
        "standard": "Neufundland-Normalzeit",
        "daylight": "Neufundland-Sommerzeit"
      }
    },
    "Niue": {
      "long": {
        "standard": "Niue-Zeit"
      }
    },
    "Norfolk": {
      "long": {
        "standard": "Norfolkinsel-Zeit"
      }
    },
    "Noronha": {
      "long": {
        "generic": "Fernando de Noronha-Zeit",
        "standard": "Fernando de Noronha-Normalzeit",
        "daylight": "Fernando de Noronha-Sommerzeit"
      }
    },
    "North_Mariana": {
      "long": {
        "standard": "Nördliche-Marianen-Zeit"
      }
    },
    "Novosibirsk": {
      "long": {
        "generic": "Nowosibirsk-Zeit",
        "standard": "Nowosibirsk-Normalzeit",
        "daylight": "Nowosibirsk-Sommerzeit"
      }
    },
    "Omsk": {
      "long": {
        "generic": "Omsk-Zeit",
        "standard": "Omsk-Normalzeit",
        "daylight": "Omsk-Sommerzeit"
      }
    }
  }

```

```

    },
    "Pakistan": {
      "long": {
        "generic": "Pakistanische Zeit",
        "standard": "Pakistanische Normalzeit",
        "daylight": "Pakistanische Sommerzeit"
      }
    },
    "Palau": {
      "long": {
        "standard": "Palau-Zeit"
      }
    },
    "Papua_New_Guinea": {
      "long": {
        "standard": "Papua-Neuguinea-Zeit"
      }
    },
    "Paraguay": {
      "long": {
        "generic": "Paraguayische Zeit",
        "standard": "Paraguayische Normalzeit",
        "daylight": "Paraguayische Sommerzeit"
      }
    },
    "Peru": {
      "long": {
        "generic": "Peruanische Zeit",
        "standard": "Peruanische Normalzeit",
        "daylight": "Peruanische Sommerzeit"
      }
    },
    "Philippines": {
      "long": {
        "generic": "Philippinische Zeit",
        "standard": "Philippinische Normalzeit",
        "daylight": "Philippinische Sommerzeit"
      }
    },
    "Phoenix_Islands": {
      "long": {
        "standard": "Phoenixinseln-Zeit"
      }
    },
    "Pierre_Miquelon": {
      "long": {
        "generic": "Saint-Pierre-und-Miquelon-Zeit",
        "standard": "Saint-Pierre-und-Miquelon-Normalzeit",
        "daylight": "Saint-Pierre-und-Miquelon-Sommerzeit"
      }
    },
    "Pitcairn": {
      "long": {
        "standard": "Pitcairninseln-Zeit"
      }
    },
    "Ponape": {

```



```

        "long": {
            "standard": "Ponape-Zeit"
        }
    },
    "Pyongyang": {
        "long": {
            "standard": "Pjöngjang-Zeit"
        }
    },
    "Qyzylorda": {
        "long": {
            "generic": "Quysylorda-Zeit",
            "standard": "Quysylorda-Normalzeit",
            "daylight": "Qysylorda-Sommerzeit"
        }
    },
    "Reunion": {
        "long": {
            "standard": "Réunion-Zeit"
        }
    },
    "Rothera": {
        "long": {
            "standard": "Rothera-Zeit"
        }
    },
    "Sakhalin": {
        "long": {
            "generic": "Sachalin-Zeit",
            "standard": "Sachalin-Normalzeit",
            "daylight": "Sachalin-Sommerzeit"
        }
    },
    "Samara": {
        "long": {
            "generic": "Samara-Zeit",
            "standard": "Samara-Normalzeit",
            "daylight": "Samara-Sommerzeit"
        }
    },
    "Samoa": {
        "long": {
            "generic": "Samoa-Zeit",
            "standard": "Samoa-Normalzeit",
            "daylight": "Samoa-Sommerzeit"
        }
    },
    "Seychelles": {
        "long": {
            "standard": "Seychellen-Zeit"
        }
    },
    "Singapore": {
        "long": {
            "standard": "Singapur-Zeit"
        }
    },
    },

```

```
"Solomon": {
  "long": {
    "standard": "Salomoninseln-Zeit"
  }
},
"South_Georgia": {
  "long": {
    "standard": "Südgeorgische Zeit"
  }
},
"Suriname": {
  "long": {
    "standard": "Suriname-Zeit"
  }
},
"Syowa": {
  "long": {
    "standard": "Syowa-Zeit"
  }
},
"Tahiti": {
  "long": {
    "standard": "Tahiti-Zeit"
  }
},
"Taipei": {
  "long": {
    "generic": "Taipeh-Zeit",
    "standard": "Taipeh-Normalzeit",
    "daylight": "Taipeh-Sommerzeit"
  }
},
"Tajikistan": {
  "long": {
    "standard": "Tadschikistan-Zeit"
  }
},
"Tokelau": {
  "long": {
    "standard": "Tokelau-Zeit"
  }
},
"Tonga": {
  "long": {
    "generic": "Tonganische Zeit",
    "standard": "Tonganische Normalzeit",
    "daylight": "Tonganische Sommerzeit"
  }
},
"Truk": {
  "long": {
    "standard": "Chuuk-Zeit"
  }
},
"Turkmenistan": {
  "long": {
    "generic": "Turkmenistan-Zeit",
```

```

        "standard": "Turkmenistan-Normalzeit",
        "daylight": "Turkmenistan-Sommerzeit"
    },
    },
    "Tuvalu": {
        "long": {
            "standard": "Tuvalu-Zeit"
        }
    },
    "Uruguay": {
        "long": {
            "generic": "Uruguayanische Zeit",
            "standard": "Uruguayanische Normalzeit",
            "daylight": "Uruguayanische Sommerzeit"
        }
    },
    "Uzbekistan": {
        "long": {
            "generic": "Usbekistan-Zeit",
            "standard": "Usbekistan-Normalzeit",
            "daylight": "Usbekistan-Sommerzeit"
        }
    },
    "Vanuatu": {
        "long": {
            "generic": "Vanuatu-Zeit",
            "standard": "Vanuatu-Normalzeit",
            "daylight": "Vanuatu-Sommerzeit"
        }
    },
    "Venezuela": {
        "long": {
            "standard": "Venezuela-Zeit"
        }
    },
    "Vladivostok": {
        "long": {
            "generic": "Wladiwostok-Zeit",
            "standard": "Wladiwostok-Normalzeit",
            "daylight": "Wladiwostok-Sommerzeit"
        }
    },
    "Volgograd": {
        "long": {
            "generic": "Wolgograd-Zeit",
            "standard": "Wolgograd-Normalzeit",
            "daylight": "Wolgograd-Sommerzeit"
        }
    },
    "Vostok": {
        "long": {
            "standard": "Wostok-Zeit"
        }
    },
    "Wake": {
        "long": {
            "standard": "Wake-Insel-Zeit"
        }
    }

```

```
{
    },
    "Wallis": {
        "long": {
            "standard": "Wallis-und-Futuna-Zeit"
        }
    },
    "Yakutsk": {
        "long": {
            "generic": "Jakutsk-Zeit",
            "standard": "Jakutsk-Normalzeit",
            "daylight": "Jakutsk-Sommerzeit"
        }
    },
    "Yekaterinburg": {
        "long": {
            "generic": "Jekaterinburg-Zeit",
            "standard": "Jekaterinburg-Normalzeit",
            "daylight": "Jekaterinburg-Sommerzeit"
        }
    }
}
}
```

TIMEZONENAMES.JSX

```
{
    "main";
    {
        "de";
        {
            "identity";
            {
                "version";
                {
                    "_number";
                    "$Revision: 12879 $" ,
                    "_cldrVersion";
                    "30.0.3";
                }
                "language";
                "de";
            }
        }
        "dates";
        {
            "timeZoneNames";
            {
                "hourFormat";
                "+HH:mm;-HH:mm",
                "gmtFormat";
                "GMT{0}" ,
                "gmtZeroFormat";
```

```

"GMT",
  "regionFormat";
"{0} Zeit",
  "regionFormat-type-daylight";
"{0} Sommerzeit",
  "regionFormat-type-standard";
"{0} Normalzeit",
  "fallbackFormat";
"{1} ({0})",
  "zone";
{
  "America";
  {
    "Adak";
    {
      "exemplarCity";
      "Adak";
    }
    "Anchorage";
    {
      "exemplarCity";
      "Anchorage";
    }
    "Anguilla";
    {
      "exemplarCity";
      "Anguilla";
    }
    "Antigua";
    {
      "exemplarCity";
      "Antigua";
    }
    "Araguaina";
    {
      "exemplarCity";
      "Araguaina";
    }
    "Argentina";
    {
      "Rio_Gallegos";
      {
        "exemplarCity";
        "Rio Gallegos";
      }
      "San_Juan";
      {
        "exemplarCity";
        "San Juan";
      }
      "Ushuaia";
      {
        "exemplarCity";
        "Ushuaia";
      }
      "La_Rioja";
      {

```

```
        "exemplarCity";
        "La Rioja";
    }
    "San_Luis";
    {
        "exemplarCity";
        "San Luis";
    }
    "Salta";
    {
        "exemplarCity";
        "Salta";
    }
    "Tucuman";
    {
        "exemplarCity";
        "Tucuman";
    }
}
"Aruba";
{
    "exemplarCity";
    "Aruba";
}
"Asuncion";
{
    "exemplarCity";
    "Asunción";
}
"Bahia";
{
    "exemplarCity";
    "Bahia";
}
"Bahia_Banderas";
{
    "exemplarCity";
    "Bahia Banderas";
}
"Barbados";
{
    "exemplarCity";
    "Barbados";
}
"Belem";
{
    "exemplarCity";
    "Belem";
}
"Belize";
{
    "exemplarCity";
    "Belize";
}
"Blanc-Sablon";
{
    "exemplarCity";
```

```
        "Blanc-Sablon";
    }
    "Boa_Vista";
    {
        "exemplarCity";
        "Boa Vista";
    }
    "Bogota";
    {
        "exemplarCity";
        "Bogotá";
    }
    "Boise";
    {
        "exemplarCity";
        "Boise";
    }
    "Buenos_Aires";
    {
        "exemplarCity";
        "Buenos Aires";
    }
    "Cambridge_Bay";
    {
        "exemplarCity";
        "Cambridge Bay";
    }
    "Campo_Grande";
    {
        "exemplarCity";
        "Campo Grande";
    }
    "Cancun";
    {
        "exemplarCity";
        "Cancún";
    }
    "Caracas";
    {
        "exemplarCity";
        "Caracas";
    }
    "Catamarca";
    {
        "exemplarCity";
        "Catamarca";
    }
    "Cayenne";
    {
        "exemplarCity";
        "Cayenne";
    }
    "Cayman";
    {
        "exemplarCity";
        "Kaimaninseln";
    }
}
```

```
"Chicago";
{
  "exemplarCity";
  "Chicago";
}
"Chihuahua";
{
  "exemplarCity";
  "Chihuahua";
}
"Coral_Harbour";
{
  "exemplarCity";
  "Atikokan";
}
"Cordoba";
{
  "exemplarCity";
  "Córdoba";
}
"Costa_Rica";
{
  "exemplarCity";
  "Costa Rica";
}
"Creston";
{
  "exemplarCity";
  "Creston";
}
"Cuiaba";
{
  "exemplarCity";
  "Cuiaba";
}
"Curacao";
{
  "exemplarCity";
  "Curaçao";
}
"Danmarkshavn";
{
  "exemplarCity";
  "Danmarkshavn";
}
"Dawson";
{
  "exemplarCity";
  "Dawson";
}
"Dawson_Creek";
{
  "exemplarCity";
  "Dawson Creek";
}
"Denver";
{
```



```
        "exemplarCity";
        "Denver";
    }
    "Detroit";
    {
        "exemplarCity";
        "Detroit";
    }
    "Dominica";
    {
        "exemplarCity";
        "Dominica";
    }
    "Edmonton";
    {
        "exemplarCity";
        "Edmonton";
    }
    "Eirunepe";
    {
        "exemplarCity";
        "Eirunepe";
    }
    "El_Salvador";
    {
        "exemplarCity";
        "El Salvador";
    }
    "Fort_Nelson";
    {
        "exemplarCity";
        "Fort Nelson";
    }
    "Fortaleza";
    {
        "exemplarCity";
        "Fortaleza";
    }
    "Glace_Bay";
    {
        "exemplarCity";
        "Glace Bay";
    }
    "Godthab";
    {
        "exemplarCity";
        "Nuuk";
    }
    "Goose_Bay";
    {
        "exemplarCity";
        "Goose Bay";
    }
    "Grand_Turk";
    {
        "exemplarCity";
        "Grand Turk";
    }
```

```
}
"Grenada";
{
  "exemplarCity";
  "Grenada";
}
"Guadeloupe";
{
  "exemplarCity";
  "Guadeloupe";
}
"Guatemala";
{
  "exemplarCity";
  "Guatemala";
}
"Guayaquil";
{
  "exemplarCity";
  "Guayaquil";
}
"Guyana";
{
  "exemplarCity";
  "Guyana";
}
"Halifax";
{
  "exemplarCity";
  "Halifax";
}
"Havana";
{
  "exemplarCity";
  "Havanna";
}
"Hermosillo";
{
  "exemplarCity";
  "Hermosillo";
}
"Indiana";
{
  "Vincennes";
  {
    "exemplarCity";
    "Vincennes, Indiana";
  }
  "Petersburg";
  {
    "exemplarCity";
    "Petersburg, Indiana";
  }
  "Tell_City";
  {
    "exemplarCity";
    "Tell City, Indiana";
  }
}
```

```
}
  "Knox";
  {
    "exemplarCity";
    "Knox, Indiana";
  }
  "Winamac";
  {
    "exemplarCity";
    "Winamac, Indiana";
  }
  "Marengo";
  {
    "exemplarCity";
    "Marengo, Indiana";
  }
  "Vevay";
  {
    "exemplarCity";
    "Vevay, Indiana";
  }
}
"Indianapolis";
{
  "exemplarCity";
  "Indianapolis";
}
"Inuvik";
{
  "exemplarCity";
  "Inuvik";
}
"Iqaluit";
{
  "exemplarCity";
  "Iqaluit";
}
"Jamaica";
{
  "exemplarCity";
  "Jamaika";
}
"Jujuy";
{
  "exemplarCity";
  "Jujuy";
}
"Juneau";
{
  "exemplarCity";
  "Juneau";
}
"Kentucky";
{
  "Monticello";
  {
    "exemplarCity";
```

```
        "Monticello, Kentucky";
    }
}
"Kralendijk";
{
    "exemplarCity";
    "Kralendijk";
}
"La_Paz";
{
    "exemplarCity";
    "La Paz";
}
"Lima";
{
    "exemplarCity";
    "Lima";
}
"Los_Angeles";
{
    "exemplarCity";
    "Los Angeles";
}
"Louisville";
{
    "exemplarCity";
    "Louisville";
}
"Lower_Princes";
{
    "exemplarCity";
    "Lower Prince's Quarter";
}
"Maceio";
{
    "exemplarCity";
    "Maceio";
}
"Managua";
{
    "exemplarCity";
    "Managua";
}
"Manaus";
{
    "exemplarCity";
    "Manaus";
}
"Marigot";
{
    "exemplarCity";
    "Marigot";
}
"Martinique";
{
    "exemplarCity";
    "Martinique";
}
```

```
}
"Matamoros";
{
  "exemplarCity";
  "Matamoros";
}
"Mazatlan";
{
  "exemplarCity";
  "Mazatlan";
}
"Mendoza";
{
  "exemplarCity";
  "Mendoza";
}
"Menominee";
{
  "exemplarCity";
  "Menominee";
}
"Merida";
{
  "exemplarCity";
  "Merida";
}
"Metlakatla";
{
  "exemplarCity";
  "Metlakatla";
}
"Mexico_City";
{
  "exemplarCity";
  "Mexiko-Stadt";
}
"Miquelon";
{
  "exemplarCity";
  "Miquelon";
}
"Moncton";
{
  "exemplarCity";
  "Moncton";
}
"Monterrey";
{
  "exemplarCity";
  "Monterrey";
}
"Montevideo";
{
  "exemplarCity";
  "Montevideo";
}
"Montserrat";
```

```
{
    "exemplarCity";
    "Montserrat";
}
"Nassau";
{
    "exemplarCity";
    "Nassau";
}
"New_York";
{
    "exemplarCity";
    "New York";
}
"Nipigon";
{
    "exemplarCity";
    "Nipigon";
}
"Nome";
{
    "exemplarCity";
    "Nome";
}
"Noronha";
{
    "exemplarCity";
    "Noronha";
}
"North_Dakota";
{
    "Beulah";
    {
        "exemplarCity";
        "Beulah, North Dakota";
    }
    "New_Salem";
    {
        "exemplarCity";
        "New Salem, North Dakota";
    }
    "Center";
    {
        "exemplarCity";
        "Center, North Dakota";
    }
}
"Ojinaga";
{
    "exemplarCity";
    "Ojinaga";
}
"Panama";
{
    "exemplarCity";
    "Panama";
}
```

```
"Pangnirtung";
{
  "exemplarCity";
  "Pangnirtung";
}
"Paramaribo";
{
  "exemplarCity";
  "Paramaribo";
}
"Phoenix";
{
  "exemplarCity";
  "Phoenix";
}
"Port-au-Prince";
{
  "exemplarCity";
  "Port-au-Prince";
}
"Port_of_Spain";
{
  "exemplarCity";
  "Port of Spain";
}
"Porto_Velho";
{
  "exemplarCity";
  "Porto Velho";
}
"Puerto_Rico";
{
  "exemplarCity";
  "Puerto Rico";
}
"Rainy_River";
{
  "exemplarCity";
  "Rainy River";
}
"Rankin_Inlet";
{
  "exemplarCity";
  "Rankin Inlet";
}
"Recife";
{
  "exemplarCity";
  "Recife";
}
"Regina";
{
  "exemplarCity";
  "Regina";
}
"Resolute";
{
```

```
        "exemplarCity";
        "Resolute";
    }
    "Rio_Branco";
    {
        "exemplarCity";
        "Rio Branco";
    }
    "Santa_Isabel";
    {
        "exemplarCity";
        "Santa Isabel";
    }
    "Santarem";
    {
        "exemplarCity";
        "Santarem";
    }
    "Santiago";
    {
        "exemplarCity";
        "Santiago";
    }
    "Santo_Domingo";
    {
        "exemplarCity";
        "Santo Domingo";
    }
    "Sao_Paulo";
    {
        "exemplarCity";
        "São Paulo";
    }
    "Scoresbysund";
    {
        "exemplarCity";
        "Ittoqqortoormiit";
    }
    "Sitka";
    {
        "exemplarCity";
        "Sitka";
    }
    "St_Barthelemy";
    {
        "exemplarCity";
        "Saint-Barthélemy";
    }
    "St_Johns";
    {
        "exemplarCity";
        "St. John's";
    }
    "St_Kitts";
    {
        "exemplarCity";
        "St. Kitts";
    }
}
```



```
}
"St_Lucia";
{
  "exemplarCity";
  "St. Lucia";
}
"St_Thomas";
{
  "exemplarCity";
  "St. Thomas";
}
"St_Vincent";
{
  "exemplarCity";
  "St. Vincent";
}
"Swift_Current";
{
  "exemplarCity";
  "Swift Current";
}
"Tegucigalpa";
{
  "exemplarCity";
  "Tegucigalpa";
}
"Thule";
{
  "exemplarCity";
  "Thule";
}
"Thunder_Bay";
{
  "exemplarCity";
  "Thunder Bay";
}
"Tijuana";
{
  "exemplarCity";
  "Tijuana";
}
"Toronto";
{
  "exemplarCity";
  "Toronto";
}
"Tortola";
{
  "exemplarCity";
  "Tortola";
}
"Vancouver";
{
  "exemplarCity";
  "Vancouver";
}
"Whitehorse";
```

```
{
    "exemplarCity";
    "Whitehorse";
}
"Winnipeg";
{
    "exemplarCity";
    "Winnipeg";
}
"Yakutat";
{
    "exemplarCity";
    "Yakutat";
}
"Yellowknife";
{
    "exemplarCity";
    "Yellowknife";
}
}
"Atlantic";
{
    "Azores";
    {
        "exemplarCity";
        "Azoren";
    }
    "Bermuda";
    {
        "exemplarCity";
        "Bermudas";
    }
    "Canary";
    {
        "exemplarCity";
        "Kanaren";
    }
    "Cape_Verde";
    {
        "exemplarCity";
        "Cabo Verde";
    }
    "Faeroe";
    {
        "exemplarCity";
        "Färöer";
    }
    "Madeira";
    {
        "exemplarCity";
        "Madeira";
    }
    "Reykjavik";
    {
        "exemplarCity";
        "Reykjavík";
    }
}
```

```
        "South_Georgia";
        {
            "exemplarCity";
            "Südgeorgien";
        }
        "St_Helena";
        {
            "exemplarCity";
            "St. Helena";
        }
        "Stanley";
        {
            "exemplarCity";
            "Stanley";
        }
    }
    "Europe";
    {
        "Amsterdam";
        {
            "exemplarCity";
            "Amsterdam";
        }
        "Andorra";
        {
            "exemplarCity";
            "Andorra";
        }
        "Astrakhan";
        {
            "exemplarCity";
            "Astrachan";
        }
        "Athens";
        {
            "exemplarCity";
            "Athen";
        }
        "Belgrade";
        {
            "exemplarCity";
            "Belgrad";
        }
        "Berlin";
        {
            "exemplarCity";
            "Berlin";
        }
        "Bratislava";
        {
            "exemplarCity";
            "Bratislava";
        }
        "Brussels";
        {
            "exemplarCity";
            "Brüssel";
        }
    }
}
```

```
}
"Bucharest";
{
  "exemplarCity";
  "Bukarest";
}
"Budapest";
{
  "exemplarCity";
  "Budapest";
}
"Busingen";
{
  "exemplarCity";
  "Büsingen";
}
"Chisinau";
{
  "exemplarCity";
  "Kischinau";
}
"Copenhagen";
{
  "exemplarCity";
  "Kopenhagen";
}
"Dublin";
{
  "long";
  {
    "daylight";
    "Irische Sommerzeit";
  }
  "exemplarCity";
  "Dublin";
}
"Gibraltar";
{
  "exemplarCity";
  "Gibraltar";
}
"Guernsey";
{
  "exemplarCity";
  "Guernsey";
}
"Helsinki";
{
  "exemplarCity";
  "Helsinki";
}
"Isle_of_Man";
{
  "exemplarCity";
  "Isle of Man";
}
"Istanbul";
```

```
{
    "exemplarCity";
    "Istanbul";
}
"Jersey";
{
    "exemplarCity";
    "Jersey";
}
"Kaliningrad";
{
    "exemplarCity";
    "Kaliningrad";
}
"Kiev";
{
    "exemplarCity";
    "Kiew";
}
"Kirov";
{
    "exemplarCity";
    "Kirow";
}
"Lisbon";
{
    "exemplarCity";
    "Lissabon";
}
"Ljubljana";
{
    "exemplarCity";
    "Ljubljana";
}
"London";
{
    "long";
    {
        "daylight";
        "Britische Sommerzeit";
    }
    "exemplarCity";
    "London";
}
"Luxembourg";
{
    "exemplarCity";
    "Luxemburg";
}
"Madrid";
{
    "exemplarCity";
    "Madrid";
}
"Malta";
{
    "exemplarCity";
```

```
        "Malta";
    }
    "Mariehamn";
    {
        "exemplarCity";
        "Mariehamn";
    }
    "Minsk";
    {
        "exemplarCity";
        "Minsk";
    }
    "Monaco";
    {
        "exemplarCity";
        "Monaco";
    }
    "Moscow";
    {
        "exemplarCity";
        "Moskau";
    }
    "Oslo";
    {
        "exemplarCity";
        "Oslo";
    }
    "Paris";
    {
        "exemplarCity";
        "Paris";
    }
    "Podgorica";
    {
        "exemplarCity";
        "Podgorica";
    }
    "Prague";
    {
        "exemplarCity";
        "Prag";
    }
    "Riga";
    {
        "exemplarCity";
        "Riga";
    }
    "Rome";
    {
        "exemplarCity";
        "Rom";
    }
    "Samara";
    {
        "exemplarCity";
        "Samara";
    }
}
```

```
"San_Marino";
{
  "exemplarCity";
  "San Marino";
}
"Sarajevo";
{
  "exemplarCity";
  "Sarajevo";
}
"Simferopol";
{
  "exemplarCity";
  "Simferopol";
}
"Skopje";
{
  "exemplarCity";
  "Skopje";
}
"Sofia";
{
  "exemplarCity";
  "Sofia";
}
"Stockholm";
{
  "exemplarCity";
  "Stockholm";
}
"Tallinn";
{
  "exemplarCity";
  "Tallinn";
}
"Tirane";
{
  "exemplarCity";
  "Tirana";
}
"Ulyanovsk";
{
  "exemplarCity";
  "Uljanowsk";
}
"Uzhgorod";
{
  "exemplarCity";
  "Uschgorod";
}
"Vaduz";
{
  "exemplarCity";
  "Vaduz";
}
"Vatican";
{
```

```
        "exemplarCity";
        "Vatikan";
    }
    "Vienna";
    {
        "exemplarCity";
        "Wien";
    }
    "Vilnius";
    {
        "exemplarCity";
        "Vilnius";
    }
    "Volgograd";
    {
        "exemplarCity";
        "Wolgograd";
    }
    "Warsaw";
    {
        "exemplarCity";
        "Warschau";
    }
    "Zagreb";
    {
        "exemplarCity";
        "Zagreb";
    }
    "Zaporozhye";
    {
        "exemplarCity";
        "Saporischja";
    }
    "Zurich";
    {
        "exemplarCity";
        "Zürich";
    }
}
"Africa";
{
    "Abidjan";
    {
        "exemplarCity";
        "Abidjan";
    }
    "Accra";
    {
        "exemplarCity";
        "Accra";
    }
    "Addis_Ababa";
    {
        "exemplarCity";
        "Addis Abeba";
    }
    "Algiers";
```



```
{
    "exemplarCity";
    "Algier";
}
"Asmera";
{
    "exemplarCity";
    "Asmara";
}
"Bamako";
{
    "exemplarCity";
    "Bamako";
}
"Bangui";
{
    "exemplarCity";
    "Bangui";
}
"Banjul";
{
    "exemplarCity";
    "Banjul";
}
"Bissau";
{
    "exemplarCity";
    "Bissau";
}
"Blantyre";
{
    "exemplarCity";
    "Blantyre";
}
"Brazzaville";
{
    "exemplarCity";
    "Brazzaville";
}
"Bujumbura";
{
    "exemplarCity";
    "Bujumbura";
}
"Cairo";
{
    "exemplarCity";
    "Kairo";
}
"Casablanca";
{
    "exemplarCity";
    "Casablanca";
}
"Ceuta";
{
    "exemplarCity";
```

```
        "Ceuta";
    }
    "Conakry";
    {
        "exemplarCity";
        "Conakry";
    }
    "Dakar";
    {
        "exemplarCity";
        "Dakar";
    }
    "Dar_es_Salaam";
    {
        "exemplarCity";
        "Daressalam";
    }
    "Djibouti";
    {
        "exemplarCity";
        "Dschibuti";
    }
    "Douala";
    {
        "exemplarCity";
        "Douala";
    }
    "El_Aaiun";
    {
        "exemplarCity";
        "El Aaiún";
    }
    "Freetown";
    {
        "exemplarCity";
        "Freetown";
    }
    "Gaborone";
    {
        "exemplarCity";
        "Gaborone";
    }
    "Harare";
    {
        "exemplarCity";
        "Harare";
    }
    "Johannesburg";
    {
        "exemplarCity";
        "Johannesburg";
    }
    "Juba";
    {
        "exemplarCity";
        "Juba";
    }
}
```

```
"Kampala";
{
  "exemplarCity";
  "Kampala";
}
"Khartoum";
{
  "exemplarCity";
  "Khartoum";
}
"Kigali";
{
  "exemplarCity";
  "Kigali";
}
"Kinshasa";
{
  "exemplarCity";
  "Kinshasa";
}
"Lagos";
{
  "exemplarCity";
  "Lagos";
}
"Libreville";
{
  "exemplarCity";
  "Libreville";
}
"Lome";
{
  "exemplarCity";
  "Lomé";
}
"Luanda";
{
  "exemplarCity";
  "Luanda";
}
"Lubumbashi";
{
  "exemplarCity";
  "Lubumbashi";
}
"Lusaka";
{
  "exemplarCity";
  "Lusaka";
}
"Malabo";
{
  "exemplarCity";
  "Malabo";
}
"Maputo";
{
```

```
        "exemplarCity";
        "Maputo";
    }
    "Maseru";
    {
        "exemplarCity";
        "Maseru";
    }
    "Mbabane";
    {
        "exemplarCity";
        "Mbabane";
    }
    "Mogadishu";
    {
        "exemplarCity";
        "Mogadischu";
    }
    "Monrovia";
    {
        "exemplarCity";
        "Monrovia";
    }
    "Nairobi";
    {
        "exemplarCity";
        "Nairobi";
    }
    "Ndjamena";
    {
        "exemplarCity";
        "N' Djamena";
    }
    "Niamey";
    {
        "exemplarCity";
        "Niamey";
    }
    "Nouakchott";
    {
        "exemplarCity";
        "Nouakchott";
    }
    "Ouagadougou";
    {
        "exemplarCity";
        "Ouagadougou";
    }
    "Porto-Novo";
    {
        "exemplarCity";
        "Porto Novo";
    }
    "Sao_Tome";
    {
        "exemplarCity";
        "São Tomé";
    }
```

```
    }  
    "Tripoli";  
    {  
        "exemplarCity";  
        "Tripolis";  
    }  
    "Tunis";  
    {  
        "exemplarCity";  
        "Tunis";  
    }  
    "Windhoek";  
    {  
        "exemplarCity";  
        "Windhoek";  
    }  
}  
"Asia";  
{  
    "Aden";  
    {  
        "exemplarCity";  
        "Aden";  
    }  
    "Almaty";  
    {  
        "exemplarCity";  
        "Almaty";  
    }  
    "Amman";  
    {  
        "exemplarCity";  
        "Amman";  
    }  
    "Anadyr";  
    {  
        "exemplarCity";  
        "Anadyr";  
    }  
    "Aqtau";  
    {  
        "exemplarCity";  
        "Aqtau";  
    }  
    "Aqtobe";  
    {  
        "exemplarCity";  
        "Aktobe";  
    }  
    "Ashgabat";  
    {  
        "exemplarCity";  
        "Aşgabat";  
    }  
    "Baghdad";  
    {  
        "exemplarCity";  
    }  
}
```

```
        "Bagdad";
    }
    "Bahrain";
    {
        "exemplarCity";
        "Bahrain";
    }
    "Baku";
    {
        "exemplarCity";
        "Baku";
    }
    "Bangkok";
    {
        "exemplarCity";
        "Bangkok";
    }
    "Barnaul";
    {
        "exemplarCity";
        "Barnaul";
    }
    "Beirut";
    {
        "exemplarCity";
        "Beirut";
    }
    "Bishkek";
    {
        "exemplarCity";
        "Bischkek";
    }
    "Brunei";
    {
        "exemplarCity";
        "Brunei";
    }
    "Calcutta";
    {
        "exemplarCity";
        "Kalkutta";
    }
    "Chita";
    {
        "exemplarCity";
        "Tschita";
    }
    "Choibalsan";
    {
        "exemplarCity";
        "Tschoibalsan";
    }
    "Colombo";
    {
        "exemplarCity";
        "Colombo";
    }
}
```

```
"Damascus";
{
  "exemplarCity";
  "Damaskus";
}
"Dhaka";
{
  "exemplarCity";
  "Dhaka";
}
"Dili";
{
  "exemplarCity";
  "Dili";
}
"Dubai";
{
  "exemplarCity";
  "Dubai";
}
"Dushanbe";
{
  "exemplarCity";
  "Duschanbe";
}
"Gaza";
{
  "exemplarCity";
  "Gaza";
}
"Hebron";
{
  "exemplarCity";
  "Hebron";
}
"Hong_Kong";
{
  "exemplarCity";
  "Hongkong";
}
"Hovd";
{
  "exemplarCity";
  "Chowd";
}
"Irkutsk";
{
  "exemplarCity";
  "Irkutsk";
}
"Jakarta";
{
  "exemplarCity";
  "Jakarta";
}
"Jayapura";
{
```

```
        "exemplarCity";
        "Jayapura";
    }
    "Jerusalem";
    {
        "exemplarCity";
        "Jerusalem";
    }
    "Kabul";
    {
        "exemplarCity";
        "Kabul";
    }
    "Kamchatka";
    {
        "exemplarCity";
        "Kamtschatka";
    }
    "Karachi";
    {
        "exemplarCity";
        "Karatschi";
    }
    "Katmandu";
    {
        "exemplarCity";
        "Kathmandu";
    }
    "Khandyga";
    {
        "exemplarCity";
        "Chandyga";
    }
    "Krasnoyarsk";
    {
        "exemplarCity";
        "Krasnojarsk";
    }
    "Kuala_Lumpur";
    {
        "exemplarCity";
        "Kuala Lumpur";
    }
    "Kuching";
    {
        "exemplarCity";
        "Kuching";
    }
    "Kuwait";
    {
        "exemplarCity";
        "Kuwait";
    }
    "Macau";
    {
        "exemplarCity";
        "Macao";
    }
```



```
}
"Magadan";
{
  "exemplarCity";
  "Magadan";
}
"Makassar";
{
  "exemplarCity";
  "Makassar";
}
"Manila";
{
  "exemplarCity";
  "Manila";
}
"Muscat";
{
  "exemplarCity";
  "Maskat";
}
"Nicosia";
{
  "exemplarCity";
  "Nikosia";
}
"Novokuznetsk";
{
  "exemplarCity";
  "Nowokuznetsk";
}
"Novosibirsk";
{
  "exemplarCity";
  "Nowosibirsk";
}
"Omsk";
{
  "exemplarCity";
  "Omsk";
}
"Oral";
{
  "exemplarCity";
  "Oral";
}
"Phnom_Penh";
{
  "exemplarCity";
  "Phnom Penh";
}
"Pontianak";
{
  "exemplarCity";
  "Pontianak";
}
"Pyongyang";
```

```
{
    "exemplarCity";
    "Pjöngjang";
}
"Qatar";
{
    "exemplarCity";
    "Katar";
}
"Qyzylorda";
{
    "exemplarCity";
    "Qysylorda";
}
"Rangoon";
{
    "exemplarCity";
    "Rangun";
}
"Riyadh";
{
    "exemplarCity";
    "Riad";
}
"Saigon";
{
    "exemplarCity";
    "Ho-Chi-Minh-Stadt";
}
"Sakhalin";
{
    "exemplarCity";
    "Sachalin";
}
"Samarkand";
{
    "exemplarCity";
    "Samarkand";
}
"Seoul";
{
    "exemplarCity";
    "Seoul";
}
"Shanghai";
{
    "exemplarCity";
    "Shanghai";
}
"Singapore";
{
    "exemplarCity";
    "Singapur";
}
"Srednekolymsk";
{
    "exemplarCity";
```

```
        "Srednekolymysk";
    }
    "Taipei";
    {
        "exemplarCity";
        "Taipeh";
    }
    "Tashkent";
    {
        "exemplarCity";
        "Taschkent";
    }
    "Tbilisi";
    {
        "exemplarCity";
        "Tiflis";
    }
    "Tehran";
    {
        "exemplarCity";
        "Teheran";
    }
    "Thimphu";
    {
        "exemplarCity";
        "Thimphu";
    }
    "Tokyo";
    {
        "exemplarCity";
        "Tokio";
    }
    "Tomsk";
    {
        "exemplarCity";
        "Tomsk";
    }
    "Ulaanbaatar";
    {
        "exemplarCity";
        "Ulaanbaatar";
    }
    "Urumqi";
    {
        "exemplarCity";
        "Ürümqi";
    }
    "Ust-Nera";
    {
        "exemplarCity";
        "Ust-Nera";
    }
    "Vientiane";
    {
        "exemplarCity";
        "Vientiane";
    }
}
```

```
"Vladivostok";
{
  "exemplarCity";
  "Wladiwostok";
}
"Yakutsk";
{
  "exemplarCity";
  "Jakutsk";
}
"Yekaterinburg";
{
  "exemplarCity";
  "Jekaterinburg";
}
"Yerevan";
{
  "exemplarCity";
  "Eriwan";
}
}
"Indian";
{
  "Antananarivo";
  {
    "exemplarCity";
    "Antananarivo";
  }
  "Chagos";
  {
    "exemplarCity";
    "Chagos";
  }
  "Christmas";
  {
    "exemplarCity";
    "Weihnachtsinsel";
  }
  "Cocos";
  {
    "exemplarCity";
    "Cocos";
  }
  "Comoro";
  {
    "exemplarCity";
    "Komoren";
  }
  "Kerguelen";
  {
    "exemplarCity";
    "Kerguelen";
  }
  "Mahe";
  {
    "exemplarCity";
    "Mahe";
  }
}
```

```
}
"Maldives";
{
  "exemplarCity";
  "Malediven";
}
"Mauritius";
{
  "exemplarCity";
  "Mauritius";
}
"Mayotte";
{
  "exemplarCity";
  "Mayotte";
}
"Reunion";
{
  "exemplarCity";
  "Réunion";
}
}
"Australia";
{
  "Adelaide";
  {
    "exemplarCity";
    "Adelaide";
  }
  "Brisbane";
  {
    "exemplarCity";
    "Brisbane";
  }
  "Broken_Hill";
  {
    "exemplarCity";
    "Broken Hill";
  }
  "Currie";
  {
    "exemplarCity";
    "Currie";
  }
  "Darwin";
  {
    "exemplarCity";
    "Darwin";
  }
  "Eucla";
  {
    "exemplarCity";
    "Eucla";
  }
  "Hobart";
  {
    "exemplarCity";
```

```
        "Hobart";
    }
    "Lindeman";
    {
        "exemplarCity";
        "Lindeman";
    }
    "Lord_Howe";
    {
        "exemplarCity";
        "Lord Howe";
    }
    "Melbourne";
    {
        "exemplarCity";
        "Melbourne";
    }
    "Perth";
    {
        "exemplarCity";
        "Perth";
    }
    "Sydney";
    {
        "exemplarCity";
        "Sydney";
    }
}
"Pacific";
{
    "Apia";
    {
        "exemplarCity";
        "Apia";
    }
    "Auckland";
    {
        "exemplarCity";
        "Auckland";
    }
    "Bougainville";
    {
        "exemplarCity";
        "Bougainville";
    }
    "Chatham";
    {
        "exemplarCity";
        "Chatham";
    }
    "Easter";
    {
        "exemplarCity";
        "Osterinsel";
    }
    "Efate";
    {
```

```
        "exemplarCity";
        "Efate";
    }
    "Enderbury";
    {
        "exemplarCity";
        "Enderbury";
    }
    "Fakaofo";
    {
        "exemplarCity";
        "Fakaofo";
    }
    "Fiji";
    {
        "exemplarCity";
        "Fidschi";
    }
    "Funafuti";
    {
        "exemplarCity";
        "Funafuti";
    }
    "Galapagos";
    {
        "exemplarCity";
        "Galapagos";
    }
    "Gambier";
    {
        "exemplarCity";
        "Gambier";
    }
    "Guadalcanal";
    {
        "exemplarCity";
        "Guadalcanal";
    }
    "Guam";
    {
        "exemplarCity";
        "Guam";
    }
    "Honolulu";
    {
        "exemplarCity";
        "Honolulu";
    }
    "Johnston";
    {
        "exemplarCity";
        "Johnston";
    }
    "Kiritimati";
    {
        "exemplarCity";
        "Kiritimati";
    }
```

```
}
"Kosrae";
{
  "exemplarCity";
  "Kosrae";
}
"Kwajalein";
{
  "exemplarCity";
  "Kwajalein";
}
"Majuro";
{
  "exemplarCity";
  "Majuro";
}
"Marquesas";
{
  "exemplarCity";
  "Marquesas";
}
"Midway";
{
  "exemplarCity";
  "Midway";
}
"Nauru";
{
  "exemplarCity";
  "Nauru";
}
"Niue";
{
  "exemplarCity";
  "Niue";
}
"Norfolk";
{
  "exemplarCity";
  "Norfolk";
}
"Noumea";
{
  "exemplarCity";
  "Noumea";
}
"Pago_Pago";
{
  "exemplarCity";
  "Pago Pago";
}
"Palau";
{
  "exemplarCity";
  "Palau";
}
"Pitcairn";
```



```
{
    "exemplarCity";
    "Pitcairn";
}
"Ponape";
{
    "exemplarCity";
    "Pohnpei";
}
"Port_Moresby";
{
    "exemplarCity";
    "Port Moresby";
}
"Rarotonga";
{
    "exemplarCity";
    "Rarotonga";
}
"Saipan";
{
    "exemplarCity";
    "Saipan";
}
"Tahiti";
{
    "exemplarCity";
    "Tahiti";
}
"Tarawa";
{
    "exemplarCity";
    "Tarawa";
}
"Tongatapu";
{
    "exemplarCity";
    "Tongatapu";
}
"Truk";
{
    "exemplarCity";
    "Chuuk";
}
"Wake";
{
    "exemplarCity";
    "Wake";
}
"Wallis";
{
    "exemplarCity";
    "Wallis";
}
}
"Arctic";
{
```

```
        "Longyearbyen";
        {
            "exemplarCity";
            "Longyearbyen";
        }
    }
    "Antarctica";
    {
        "Casey";
        {
            "exemplarCity";
            "Casey";
        }
        "Davis";
        {
            "exemplarCity";
            "Davis";
        }
        "DumontDUrville";
        {
            "exemplarCity";
            "Dumont d'Urville";
        }
        "Macquarie";
        {
            "exemplarCity";
            "Macquarie";
        }
        "Mawson";
        {
            "exemplarCity";
            "Mawson";
        }
        "McMurdo";
        {
            "exemplarCity";
            "McMurdo";
        }
        "Palmer";
        {
            "exemplarCity";
            "Palmer";
        }
        "Rothera";
        {
            "exemplarCity";
            "Rothera";
        }
        "Syowa";
        {
            "exemplarCity";
            "Syowa";
        }
        "Troll";
        {
            "exemplarCity";
            "Troll";
        }
    }
```

```
    }  
    "Vostok";  
    {  
      "exemplarCity";  
      "Wostok";  
    }  
  }  
  "Etc";  
  {  
    "GMT";  
    {  
      "exemplarCity";  
      "GMT";  
    }  
    "GMT1";  
    {  
      "exemplarCity";  
      "GMT+1";  
    }  
    "GMT10";  
    {  
      "exemplarCity";  
      "GMT+10";  
    }  
    "GMT11";  
    {  
      "exemplarCity";  
      "GMT+11";  
    }  
    "GMT12";  
    {  
      "exemplarCity";  
      "GMT+12";  
    }  
    "GMT2";  
    {  
      "exemplarCity";  
      "GMT+2";  
    }  
    "GMT3";  
    {  
      "exemplarCity";  
      "GMT+3";  
    }  
    "GMT4";  
    {  
      "exemplarCity";  
      "GMT+4";  
    }  
    "GMT5";  
    {  
      "exemplarCity";  
      "GMT+5";  
    }  
    "GMT6";  
    {  
      "exemplarCity";
```

```
        "GMT+6";
    }
    "GMT7";
    {
        "exemplarCity";
        "GMT+7";
    }
    "GMT8";
    {
        "exemplarCity";
        "GMT+8";
    }
    "GMT9";
    {
        "exemplarCity";
        "GMT+9";
    }
    "GMT-1";
    {
        "exemplarCity";
        "GMT-1";
    }
    "GMT-10";
    {
        "exemplarCity";
        "GMT-10";
    }
    "GMT-11";
    {
        "exemplarCity";
        "GMT-11";
    }
    "GMT-12";
    {
        "exemplarCity";
        "GMT-12";
    }
    "GMT-13";
    {
        "exemplarCity";
        "GMT-13";
    }
    "GMT-14";
    {
        "exemplarCity";
        "GMT-14";
    }
    "GMT-2";
    {
        "exemplarCity";
        "GMT-2";
    }
    "GMT-3";
    {
        "exemplarCity";
        "GMT-3";
    }
}
```

```

        "GMT-4";
        {
            "exemplarCity";
            "GMT-4";
        }
        "GMT-5";
        {
            "exemplarCity";
            "GMT-5";
        }
        "GMT-6";
        {
            "exemplarCity";
            "GMT-6";
        }
        "GMT-7";
        {
            "exemplarCity";
            "GMT-7";
        }
        "GMT-8";
        {
            "exemplarCity";
            "GMT-8";
        }
        "GMT-9";
        {
            "exemplarCity";
            "GMT-9";
        }
        "Unknown";
        {
            "exemplarCity";
            "Unbekannt";
        }
    }
}
"metazone";
{
    "Acre";
    {
        "long";
        {
            "generic";
            "Acre-Zeit",
            "standard";
            "Acre-Normalzeit",
            "daylight";
            "Acre-Sommerzeit";
        }
    }
}
"Afghanistan";
{
    "long";
    {
        "standard";
        "Afghanistan-Zeit";
    }
}

```

```

    }
  }
  "Africa_Central";
  {
    "long";
    {
      "standard";
      "Zentralafrikanische Zeit";
    }
  }
  "Africa_Eastern";
  {
    "long";
    {
      "standard";
      "Ostafrikanische Zeit";
    }
  }
  "Africa_Southern";
  {
    "long";
    {
      "standard";
      "Südafrikanische Zeit";
    }
  }
  "Africa_Western";
  {
    "long";
    {
      "generic";
      "Westafrikanische Zeit",
      "standard";
      "Westafrikanische Normalzeit",
      "daylight";
      "Westafrikanische Sommerzeit";
    }
  }
  "Alaska";
  {
    "long";
    {
      "generic";
      "Alaska-Zeit",
      "standard";
      "Alaska-Normalzeit",
      "daylight";
      "Alaska-Sommerzeit";
    }
  }
  "Almaty";
  {
    "long";
    {
      "generic";
      "Almaty-Zeit",
      "standard";
    }
  }

```

```
        "Almaty-Normalzeit",
        "daylight";
        "Almaty-Sommerzeit";
    }
}
"Amazon";
{
    "long";
    {
        "generic";
        "Amazonas-Zeit",
        "standard";
        "Amazonas-Normalzeit",
        "daylight";
        "Amazonas-Sommerzeit";
    }
}
"America_Central";
{
    "long";
    {
        "generic";
        "Nordamerikanische Inlandzeit",
        "standard";
        "Nordamerikanische Inland-Normalzeit",
        "daylight";
        "Nordamerikanische Inland-Sommerzeit";
    }
}
"America_Eastern";
{
    "long";
    {
        "generic";
        "Nordamerikanische Ostküstenzeit",
        "standard";
        "Nordamerikanische Ostküsten-Normalzeit",
        "daylight";
        "Nordamerikanische Ostküsten-Sommerzeit";
    }
}
"America_Mountain";
{
    "long";
    {
        "generic";
        "Rocky-Mountain-Zeit",
        "standard";
        "Rocky Mountain-Normalzeit",
        "daylight";
        "Rocky-Mountain-Sommerzeit";
    }
}
"America_Pacific";
{
    "long";
    {
```

```

        "generic";
        "Nordamerikanische Westküstenzeit",
        "standard";
        "Nordamerikanische Westküsten-Normalzeit",
        "daylight";
        "Nordamerikanische Westküsten-Sommerzeit";
    }
}
"Anadyr";
{
    "long";
    {
        "generic";
        "Anadyr Zeit",
        "standard";
        "Anadyr Normalzeit",
        "daylight";
        "Anadyr Sommerzeit";
    }
}
"Apia";
{
    "long";
    {
        "generic";
        "Apia-Zeit",
        "standard";
        "Apia-Normalzeit",
        "daylight";
        "Apia-Sommerzeit";
    }
}
"Aqtau";
{
    "long";
    {
        "generic";
        "Aqtau-Zeit",
        "standard";
        "Aqtau-Normalzeit",
        "daylight";
        "Aqtau-Sommerzeit";
    }
}
"Aqtobe";
{
    "long";
    {
        "generic";
        "Aqtöbe-Zeit",
        "standard";
        "Aqtöbe-Normalzeit",
        "daylight";
        "Aqtöbe-Sommerzeit";
    }
}
"Arabian";

```



```
{
  "long";
  {
    "generic";
    "Arabische Zeit",
    "standard";
    "Arabische Normalzeit",
    "daylight";
    "Arabische Sommerzeit";
  }
}
"Argentina";
{
  "long";
  {
    "generic";
    "Argentinische Zeit",
    "standard";
    "Argentinische Normalzeit",
    "daylight";
    "Argentinische Sommerzeit";
  }
}
"Argentina_Western";
{
  "long";
  {
    "generic";
    "Westargentinische Zeit",
    "standard";
    "Westargentinische Normalzeit",
    "daylight";
    "Westargentinische Sommerzeit";
  }
}
"Armenia";
{
  "long";
  {
    "generic";
    "Armenische Zeit",
    "standard";
    "Armenische Normalzeit",
    "daylight";
    "Armenische Sommerzeit";
  }
}
"Atlantic";
{
  "long";
  {
    "generic";
    "Atlantik-Zeit",
    "standard";
    "Atlantik-Normalzeit",
    "daylight";
    "Atlantik-Sommerzeit";
  }
}
```

```

    }
  }
  "Australia_Central";
  {
    "long";
    {
      "generic";
      "Zentralaustralische Zeit",
      "standard";
      "Zentralaustralische Normalzeit",
      "daylight";
      "Zentralaustralische Sommerzeit";
    }
  }
  "Australia_CentralWestern";
  {
    "long";
    {
      "generic";
      "Zentral-/Westaustralische Zeit",
      "standard";
      "Zentral-/Westaustralische Normalzeit",
      "daylight";
      "Zentral-/Westaustralische Sommerzeit";
    }
  }
  "Australia_Eastern";
  {
    "long";
    {
      "generic";
      "Ostaustralische Zeit",
      "standard";
      "Ostaustralische Normalzeit",
      "daylight";
      "Ostaustralische Sommerzeit";
    }
  }
  "Australia_Western";
  {
    "long";
    {
      "generic";
      "Westaustralische Zeit",
      "standard";
      "Westaustralische Normalzeit",
      "daylight";
      "Westaustralische Sommerzeit";
    }
  }
  "Azerbaijan";
  {
    "long";
    {
      "generic";
      "Aserbaidshanische Zeit",
      "standard";
    }
  }

```

```

        "Aserbeidschanische Normalzeit",
        "daylight";
        "Aserbaidtschanische Sommerzeit";
    }
}
"Azores";
{
    "long";
    {
        "generic";
        "Azoren-Zeit",
        "standard";
        "Azoren-Normalzeit",
        "daylight";
        "Azoren-Sommerzeit";
    }
}
"Bangladesh";
{
    "long";
    {
        "generic";
        "Bangladesch-Zeit",
        "standard";
        "Bangladesch-Normalzeit",
        "daylight";
        "Bangladesch-Sommerzeit";
    }
}
"Bhutan";
{
    "long";
    {
        "standard";
        "Bhutan-Zeit";
    }
}
"Bolivia";
{
    "long";
    {
        "standard";
        "Bolivianische Zeit";
    }
}
"Brasilia";
{
    "long";
    {
        "generic";
        "Brasília-Zeit",
        "standard";
        "Brasília-Normalzeit",
        "daylight";
        "Brasília-Sommerzeit";
    }
}
}

```

```
"Brunei";
{
  "long";
  {
    "standard";
    "Brunei-Zeit";
  }
}
"Cape_Verde";
{
  "long";
  {
    "generic";
    "Cabo-Verde-Zeit",
    "standard";
    "Cabo-Verde-Normalzeit",
    "daylight";
    "Cabo-Verde-Sommerzeit";
  }
}
"Casey";
{
  "long";
  {
    "standard";
    "Casey-Zeit";
  }
}
"Chamorro";
{
  "long";
  {
    "standard";
    "Chamorro-Zeit";
  }
}
"Chatham";
{
  "long";
  {
    "generic";
    "Chatham-Zeit",
    "standard";
    "Chatham-Normalzeit",
    "daylight";
    "Chatham-Sommerzeit";
  }
}
"Chile";
{
  "long";
  {
    "generic";
    "Chilenische Zeit",
    "standard";
    "Chilenische Normalzeit",
    "daylight";
  }
}
```

```

        "Chilenische Sommerzeit";
    }
}
"China";
{
    "long";
    {
        "generic";
        "Chinesische Zeit",
        "standard";
        "Chinesische Normalzeit",
        "daylight";
        "Chinesische Sommerzeit";
    }
}
"Choibalsan";
{
    "long";
    {
        "generic";
        "Tschoibalsan-Zeit",
        "standard";
        "Tschoibalsan-Normalzeit",
        "daylight";
        "Tschoibalsan-Sommerzeit";
    }
}
"Christmas";
{
    "long";
    {
        "standard";
        "Weihnachtsinsel-Zeit";
    }
}
"Cocos";
{
    "long";
    {
        "standard";
        "Kokosinseln-Zeit";
    }
}
"Colombia";
{
    "long";
    {
        "generic";
        "Kolumbianische Zeit",
        "standard";
        "Kolumbianische Normalzeit",
        "daylight";
        "Kolumbianische Sommerzeit";
    }
}
"Cook";
{

```

```
        "long";
        {
            "generic";
            "Cookinseln-Zeit",
            "standard";
            "Cookinseln-Normalzeit",
            "daylight";
            "Cookinseln-Sommerzeit";
        }
    }
    "Cuba";
    {
        "long";
        {
            "generic";
            "Kubanische Zeit",
            "standard";
            "Kubanische Normalzeit",
            "daylight";
            "Kubanische Sommerzeit";
        }
    }
    "Davis";
    {
        "long";
        {
            "standard";
            "Davis-Zeit";
        }
    }
    "DumontDUrville";
    {
        "long";
        {
            "standard";
            "Dumont-d'Urville-Zeit";
        }
    }
    "East_Timor";
    {
        "long";
        {
            "standard";
            "Osttimor-Zeit";
        }
    }
    "Easter";
    {
        "long";
        {
            "generic";
            "Osterinsel-Zeit",
            "standard";
            "Osterinsel-Normalzeit",
            "daylight";
            "Osterinsel-Sommerzeit";
        }
    }
```

```
}
"Ecuador";
{
  "long";
  {
    "standard";
    "Ecuadorianische Zeit";
  }
}
"Europe_Central";
{
  "long";
  {
    "generic";
    "Mittleuropäische Zeit",
    "standard";
    "Mittleuropäische Normalzeit",
    "daylight";
    "Mittleuropäische Sommerzeit";
  }
  "short";
  {
    "generic";
    "MEZ",
    "standard";
    "MEZ",
    "daylight";
    "MESZ";
  }
}
"Europe_Eastern";
{
  "long";
  {
    "generic";
    "Osteuropäische Zeit",
    "standard";
    "Osteuropäische Normalzeit",
    "daylight";
    "Osteuropäische Sommerzeit";
  }
  "short";
  {
    "generic";
    "OEZ",
    "standard";
    "OEZ",
    "daylight";
    "OESZ";
  }
}
"Europe_Further_Eastern";
{
  "long";
  {
    "standard";
    "Kaliningrader Zeit";
```

```

    }
  }
  "Europe_Western";
  {
    "long";
    {
      "generic";
      "Westeuropäische Zeit",
      "standard";
      "Westeuropäische Normalzeit",
      "daylight";
      "Westeuropäische Sommerzeit";
    }
    "short";
    {
      "generic";
      "WEZ",
      "standard";
      "WEZ",
      "daylight";
      "WESZ";
    }
  }
}
"Falkland";
{
  "long";
  {
    "generic";
    "Falklandinseln-Zeit",
    "standard";
    "Falklandinseln-Normalzeit",
    "daylight";
    "Falklandinseln-Sommerzeit";
  }
}
"Fiji";
{
  "long";
  {
    "generic";
    "Fidschi-Zeit",
    "standard";
    "Fidschi-Normalzeit",
    "daylight";
    "Fidschi-Sommerzeit";
  }
}
"French_Guiana";
{
  "long";
  {
    "standard";
    "Französisch-Guayana-Zeit";
  }
}
"French_Southern";
{

```



```

        "long";
        {
            "standard";
            "Französische Süd- und Antarktisgebiete-
Zeit";
        }
    }
    "Galapagos";
    {
        "long";
        {
            "standard";
            "Galapagos-Zeit";
        }
    }
    "Gambier";
    {
        "long";
        {
            "standard";
            "Gambier-Zeit";
        }
    }
    "Georgia";
    {
        "long";
        {
            "generic";
            "Georgische Zeit",
            "standard";
            "Georgische Normalzeit",
            "daylight";
            "Georgische Sommerzeit";
        }
    }
    "Gilbert_Islands";
    {
        "long";
        {
            "standard";
            "Gilbert-Inseln-Zeit";
        }
    }
    "GMT";
    {
        "long";
        {
            "standard";
            "Mittlere Greenwich-Zeit";
        }
    }
    "Greenland_Eastern";
    {
        "long";
        {
            "generic";
            "Ostgrönland-Zeit",

```

```

        "standard";
        "Ostgrönland-Normalzeit",
        "daylight";
        "Ostgrönland-Sommerzeit";
    }
}
"Greenland_Western";
{
    "long";
    {
        "generic";
        "Westgrönland-Zeit",
        "standard";
        "Westgrönland-Normalzeit",
        "daylight";
        "Westgrönland-Sommerzeit";
    }
}
"Guam";
{
    "long";
    {
        "standard";
        "Guam-Zeit";
    }
}
"Gulf";
{
    "long";
    {
        "standard";
        "Golf-Zeit";
    }
}
"Guyana";
{
    "long";
    {
        "standard";
        "Guyana-Zeit";
    }
}
"Hawaii_Aleutian";
{
    "long";
    {
        "generic";
        "Hawaii-Aleuten-Zeit",
        "standard";
        "Hawaii-Aleuten-Normalzeit",
        "daylight";
        "Hawaii-Aleuten-Sommerzeit";
    }
}
"Hong_Kong";
{
    "long";

```

```

        {
            "generic";
            "Hongkong-Zeit",
            "standard";
            "Hongkong-Normalzeit",
            "daylight";
            "Hongkong-Sommerzeit";
        }
    }
    "Hovd";
    {
        "long";
        {
            "generic";
            "Chowd-Zeit",
            "standard";
            "Chowd-Normalzeit",
            "daylight";
            "Chowd-Sommerzeit";
        }
    }
    "India";
    {
        "long";
        {
            "standard";
            "Indische Zeit";
        }
    }
    "Indian_Ocean";
    {
        "long";
        {
            "standard";
            "Indischer Ozean-Zeit";
        }
    }
    "Indochina";
    {
        "long";
        {
            "standard";
            "Indochina-Zeit";
        }
    }
    "Indonesia_Central";
    {
        "long";
        {
            "standard";
            "Zentralindonesische Zeit";
        }
    }
    "Indonesia_Eastern";
    {
        "long";
        {

```

```

        "standard";
        "Ostindonesische Zeit";
    }
}
"Indonesia_Western";
{
    "long";
    {
        "standard";
        "Westindonesische Zeit";
    }
}
"Iran";
{
    "long";
    {
        "generic";
        "Iranische Zeit",
        "standard";
        "Iranische Normalzeit",
        "daylight";
        "Iranische Sommerzeit";
    }
}
"Irkutsk";
{
    "long";
    {
        "generic";
        "Irkutsk-Zeit",
        "standard";
        "Irkutsk-Normalzeit",
        "daylight";
        "Irkutsk-Sommerzeit";
    }
}
"Israel";
{
    "long";
    {
        "generic";
        "Israelische Zeit",
        "standard";
        "Israelische Normalzeit",
        "daylight";
        "Israelische Sommerzeit";
    }
}
"Japan";
{
    "long";
    {
        "generic";
        "Japanische Zeit",
        "standard";
        "Japanische Normalzeit",
        "daylight";
    }
}

```

```

        "Japanische Sommerzeit";
    }
}
"Kamchatka";
{
    "long";
    {
        "generic";
        "Kamtschatka-Zeit",
        "standard";
        "Kamtschatka-Normalzeit",
        "daylight";
        "Kamtschatka-Sommerzeit";
    }
}
"Kazakhstan_Eastern";
{
    "long";
    {
        "standard";
        "Ostkasachische Zeit";
    }
}
"Kazakhstan_Western";
{
    "long";
    {
        "standard";
        "Westkasachische Zeit";
    }
}
"Korea";
{
    "long";
    {
        "generic";
        "Koreanische Zeit",
        "standard";
        "Koreanische Normalzeit",
        "daylight";
        "Koreanische Sommerzeit";
    }
}
"Kosrae";
{
    "long";
    {
        "standard";
        "Kosrae-Zeit";
    }
}
"Krasnoyarsk";
{
    "long";
    {
        "generic";
        "Krasnojarsk-Zeit",

```

```

        "standard";
        "Krasnojarsk-Normalzeit",
        "daylight";
        "Krasnojarsk-Sommerzeit";
    }
}
"Kyrgystan";
{
    "long";
    {
        "standard";
        "Kirgisistan-Zeit";
    }
}
"Lanka";
{
    "long";
    {
        "standard";
        "Sri-Lanka-Zeit";
    }
}
"Line_Islands";
{
    "long";
    {
        "standard";
        "Linieninseln-Zeit";
    }
}
"Lord_Howe";
{
    "long";
    {
        "generic";
        "Lord-Howe-Zeit",
        "standard";
        "Lord-Howe-Normalzeit",
        "daylight";
        "Lord-Howe-Sommerzeit";
    }
}
"Macau";
{
    "long";
    {
        "generic";
        "Macau-Zeit",
        "standard";
        "Macau-Normalzeit",
        "daylight";
        "Macau-Sommerzeit";
    }
}
"Macquarie";
{
    "long";

```

```
        {
            "standard";
            "Macquarieinsel-Zeit";
        }
    }
    "Magadan";
    {
        "long";
        {
            "generic";
            "Magadan-Zeit",
            "standard";
            "Magadan-Normalzeit",
            "daylight";
            "Magadan-Sommerzeit";
        }
    }
    "Malaysia";
    {
        "long";
        {
            "standard";
            "Malaysische Zeit";
        }
    }
    "Maldives";
    {
        "long";
        {
            "standard";
            "Malediven-Zeit";
        }
    }
    "Marquesas";
    {
        "long";
        {
            "standard";
            "Marquesas-Zeit";
        }
    }
    "Marshall_Islands";
    {
        "long";
        {
            "standard";
            "Marshallinseln-Zeit";
        }
    }
    "Mauritius";
    {
        "long";
        {
            "generic";
            "Mauritius-Zeit",
            "standard";
            "Mauritius-Normalzeit",
```

```

        "daylight";
        "Mauritius-Sommerzeit";
    }
}
"Mawson";
{
    "long";
    {
        "standard";
        "Mawson-Zeit";
    }
}
"Mexico_Northwest";
{
    "long";
    {
        "generic";
        "Mexiko Nordwestliche Zone-Zeit",
        "standard";
        "Mexiko Nordwestliche Zone-Normalzeit",
        "daylight";
        "Mexiko Nordwestliche Zone-Sommerzeit";
    }
}
"Mexico_Pacific";
{
    "long";
    {
        "generic";
        "Mexiko Pazifikzone-Zeit",
        "standard";
        "Mexiko Pazifikzone-Normalzeit",
        "daylight";
        "Mexiko Pazifikzone-Sommerzeit";
    }
}
"Mongolia";
{
    "long";
    {
        "generic";
        "Ulaanbaatar-Zeit",
        "standard";
        "Ulaanbaatar-Normalzeit",
        "daylight";
        "Ulaanbaatar-Sommerzeit";
    }
}
"Moscow";
{
    "long";
    {
        "generic";
        "Moskauer Zeit",
        "standard";
        "Moskauer Normalzeit",
        "daylight";
    }
}

```



```
        "Moskauer Sommerzeit";
    }
}
"Myanmar";
{
    "long";
    {
        "standard";
        "Myanmar-Zeit";
    }
}
"Nauru";
{
    "long";
    {
        "standard";
        "Nauru-Zeit";
    }
}
"Nepal";
{
    "long";
    {
        "standard";
        "Nepalesische Zeit";
    }
}
"New_Caledonia";
{
    "long";
    {
        "generic";
        "Neukaledonische Zeit",
        "standard";
        "Neukaledonische Normalzeit",
        "daylight";
        "Neukaledonische Sommerzeit";
    }
}
"New_Zealand";
{
    "long";
    {
        "generic";
        "Neuseeland-Zeit",
        "standard";
        "Neuseeland-Normalzeit",
        "daylight";
        "Neuseeland-Sommerzeit";
    }
}
"Newfoundland";
{
    "long";
    {
        "generic";
        "Neufundland-Zeit",
```

```

        "standard";
        "Neufundland-Normalzeit",
        "daylight";
        "Neufundland-Sommerzeit";
    }
}
"Niue";
{
    "long";
    {
        "standard";
        "Niue-Zeit";
    }
}
"Norfolk";
{
    "long";
    {
        "standard";
        "Norfolkinsel-Zeit";
    }
}
"Noronha";
{
    "long";
    {
        "generic";
        "Fernando de Noronha-Zeit",
        "standard";
        "Fernando de Noronha-Normalzeit",
        "daylight";
        "Fernando de Noronha-Sommerzeit";
    }
}
"North_Mariana";
{
    "long";
    {
        "standard";
        "Nördliche-Marianen-Zeit";
    }
}
"Novosibirsk";
{
    "long";
    {
        "generic";
        "Nowosibirsk-Zeit",
        "standard";
        "Nowosibirsk-Normalzeit",
        "daylight";
        "Nowosibirsk-Sommerzeit";
    }
}
"Omsk";
{
    "long";

```

```
        {
            "generic";
            "Omsk-Zeit",
                "standard";
            "Omsk-Normalzeit",
                "daylight";
            "Omsk-Sommerzeit";
        }
    }
    "Pakistan";
    {
        "long";
        {
            "generic";
            "Pakistanische Zeit",
                "standard";
            "Pakistanische Normalzeit",
                "daylight";
            "Pakistanische Sommerzeit";
        }
    }
    "Palau";
    {
        "long";
        {
            "standard";
            "Palau-Zeit";
        }
    }
    "Papua_New_Guinea";
    {
        "long";
        {
            "standard";
            "Papua-Neuguinea-Zeit";
        }
    }
    "Paraguay";
    {
        "long";
        {
            "generic";
            "Paraguayische Zeit",
                "standard";
            "Paraguayische Normalzeit",
                "daylight";
            "Paraguayische Sommerzeit";
        }
    }
    "Peru";
    {
        "long";
        {
            "generic";
            "Peruanische Zeit",
                "standard";
            "Peruanische Normalzeit",
```

```

        "daylight";
        "Peruanische Sommerzeit";
    }
}
"Philippines";
{
    "long";
    {
        "generic";
        "Philippinische Zeit",
        "standard";
        "Philippinische Normalzeit",
        "daylight";
        "Philippinische Sommerzeit";
    }
}
"Phoenix_Islands";
{
    "long";
    {
        "standard";
        "Phoenixinseln-Zeit";
    }
}
"Pierre_Miquelon";
{
    "long";
    {
        "generic";
        "Saint-Pierre-und-Miquelon-Zeit",
        "standard";
        "Saint-Pierre-und-Miquelon-Normalzeit",
        "daylight";
        "Saint-Pierre-und-Miquelon-Sommerzeit";
    }
}
"Pitcairn";
{
    "long";
    {
        "standard";
        "Pitcairninnseln-Zeit";
    }
}
"Ponape";
{
    "long";
    {
        "standard";
        "Ponape-Zeit";
    }
}
"Pyongyang";
{
    "long";
    {
        "standard";

```

```

        "Pjöngjang-Zeit";
    }
}
"Qyzylorda";
{
    "long";
    {
        "generic";
        "Quysylorda-Zeit",
        "standard";
        "Quysylorda-Normalzeit",
        "daylight";
        "Qysylorda-Sommerzeit";
    }
}
"Reunion";
{
    "long";
    {
        "standard";
        "Réunion-Zeit";
    }
}
"Rothera";
{
    "long";
    {
        "standard";
        "Rothera-Zeit";
    }
}
"Sakhalin";
{
    "long";
    {
        "generic";
        "Sachalin-Zeit",
        "standard";
        "Sachalin-Normalzeit",
        "daylight";
        "Sachalin-Sommerzeit";
    }
}
"Samara";
{
    "long";
    {
        "generic";
        "Samara-Zeit",
        "standard";
        "Samara-Normalzeit",
        "daylight";
        "Samara-Sommerzeit";
    }
}
"Samoa";
{

```

```

        "long";
        {
            "generic";
            "Samoa-Zeit",
            "standard";
            "Samoa-Normalzeit",
            "daylight";
            "Samoa-Sommerzeit";
        }
    }
    "Seychelles";
    {
        "long";
        {
            "standard";
            "Seychellen-Zeit";
        }
    }
    "Singapore";
    {
        "long";
        {
            "standard";
            "Singapur-Zeit";
        }
    }
    "Solomon";
    {
        "long";
        {
            "standard";
            "Salomoninseln-Zeit";
        }
    }
    "South_Georgia";
    {
        "long";
        {
            "standard";
            "Südgeorgische Zeit";
        }
    }
    "Suriname";
    {
        "long";
        {
            "standard";
            "Suriname-Zeit";
        }
    }
    "Syowa";
    {
        "long";
        {
            "standard";
            "Syowa-Zeit";
        }
    }

```

```
}
"Tahiti";
{
  "long";
  {
    "standard";
    "Tahiti-Zeit";
  }
}
"Taipei";
{
  "long";
  {
    "generic";
    "Taipeh-Zeit",
    "standard";
    "Taipeh-Normalzeit",
    "daylight";
    "Taipeh-Sommerzeit";
  }
}
"Tajikistan";
{
  "long";
  {
    "standard";
    "Tadschikistan-Zeit";
  }
}
"Tokelau";
{
  "long";
  {
    "standard";
    "Tokelau-Zeit";
  }
}
"Tonga";
{
  "long";
  {
    "generic";
    "Tonganische Zeit",
    "standard";
    "Tonganische Normalzeit",
    "daylight";
    "Tonganische Sommerzeit";
  }
}
"Truk";
{
  "long";
  {
    "standard";
    "Chuuk-Zeit";
  }
}
```

```
"Turkmenistan";
{
  "long";
  {
    "generic";
    "Turkmenistan-Zeit",
    "standard";
    "Turkmenistan-Normalzeit",
    "daylight";
    "Turkmenistan-Sommerzeit";
  }
}
"Tuvalu";
{
  "long";
  {
    "standard";
    "Tuvalu-Zeit";
  }
}
"Uruguay";
{
  "long";
  {
    "generic";
    "Uruguayische Zeit",
    "standard";
    "Uruguayische Normalzeit",
    "daylight";
    "Uruguayische Sommerzeit";
  }
}
"Uzbekistan";
{
  "long";
  {
    "generic";
    "Usbekistan-Zeit",
    "standard";
    "Usbekistan-Normalzeit",
    "daylight";
    "Usbekistan-Sommerzeit";
  }
}
"Vanuatu";
{
  "long";
  {
    "generic";
    "Vanuatu-Zeit",
    "standard";
    "Vanuatu-Normalzeit",
    "daylight";
    "Vanuatu-Sommerzeit";
  }
}
"Venezuela";
```



```
{
  "long";
  {
    "standard";
    "Venezuela-Zeit";
  }
}
"Vladivostok";
{
  "long";
  {
    "generic";
    "Wladiwostok-Zeit",
    "standard";
    "Wladiwostok-Normalzeit",
    "daylight";
    "Wladiwostok-Sommerzeit";
  }
}
"Volgograd";
{
  "long";
  {
    "generic";
    "Wolgograd-Zeit",
    "standard";
    "Wolgograd-Normalzeit",
    "daylight";
    "Wolgograd-Sommerzeit";
  }
}
"Vostok";
{
  "long";
  {
    "standard";
    "Wostok-Zeit";
  }
}
"Wake";
{
  "long";
  {
    "standard";
    "Wake-Insel-Zeit";
  }
}
"Wallis";
{
  "long";
  {
    "standard";
    "Wallis-und-Futuna-Zeit";
  }
}
"Yakutsk";
{
```

```
        "long";
    {
        "generic";
        "Jakutsk-Zeit",
            "standard";
        "Jakutsk-Normalzeit",
            "daylight";
        "Jakutsk-Sommerzeit";
    }
}
"Yekaterinburg";
{
    "long";
    {
        "generic";
        "Jekaterinburg-Zeit",
            "standard";
        "Jekaterinburg-Normalzeit",
            "daylight";
        "Jekaterinburg-Sommerzeit";
    }
}
}
}
}
}
}
```

[Functional-component]

CA-GREGORIAN.JSON

```
{  
    "main": {  
        "de": {  
            "identity": {  
                "version": {  
                    "_number": "$Revision: 12879 $",  
                    "_cldrVersion": "30.0.3"  
                },  
                "language": "de"  
            },  
            "dates": {  
                "calendars": {  
                    "gregorian": {  
                        "months": {  
                            "format": {  
                                "abbreviated": {  
                                    "1": "Jan.",  
                                    "2": "Feb.",  
                                    "3": "März",  
                                    "4": "Apr.",  
                                    "5": "Mai",  
                                    "6": "Juni",  
                                    "7": "Juli",
```

```
        "8": "Aug.",
        "9": "Sep.",
        "10": "Okt.",
        "11": "Nov.",
        "12": "Dez."
    },
    "narrow": {
        "1": "J",
        "2": "F",
        "3": "M",
        "4": "A",
        "5": "M",
        "6": "J",
        "7": "J",
        "8": "A",
        "9": "S",
        "10": "O",
        "11": "N",
        "12": "D"
    },
    "wide": {
        "1": "Januar",
        "2": "Februar",
        "3": "März",
        "4": "April",
        "5": "Mai",
        "6": "Juni",
        "7": "Juli",
        "8": "August",
        "9": "September",
        "10": "Oktober",
        "11": "November",
        "12": "Dezember"
    }
},
"stand-alone": {
    "abbreviated": {
        "1": "Jan",
        "2": "Feb",
        "3": "Mär",
        "4": "Apr",
        "5": "Mai",
        "6": "Jun",
        "7": "Jul",
        "8": "Aug",
        "9": "Sep",
        "10": "Okt",
        "11": "Nov",
        "12": "Dez"
    },
    "narrow": {
        "1": "J",
        "2": "F",
        "3": "M",
        "4": "A",
        "5": "M",
        "6": "J",
```

```
        "7": "J",
        "8": "A",
        "9": "S",
        "10": "O",
        "11": "N",
        "12": "D"
    },
    "wide": {
        "1": "Januar",
        "2": "Februar",
        "3": "März",
        "4": "April",
        "5": "Mai",
        "6": "Juni",
        "7": "Juli",
        "8": "August",
        "9": "September",
        "10": "Oktober",
        "11": "November",
        "12": "Dezember"
    }
}
},
"days": {
    "format": {
        "abbreviated": {
            "sun": "So.",
            "mon": "Mo.",
            "tue": "Di.",
            "wed": "Mi.",
            "thu": "Do.",
            "fri": "Fr.",
            "sat": "Sa."
        },
        "narrow": {
            "sun": "S",
            "mon": "M",
            "tue": "D",
            "wed": "M",
            "thu": "D",
            "fri": "F",
            "sat": "S"
        },
        "short": {
            "sun": "So.",
            "mon": "Mo.",
            "tue": "Di.",
            "wed": "Mi.",
            "thu": "Do.",
            "fri": "Fr.",
            "sat": "Sa."
        },
        "wide": {
            "sun": "Sonntag",
            "mon": "Montag",
            "tue": "Dienstag",
            "wed": "Mittwoch",
```

```
        "thu": "Donnerstag",
        "fri": "Freitag",
        "sat": "Samstag"
      }
    },
    "stand-alone": {
      "abbreviated": {
        "sun": "So",
        "mon": "Mo",
        "tue": "Di",
        "wed": "Mi",
        "thu": "Do",
        "fri": "Fr",
        "sat": "Sa"
      },
      "narrow": {
        "sun": "S",
        "mon": "M",
        "tue": "D",
        "wed": "M",
        "thu": "D",
        "fri": "F",
        "sat": "S"
      },
      "short": {
        "sun": "So.",
        "mon": "Mo.",
        "tue": "Di.",
        "wed": "Mi.",
        "thu": "Do.",
        "fri": "Fr.",
        "sat": "Sa."
      },
      "wide": {
        "sun": "Sonntag",
        "mon": "Montag",
        "tue": "Dienstag",
        "wed": "Mittwoch",
        "thu": "Donnerstag",
        "fri": "Freitag",
        "sat": "Samstag"
      }
    }
  },
  "quarters": {
    "format": {
      "abbreviated": {
        "1": "Q1",
        "2": "Q2",
        "3": "Q3",
        "4": "Q4"
      },
      "narrow": {
        "1": "1",
        "2": "2",
        "3": "3",
        "4": "4"
      }
    }
  }
}
```

```

    },
    "wide": {
      "1": "1. Quartal",
      "2": "2. Quartal",
      "3": "3. Quartal",
      "4": "4. Quartal"
    }
  },
  "stand-alone": {
    "abbreviated": {
      "1": "Q1",
      "2": "Q2",
      "3": "Q3",
      "4": "Q4"
    },
    "narrow": {
      "1": "1",
      "2": "2",
      "3": "3",
      "4": "4"
    }
  },
  "wide": {
    "1": "1. Quartal",
    "2": "2. Quartal",
    "3": "3. Quartal",
    "4": "4. Quartal"
  }
}
},
"dayPeriods": {
  "format": {
    "abbreviated": {
      "midnight": "Mitternacht",
      "am": "vorm.",
      "pm": "nachm.",
      "morning1": "morgens",
      "morning2": "vormittags",
      "afternoon1": "mittags",
      "afternoon2": "nachmittags",
      "evening1": "abends",
      "night1": "nachts"
    },
    "narrow": {
      "midnight": "Mitternacht",
      "am": "vm.",
      "pm": "nm.",
      "morning1": "morgens",
      "morning2": "vormittags",
      "afternoon1": "mittags",
      "afternoon2": "nachmittags",
      "evening1": "abends",
      "night1": "nachts"
    },
    "wide": {
      "midnight": "Mitternacht",
      "am": "vorm.",
      "pm": "nachm.",

```

```

        "morning1": "morgens",
        "morning2": "vormittags",
        "afternoon1": "mittags",
        "afternoon2": "nachmittags",
        "evening1": "abends",
        "night1": "nachts"
    }
},
"stand-alone": {
    "abbreviated": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "Morgen",
        "morning2": "Vormittag",
        "afternoon1": "Mittag",
        "afternoon2": "Nachmittag",
        "evening1": "Abend",
        "night1": "Nacht"
    },
    "narrow": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "Morgen",
        "morning2": "Vormittag",
        "afternoon1": "Mittag",
        "afternoon2": "Nachmittag",
        "evening1": "Abend",
        "night1": "Nacht"
    },
    "wide": {
        "midnight": "Mitternacht",
        "am": "vorm.",
        "pm": "nachm.",
        "morning1": "Morgen",
        "morning2": "Vormittag",
        "afternoon1": "Mittag",
        "afternoon2": "Nachmittag",
        "evening1": "Abend",
        "night1": "Nacht"
    }
}
},
"eras": {
    "eraNames": {
        "0": "v. Chr.",
        "0-alt-variant": "vor unserer Zeitrechnung",
        "1": "n. Chr.",
        "1-alt-variant": "unserer Zeitrechnung"
    },
    "eraAbbr": {
        "0": "v. Chr.",
        "0-alt-variant": "v. u. Z.",
        "1": "n. Chr.",
        "1-alt-variant": "u. Z."
    }
},

```

```

    "eraNarrow": {
      "0": "v. Chr.",
      "0-alt-variant": "v. u. Z.",
      "1": "n. Chr.",
      "1-alt-variant": "u. Z."
    }
  },
  "dateFormats": {
    "full": "EEEE, d. MMMM y",
    "long": "d. MMMM y",
    "medium": "dd.MM.y",
    "short": "dd.MM.yy"
  },
  "timeFormats": {
    "full": "HH:mm:ss zzzz",
    "long": "HH:mm:ss z",
    "medium": "HH:mm:ss",
    "short": "HH:mm"
  },
  "dateTimeFormats": {
    "full": "{1} 'um' {0}",
    "long": "{1} 'um' {0}",
    "medium": "{1}, {0}",
    "short": "{1}, {0}",
    "availableFormats": {
      "d": "d",
      "E": "ccc",
      "Ed": "E, d.",
      "Ehm": "E h:mm a",
      "EHm": "E, HH:mm",
      "Ehms": "E, h:mm:ss a",
      "EHms": "E, HH:mm:ss",
      "Gy": "y G",
      "GyMMM": "MMM y G",
      "GyMMMd": "d. MMM y G",
      "GyMMMED": "E, d. MMM y G",
      "h": "h 'Uhr' a",
      "H": "HH 'Uhr'",
      "hm": "h:mm a",
      "Hm": "HH:mm",
      "hms": "h:mm:ss a",
      "Hms": "HH:mm:ss",
      "hmsv": "h:mm:ss a v",
      "Hmsv": "HH:mm:ss v",
      "hmv": "h:mm a v",
      "Hmv": "HH:mm v",
      "M": "L",
      "Md": "d.M.",
      "MEd": "E, d.M.",
      "MMd": "d.MM.",
      "MMdd": "dd.MM.",
      "MMM": "LLL",
      "MMMd": "d. MMM",
      "MMMED": "E, d. MMM",
      "MMMMd": "d. MMMM",
      "MMMMEd": "E, d. MMMM",
      "MMMMW": "'Woche' W 'im' MMM",

```



```

"MMMMW": "'Woche' W 'im' MMM",
"ms": "mm:ss",
"Y": "Y",
"yM": "M.y",
"yMd": "d.M.y",
"yMEd": "E, d.M.y",
"yMM": "MM.y",
"yMMdd": "dd.MM.y",
"yMMM": "MMM y",
"yMMMd": "d. MMM y",
"yMMMEd": "E, d. MMM y",
"yMMMM": "MMMM y",
"yQQQ": "QQQ y",
"yQQQQ": "QQQQ y",
"yw": "'Woche' w 'des' 'Jahres' y",
"yw": "'Woche' w 'des' 'Jahres' y"
},
"appendItems": {
  "Day": "{0} ({2}: {1})",
  "Day-Of-Week": "{0} {1}",
  "Era": "{1} {0}",
  "Hour": "{0} ({2}: {1})",
  "Minute": "{0} ({2}: {1})",
  "Month": "{0} ({2}: {1})",
  "Quarter": "{0} ({2}: {1})",
  "Second": "{0} ({2}: {1})",
  "Timezone": "{0} {1}",
  "Week": "{0} ({2}: {1})",
  "Year": "{1} {0}"
},
"intervalFormats": {
  "intervalFormatFallback": "{0} - {1}",
  "d": {
    "d": "d.-d."
  },
  "h": {
    "a": "h 'Uhr' a - h 'Uhr' a",
    "h": "h - h 'Uhr' a"
  },
  "H": {
    "H": "HH-HH 'Uhr'"
  },
  "hm": {
    "a": "h:mm a - h:mm a",
    "h": "h:mm-h:mm a",
    "m": "h:mm-h:mm a"
  },
  "Hm": {
    "H": "HH:mm-HH:mm 'Uhr'",
    "m": "HH:mm-HH:mm 'Uhr'"
  },
  "hmv": {
    "a": "h:mm a - h:mm a v",
    "h": "h:mm-h:mm a v",
    "m": "h:mm-h:mm a v"
  },
  "Hmv": {

```

```

    "H": "HH:mm-HH:mm 'Uhr' v",
    "m": "HH:mm-HH:mm 'Uhr' v"
  },
  "hv": {
    "a": "h a - h a v",
    "h": "h-h a v"
  },
  "Hv": {
    "H": "HH-HH 'Uhr' v"
  },
  "M": {
    "M": "M.-M."
  },
  "Md": {
    "d": "dd.MM. - dd.MM.",
    "M": "dd.MM. - dd.MM."
  },
  "MEd": {
    "d": "E, dd.MM. - E, dd.MM.",
    "M": "E, dd.MM. - E, dd.MM."
  },
  "MMM": {
    "M": "MMM-MMM"
  },
  "MMMd": {
    "d": "d.-d. MMM",
    "M": "d. MMM - d. MMM"
  },
  "MMMEd": {
    "d": "E, d. - E, d. MMM",
    "M": "E, d. MMM - E, d. MMM"
  },
  "MMMM": {
    "M": "LLLL-LLLL"
  },
  "Y": {
    "Y": "Y-Y"
  },
  "YM": {
    "M": "MM.y - MM.y",
    "Y": "MM.y - MM.y"
  },
  "YMd": {
    "d": "dd.MM.y - dd.MM.y",
    "M": "dd.MM.y - dd.MM.y",
    "Y": "dd.MM.y - dd.MM.y"
  },
  "YMEd": {
    "d": "E, dd.MM.y - E, dd.MM.y",
    "M": "E, dd.MM.y - E, dd.MM.y",
    "Y": "E, dd.MM.y - E, dd.MM.y"
  },
  "YMMM": {
    "M": "MMM-MMM y",
    "Y": "MMM y - MMM y"
  },
  "YMMMd": {

```

```

    "d": "d.-d. MMM y",
    "M": "d. MMM - d. MMM y",
    "y": "d. MMM y - d. MMM y"
},
    "yMMMEd": {
        "d": "E, d. - E, d. MMM y",
        "M": "E, d. MMM - E, d. MMM y",
        "y": "E, d. MMM y - E, d. MMM y"
    },
    "yMMMM": {
        "M": "MMMM-MMMM y",
        "y": "MMMM y - MMMM y"
    }
}
}
}
}
}
}
}
}
}
```

CA-GREGORIAN.JSX

```
{
  "main";
  {
    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 12879 $",
          "_cldrVersion";
          "30.0.3";
        }
        "language";
        "de";
      }
    }
    "dates";
    {
      "calendars";
      {
        "gregorian";
        {
          "months";
          {
            "format";
            {
              "abbreviated";
              {
                "1";
                "Jan.",
                "2";
              }
            }
          }
        }
      }
    }
  }
}
```

```
        "Feb.",  
        "3";  
        "März",  
        "4";  
        "Apr.",  
        "5";  
        "Mai",  
        "6";  
        "Juni",  
        "7";  
        "Juli",  
        "8";  
        "Aug.",  
        "9";  
        "Sep.",  
        "10";  
        "Okt.",  
        "11";  
        "Nov.",  
        "12";  
        "Dez.";  
    }  
    "narrow";  
    {  
        "1";  
        "J",  
        "2";  
        "F",  
        "3";  
        "M",  
        "4";  
        "A",  
        "5";  
        "M",  
        "6";  
        "J",  
        "7";  
        "J",  
        "8";  
        "A",  
        "9";  
        "S",  
        "10";  
        "O",  
        "11";  
        "N",  
        "12";  
        "D";  
    }  
    "wide";  
    {  
        "1";  
        "Januar",  
        "2";  
        "Februar",  
        "3";  
        "März",
```

```
        "4";
        "April",
        "5";
        "Mai",
        "6";
        "Juni",
        "7";
        "Juli",
        "8";
        "August",
        "9";
        "September",
        "10";
        "Oktober",
        "11";
        "November",
        "12";
        "Dezember";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "Jan",
        "2";
        "Feb",
        "3";
        "Mär",
        "4";
        "Apr",
        "5";
        "Mai",
        "6";
        "Jun",
        "7";
        "Jul",
        "8";
        "Aug",
        "9";
        "Sep",
        "10";
        "Okt",
        "11";
        "Nov",
        "12";
        "Dez";
    }
    "narrow";
    {
        "1";
        "J",
        "2";
        "F",
        "3";
        "M",
```

```

        "4";
        "A",
        "5";
        "M",
        "6";
        "J",
        "7";
        "J",
        "8";
        "A",
        "9";
        "S",
        "10";
        "O",
        "11";
        "N",
        "12";
        "D";
    }
    "wide";
    {
        "1";
        "Januar",
        "2";
        "Februar",
        "3";
        "März",
        "4";
        "April",
        "5";
        "Mai",
        "6";
        "Juni",
        "7";
        "Juli",
        "8";
        "August",
        "9";
        "September",
        "10";
        "Oktober",
        "11";
        "November",
        "12";
        "Dezember";
    }
}
"days";
{
    "format";
    {
        "abbreviated";
        {
            "sun";
            "So.",
            "mon";

```

```
        "Mo.",  
        "tue";  
        "Di.",  
        "wed";  
        "Mi.",  
        "thu";  
        "Do.",  
        "fri";  
        "Fr.",  
        "sat";  
        "Sa.";  
    }  
    "narrow";  
    {  
        "sun";  
        "S",  
        "mon";  
        "M",  
        "tue";  
        "D",  
        "wed";  
        "M",  
        "thu";  
        "D",  
        "fri";  
        "F",  
        "sat";  
        "S";  
    }  
    "short";  
    {  
        "sun";  
        "So.",  
        "mon";  
        "Mo.",  
        "tue";  
        "Di.",  
        "wed";  
        "Mi.",  
        "thu";  
        "Do.",  
        "fri";  
        "Fr.",  
        "sat";  
        "Sa.";  
    }  
    "wide";  
    {  
        "sun";  
        "Sonntag",  
        "mon";  
        "Montag",  
        "tue";  
        "Dienstag",  
        "wed";  
        "Mittwoch",  
        "thu";
```

```
        "Donnerstag",
        "fri";
        "Freitag",
        "sat";
        "Samstag";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "sun";
        "So",
        "mon";
        "Mo",
        "tue";
        "Di",
        "wed";
        "Mi",
        "thu";
        "Do",
        "fri";
        "Fr",
        "sat";
        "Sa";
    }
    "narrow";
    {
        "sun";
        "S",
        "mon";
        "M",
        "tue";
        "D",
        "wed";
        "M",
        "thu";
        "D",
        "fri";
        "F",
        "sat";
        "S";
    }
    "short";
    {
        "sun";
        "So.",
        "mon";
        "Mo.",
        "tue";
        "Di.",
        "wed";
        "Mi.",
        "thu";
        "Do.",
        "fri";
        "Fr.",
```



```

        "sat";
        "Sa.";
    }
    "wide";
    {
        "sun";
        "Sonntag",
        "mon";
        "Montag",
        "tue";
        "Dienstag",
        "wed";
        "Mittwoch",
        "thu";
        "Donnerstag",
        "fri";
        "Freitag",
        "sat";
        "Samstag";
    }
}
"quarters";
{
    "format";
    {
        "abbreviated";
        {
            "1";
            "Q1",
            "2";
            "Q2",
            "3";
            "Q3",
            "4";
            "Q4";
        }
        "narrow";
        {
            "1";
            "1",
            "2";
            "2",
            "3";
            "3",
            "4";
            "4";
        }
    }
    "wide";
    {
        "1";
        "1. Quartal",
        "2";
        "2. Quartal",
        "3";
        "3. Quartal",
        "4";
    }
}

```

```

        "4. Quartal";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "Q1",
        "2";
        "Q2",
        "3";
        "Q3",
        "4";
        "Q4";
    }
    "narrow";
    {
        "1";
        "1",
        "2";
        "2",
        "3";
        "3",
        "4";
        "4";
    }
    "wide";
    {
        "1";
        "1. Quartal",
        "2";
        "2. Quartal",
        "3";
        "3. Quartal",
        "4";
        "4. Quartal";
    }
}
}
"dayPeriods";
{
    "format";
    {
        "abbreviated";
        {
            "midnight";
            "Mitternacht",
            "am";
            "vorm.",
            "pm";
            "nachm.",
            "morning1";
            "morgens",
            "morning2";
            "vormittags",
            "afternoon1";
        }
    }
}

```

```

        "mittags",
        "afternoon2";
    "nachmittags",
        "evening1";
    "abends",
        "night1";
    "nachts";
}
"narrow";
{
    "midnight";
    "Mitternacht",
        "am";
    "vm.",
        "pm";
    "nm.",
        "morning1";
    "morgens",
        "morning2";
    "vormittags",
        "afternoon1";
    "mittags",
        "afternoon2";
    "nachmittags",
        "evening1";
    "abends",
        "night1";
    "nachts";
}
"wide";
{
    "midnight";
    "Mitternacht",
        "am";
    "vorm.",
        "pm";
    "nachm.",
        "morning1";
    "morgens",
        "morning2";
    "vormittags",
        "afternoon1";
    "mittags",
        "afternoon2";
    "nachmittags",
        "evening1";
    "abends",
        "night1";
    "nachts";
}
}
"stand-alone";
{
    "abbreviated";
    {
        "midnight";
        "Mitternacht",

```

```
        "am";
        "vorm.",
        "pm";
        "nachm.",
        "morning1";
        "Morgen",
        "morning2";
        "Vormittag",
        "afternoon1";
        "Mittag",
        "afternoon2";
        "Nachmittag",
        "evening1";
        "Abend",
        "night1";
        "Nacht";
    }
    "narrow";
    {
        "midnight";
        "Mitternacht",
        "am";
        "vorm.",
        "pm";
        "nachm.",
        "morning1";
        "Morgen",
        "morning2";
        "Vormittag",
        "afternoon1";
        "Mittag",
        "afternoon2";
        "Nachmittag",
        "evening1";
        "Abend",
        "night1";
        "Nacht";
    }
    "wide";
    {
        "midnight";
        "Mitternacht",
        "am";
        "vorm.",
        "pm";
        "nachm.",
        "morning1";
        "Morgen",
        "morning2";
        "Vormittag",
        "afternoon1";
        "Mittag",
        "afternoon2";
        "Nachmittag",
        "evening1";
        "Abend",
        "night1";
    }
```

```

        "Nacht";
    }
}
}
"eras";
{
    "eraNames";
    {
        "0";
        "v. Chr.",
        "0-alt-variant";
        "vor unserer Zeitrechnung",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "unserer Zeitrechnung";
    }
    "eraAbbr";
    {
        "0";
        "v. Chr.",
        "0-alt-variant";
        "v. u. Z.",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "u. Z.";
    }
    "eraNarrow";
    {
        "0";
        "v. Chr.",
        "0-alt-variant";
        "v. u. Z.",
        "1";
        "n. Chr.",
        "1-alt-variant";
        "u. Z.";
    }
}
"dateFormats";
{
    "full";
    "EEEE, d. MMMM y",
    "long";
    "d. MMMM y",
    "medium";
    "dd.MM.y",
    "short";
    "dd.MM.yy";
}
"timeFormats";
{
    "full";
    "HH:mm:ss zzzz",
    "long";
    "HH:mm:ss z",

```

```

        "medium";
        "HH:mm:ss",
        "short";
        "HH:mm";
    }
    "dateTimeFormats";
    {
        "full";
        "{1} 'um' {0}",
        "long";
        "{1} 'um' {0}",
        "medium";
        "{1}, {0}",
        "short";
        "{1}, {0}",
        "availableFormats";
        {
            "d";
            "d",
            "E";
            "ccc",
            "Ed";
            "E, d.",
            "Ehm";
            "E h:mm a",
            "EHm";
            "E, HH:mm",
            "Ehms";
            "E, h:mm:ss a",
            "EHms";
            "E, HH:mm:ss",
            "Gy";
            "y G",
            "GyMMM";
            "MMM y G",
            "GyMMMd";
            "d. MMM y G",
            "GyMMMED";
            "E, d. MMM y G",
            "h";
            "h 'Uhr' a",
            "H";
            "HH 'Uhr' ",
            "hm";
            "h:mm a",
            "Hm";
            "HH:mm",
            "hms";
            "h:mm:ss a",
            "Hms";
            "HH:mm:ss",
            "hmsv";
            "h:mm:ss a v",
            "Hmsv";
            "HH:mm:ss v",
            "hmv";
            "h:mm a v",

```

```

        "Hmv";
        "HH:mm v",
        "M";
        "L",
        "Md";
        "d.M.",
        "MEd";
        "E, d.M.",
        "MMd";
        "d.MM.",
        "MMdd";
        "dd.MM.",
        "MMM";
        "LLL",
        "MMMd";
        "d. MMM",
        "MMMEd";
        "E, d. MMM",
        "MMMMd";
        "d. MMMM",
        "MMMMEd";
        "E, d. MMMM",
        "MMMMW";
        "'Woche' W 'im' MMM",
        "MMMMW";
        "'Woche' W 'im' MMM",
        "ms";
        "mm:ss",
        "y";
        "Y",
        "yM";
        "M.y",
        "yMd";
        "d.M.y",
        "yMEd";
        "E, d.M.y",
        "yMM";
        "MM.y",
        "yMMdd";
        "dd.MM.y",
        "yMMM";
        "MMM y",
        "yMMMd";
        "d. MMM y",
        "yMMMEd";
        "E, d. MMM y",
        "yMMMM";
        "MMMM y",
        "yQQQ";
        "QQQ y",
        "yQQQQ";
        "QQQQ y",
        "yw";
        "'Woche' w 'des' 'Jahres' y",
        "yw";
        "'Woche' w 'des' 'Jahres' y";
    }

```

```

"appendItems";
{
  "Day";
  "{0} ({2}: {1})",
  "Day-Of-Week";
  "{0} {1}",
  "Era";
  "{1} {0}",
  "Hour";
  "{0} ({2}: {1})",
  "Minute";
  "{0} ({2}: {1})",
  "Month";
  "{0} ({2}: {1})",
  "Quarter";
  "{0} ({2}: {1})",
  "Second";
  "{0} ({2}: {1})",
  "Timezone";
  "{0} {1}",
  "Week";
  "{0} ({2}: {1})",
  "Year";
  "{1} {0}";
}
"intervalFormats";
{
  "intervalFormatFallback";
  "{0} - {1}",
  "d";
  {
    "d";
    "d.-d.";
  }
  "h";
  {
    "a";
    "h 'Uhr' a - h 'Uhr' a",
    "h";
    "h - h 'Uhr' a";
  }
  "H";
  {
    "H";
    "HH-HH 'Uhr'";
  }
  "hm";
  {
    "a";
    "h:mm a - h:mm a",
    "h";
    "h:mm-h:mm a",
    "m";
    "h:mm-h:mm a";
  }
  "Hm";
  {

```



```

        "H";
        "HH:mm-HH:mm 'Uhr'",
        "m";
        "HH:mm-HH:mm 'Uhr'";
    }
    "hmv";
    {
        "a";
        "h:mm a - h:mm a v",
        "h";
        "h:mm-h:mm a v",
        "m";
        "h:mm-h:mm a v";
    }
    "Hmv";
    {
        "H";
        "HH:mm-HH:mm 'Uhr' v",
        "m";
        "HH:mm-HH:mm 'Uhr' v";
    }
    "hv";
    {
        "a";
        "h a - h a v",
        "h";
        "h-h a v";
    }
    "Hv";
    {
        "H";
        "HH-HH 'Uhr' v";
    }
    "M";
    {
        "M";
        "M.-M.";
    }
    "Md";
    {
        "d";
        "dd.MM. - dd.MM.",
        "M";
        "dd.MM. - dd.MM.";
    }
    "MED";
    {
        "d";
        "E, dd.MM. - E, dd.MM.",
        "M";
        "E, dd.MM. - E, dd.MM.";
    }
    "MMM";
    {
        "M";
        "MMM-MMM";
    }
}

```

```

"MMMd";
{
  "d";
  "d.-d. MMM",
  "M";
  "d. MMM - d. MMM";
}
"MMMED";
{
  "d";
  "E, d. - E, d. MMM",
  "M";
  "E, d. MMM - E, d. MMM";
}
"MMMM";
{
  "M";
  "LLLL-LLLL";
}
"Y";
{
  "Y";
  "Y-Y";
}
"YM";
{
  "M";
  "MM.Y - MM.Y",
  "Y";
  "MM.Y - MM.Y";
}
"yMd";
{
  "d";
  "dd.MM.Y - dd.MM.Y",
  "M";
  "dd.MM.Y - dd.MM.Y",
  "Y";
  "dd.MM.Y - dd.MM.Y";
}
"yMED";
{
  "d";
  "E, dd.MM.Y - E, dd.MM.Y",
  "M";
  "E, dd.MM.Y - E, dd.MM.Y",
  "Y";
  "E, dd.MM.Y - E, dd.MM.Y";
}
"yMMM";
{
  "M";
  "MMM-MMM Y",
  "Y";
  "MMM Y - MMM Y";
}
"yMMMd";

```

```
{
    "d";
    "d.-d. MMM y",
        "M";
    "d. MMM - d. MMM y",
        "y";
    "d. MMM y - d. MMM y";
}
"yMMMEd";
{
    "d";
    "E, d. - E, d. MMM y",
        "M";
    "E, d. MMM - E, d. MMM y",
        "y";
    "E, d. MMM y - E, d. MMM y";
}
"yMMMM";
{
    "M";
    "MMMM-MMMM y",
        "y";
    "MMMM y - MMMM y";
}
}
}
}
}
}
}
```

CURRENCIES.JSON

```
{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 13259 $",
          "_cldrVersion": "31"
        },
        "language": "de"
      },
      "numbers": {
        "currencies": {
          "ADP": {
            "displayName": "Andorranische Pesete",
            "displayName-count-one": "Andorranische Pesete",
            "displayName-count-other": "Andorranische Peseten",
            "symbol": "ADP"
          },
          "AED": {
            "displayName": "VAE-Dirham",
            "displayName-count-one": "VAE-Dirham",
            "displayName-count-other": "VAE-Dirhamen"
          }
        }
      }
    }
  }
}
```

```

        "displayName-count-other": "VAE-Dirham",
        "symbol": "AED"
    },
    "AFA": {
        "displayName": "Afghanische Afghani (1927-2002)",
        "displayName-count-one": "Afghanische Afghani (1927-2002)",
        "displayName-count-other": "Afghanische Afghani (1927-2002)",
        "symbol": "AFA"
    },
    "AFN": {
        "displayName": "Afghanischer Afghani",
        "displayName-count-one": "Afghanischer Afghani",
        "displayName-count-other": "Afghanische Afghani",
        "symbol": "AFN"
    },
    "ALK": {
        "displayName": "Albanischer Lek (1946-1965)",
        "displayName-count-one": "Albanischer Lek (1946-1965)",
        "displayName-count-other": "Albanische Lek (1946-1965)"
    },
    "ALL": {
        "displayName": "Albanischer Lek",
        "displayName-count-one": "Albanischer Lek",
        "displayName-count-other": "Albanische Lek",
        "symbol": "ALL"
    },
    "AMD": {
        "displayName": "Armenischer Dram",
        "displayName-count-one": "Armenischer Dram",
        "displayName-count-other": "Armenische Dram",
        "symbol": "AMD"
    },
    "ANG": {
        "displayName": "Niederländische-Antillen-Gulden",
        "displayName-count-one": "Niederländische-Antillen-Gulden",
        "displayName-count-other": "Niederländische-Antillen-Gulden",
        "symbol": "ANG"
    },
    "AOA": {
        "displayName": "Angolanischer Kwanza",
        "displayName-count-one": "Angolanischer Kwanza",
        "displayName-count-other": "Angolanische Kwanza",
        "symbol": "AOA",
        "symbol-alt-narrow": "Kz"
    },
    "AOK": {
        "displayName": "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one": "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other": "Angolanische Kwanza (1977-1990)",
        "symbol": "AOK"
    },
    "AON": {
        "displayName": "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one": "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-other": "Angolanische Neue Kwanza (1990-2000)",
    },

```

```

        "symbol": "AON"
    },
    "AOR": {
        "displayName": "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one": "Angolanischer Kwanza Reajustado (1995-
1999) ",
        "displayName-count-other": "Angolanische Kwanza Reajustado
(1995-1999) ",
        "symbol": "AOR"
    },
    "ARA": {
        "displayName": "Argentinischer Austral",
        "displayName-count-one": "Argentinischer Austral",
        "displayName-count-other": "Argentinische Austral",
        "symbol": "ARA"
    },
    "ARL": {
        "displayName": "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one": "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other": "Argentinische Pesos Ley (1970-
1983) ",
        "symbol": "ARL"
    },
    "ARM": {
        "displayName": "Argentinischer Peso (1881-1970)",
        "displayName-count-one": "Argentinischer Peso (1881-1970)",
        "displayName-count-other": "Argentinische Pesos (1881-1970)",
        "symbol": "ARM"
    },
    "ARP": {
        "displayName": "Argentinischer Peso (1983-1985)",
        "displayName-count-one": "Argentinischer Peso (1983-1985)",
        "displayName-count-other": "Argentinische Peso (1983-1985)",
        "symbol": "ARP"
    },
    "ARS": {
        "displayName": "Argentinischer Peso",
        "displayName-count-one": "Argentinischer Peso",
        "displayName-count-other": "Argentinische Pesos",
        "symbol": "ARS",
        "symbol-alt-narrow": "$"
    },
    "ATS": {
        "displayName": "Österreichischer Schilling",
        "displayName-count-one": "Österreichischer Schilling",
        "displayName-count-other": "Österreichische Schilling",
        "symbol": "öS"
    },
    "AUD": {
        "displayName": "Australischer Dollar",
        "displayName-count-one": "Australischer Dollar",
        "displayName-count-other": "Australische Dollar",
        "symbol": "AU$",
        "symbol-alt-narrow": "$"
    },
    "AWG": {
        "displayName": "Aruba-Florin",

```

```

        "displayName-count-one": "Aruba-Florin",
        "displayName-count-other": "Aruba-Florin",
        "symbol": "AWG"
    },
    "AZM": {
        "displayName": "Aserbajdschan-Manat (1993-2006)",
        "displayName-count-one": "Aserbajdschan-Manat (1993-2006)",
        "displayName-count-other": "Aserbajdschan-Manat (1993-2006)",
        "symbol": "AZM"
    },
    "AZN": {
        "displayName": "Aserbajdschan-Manat",
        "displayName-count-one": "Aserbajdschan-Manat",
        "displayName-count-other": "Aserbajdschan-Manat",
        "symbol": "AZN"
    },
    "BAD": {
        "displayName": "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-one": "Bosnien und Herzegowina Dinar (1992-
1994)",
        "displayName-count-other": "Bosnien und Herzegowina Dinar (1992-
1994)",
        "symbol": "BAD"
    },
    "BAM": {
        "displayName": "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one": "Bosnien und Herzegowina Konvertierbare
Mark",
        "displayName-count-other": "Bosnien und Herzegowina
Konvertierbare Mark",
        "symbol": "BAM",
        "symbol-alt-narrow": "KM"
    },
    "BAN": {
        "displayName": "Bosnien und Herzegowina Neuer Dinar (1994-
1997)",
        "displayName-count-one": "Bosnien und Herzegowina Neuer Dinar
(1994-1997)",
        "displayName-count-other": "Bosnien und Herzegowina Neue Dinar
(1994-1997)",
        "symbol": "BAN"
    },
    "BBD": {
        "displayName": "Barbados-Dollar",
        "displayName-count-one": "Barbados-Dollar",
        "displayName-count-other": "Barbados-Dollar",
        "symbol": "BBD",
        "symbol-alt-narrow": "$"
    },
    "BDT": {
        "displayName": "Bangladesch-Taka",
        "displayName-count-one": "Bangladesch-Taka",
        "displayName-count-other": "Bangladesch-Taka",
        "symbol": "BDT",
        "symbol-alt-narrow": "৳"
    },
    },

```

```

"BEC": {
  "displayName": "Belgischer Franc (konvertibel)",
  "displayName-count-one": "Belgischer Franc (konvertibel)",
  "displayName-count-other": "Belgische Franc (konvertibel)",
  "symbol": "BEC"
},
"BEF": {
  "displayName": "Belgischer Franc",
  "displayName-count-one": "Belgischer Franc",
  "displayName-count-other": "Belgische Franc",
  "symbol": "BEF"
},
"BEL": {
  "displayName": "Belgischer Finanz-Franc",
  "displayName-count-one": "Belgischer Finanz-Franc",
  "displayName-count-other": "Belgische Finanz-Franc",
  "symbol": "BEL"
},
"BGL": {
  "displayName": "Bulgarische Lew (1962-1999)",
  "displayName-count-one": "Bulgarische Lew (1962-1999)",
  "displayName-count-other": "Bulgarische Lew (1962-1999)",
  "symbol": "BGL"
},
"BGM": {
  "displayName": "Bulgarischer Lew (1952-1962)",
  "displayName-count-one": "Bulgarischer Lew (1952-1962)",
  "displayName-count-other": "Bulgarische Lew (1952-1962)",
  "symbol": "BGK"
},
"BGN": {
  "displayName": "Bulgarischer Lew",
  "displayName-count-one": "Bulgarischer Lew",
  "displayName-count-other": "Bulgarische Lew",
  "symbol": "BGN"
},
"BGO": {
  "displayName": "Bulgarischer Lew (1879-1952)",
  "displayName-count-one": "Bulgarischer Lew (1879-1952)",
  "displayName-count-other": "Bulgarische Lew (1879-1952)",
  "symbol": "BGJ"
},
"BHD": {
  "displayName": "Bahrain-Dinar",
  "displayName-count-one": "Bahrain-Dinar",
  "displayName-count-other": "Bahrain-Dinar",
  "symbol": "BHD"
},
"BIF": {
  "displayName": "Burundi-Franc",
  "displayName-count-one": "Burundi-Franc",
  "displayName-count-other": "Burundi-Francs",
  "symbol": "BIF"
},
"BMD": {
  "displayName": "Bermuda-Dollar",
  "displayName-count-one": "Bermuda-Dollar",

```

```

        "displayName-count-other": "Bermuda-Dollar",
        "symbol": "BMD",
        "symbol-alt-narrow": "$"
    },
    "BND": {
        "displayName": "Brunei-Dollar",
        "displayName-count-one": "Brunei-Dollar",
        "displayName-count-other": "Brunei-Dollar",
        "symbol": "BND",
        "symbol-alt-narrow": "$"
    },
    "BOB": {
        "displayName": "Bolivanischer Boliviano",
        "displayName-count-one": "Bolivanischer Boliviano",
        "displayName-count-other": "Bolivianische Bolivianos",
        "symbol": "BOB",
        "symbol-alt-narrow": "Bs"
    },
    "BOL": {
        "displayName": "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-one": "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-other": "Bolivianische Bolivianos (1863-
1963)",
        "symbol": "BOL"
    },
    "BOP": {
        "displayName": "Bolivianischer Peso",
        "displayName-count-one": "Bolivianischer Peso",
        "displayName-count-other": "Bolivianische Peso",
        "symbol": "BOP"
    },
    "BOV": {
        "displayName": "Boliviansiche Mvdol",
        "displayName-count-one": "Boliviansiche Mvdol",
        "displayName-count-other": "Bolivianische Mvdol",
        "symbol": "BOV"
    },
    "BRB": {
        "displayName": "Brasilianischer Cruzeiro Novo (1967-1986)",
        "displayName-count-one": "Brasilianischer Cruzeiro Novo (1967-
1986)",
        "displayName-count-other": "Brasilianische Cruzeiro Novo (1967-
1986)",
        "symbol": "BRB"
    },
    "BRC": {
        "displayName": "Brasilianischer Cruzado (1986-1989)",
        "displayName-count-one": "Brasilianischer Cruzado (1986-1989)",
        "displayName-count-other": "Brasilianische Cruzado (1986-1989)",
        "symbol": "BRC"
    },
    "BRE": {
        "displayName": "Brasilianischer Cruzeiro (1990-1993)",
        "displayName-count-one": "Brasilianischer Cruzeiro (1990-1993)",
        "displayName-count-other": "Brasilianische Cruzeiro (1990-
1993)",
        "symbol": "BRE"
    }

```



```

    },
    "BRL": {
      "displayName": "Brasilianischer Real",
      "displayName-count-one": "Brasilianischer Real",
      "displayName-count-other": "Brasilianische Real",
      "symbol": "R$",
      "symbol-alt-narrow": "R$"
    },
    "BRN": {
      "displayName": "Brasilianischer Cruzado Novo (1989-1990)",
      "displayName-count-one": "Brasilianischer Cruzado Novo (1989-
1990)",
      "displayName-count-other": "Brasilianische Cruzado Novo (1989-
1990)",
      "symbol": "BRN"
    },
    "BRR": {
      "displayName": "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-one": "Brasilianischer Cruzeiro (1993-1994)",
      "displayName-count-other": "Brasilianische Cruzeiro (1993-
1994)",
      "symbol": "BRR"
    },
    "BRZ": {
      "displayName": "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-one": "Brasilianischer Cruzeiro (1942-1967)",
      "displayName-count-other": "Brasilianischer Cruzeiro (1942-
1967)",
      "symbol": "BRZ"
    },
    "BSD": {
      "displayName": "Bahamas-Dollar",
      "displayName-count-one": "Bahamas-Dollar",
      "displayName-count-other": "Bahamas-Dollar",
      "symbol": "BSD",
      "symbol-alt-narrow": "$"
    },
    "BTN": {
      "displayName": "Bhutan-Ngultrum",
      "displayName-count-one": "Bhutan-Ngultrum",
      "displayName-count-other": "Bhutan-Ngultrum",
      "symbol": "BTN"
    },
    "BUK": {
      "displayName": "Birmanischer Kyat",
      "displayName-count-one": "Birmanischer Kyat",
      "displayName-count-other": "Birmanische Kyat",
      "symbol": "BUK"
    },
    "BWP": {
      "displayName": "Botswanischer Pula",
      "displayName-count-one": "Botswanischer Pula",
      "displayName-count-other": "Botswanische Pula",
      "symbol": "BWP",
      "symbol-alt-narrow": "P"
    },
    "BYB": {

```

```

        "displayName": "Belarus-Rubel (1994-1999)",
        "displayName-count-one": "Belarus-Rubel (1994-1999)",
        "displayName-count-other": "Belarus-Rubel (1994-1999)",
        "symbol": "BYB"
    },
    "BYN": {
        "displayName": "Weißrussischer Rubel",
        "displayName-count-one": "Weißrussischer Rubel",
        "displayName-count-other": "Weißrussische Rubel",
        "symbol": "BYN",
        "symbol-alt-narrow": "p."
    },
    "BYR": {
        "displayName": "Weißrussischer Rubel (2000-2016)",
        "displayName-count-one": "Weißrussischer Rubel (2000-2016)",
        "displayName-count-other": "Weißrussische Rubel (2000-2016)",
        "symbol": "BYR"
    },
    "BZD": {
        "displayName": "Belize-Dollar",
        "displayName-count-one": "Belize-Dollar",
        "displayName-count-other": "Belize-Dollar",
        "symbol": "BZD",
        "symbol-alt-narrow": "$"
    },
    "CAD": {
        "displayName": "Kanadischer Dollar",
        "displayName-count-one": "Kanadischer Dollar",
        "displayName-count-other": "Kanadische Dollar",
        "symbol": "CA$",
        "symbol-alt-narrow": "$"
    },
    "CDF": {
        "displayName": "Kongo-Franc",
        "displayName-count-one": "Kongo-Franc",
        "displayName-count-other": "Kongo-Francs",
        "symbol": "CDF"
    },
    "CHE": {
        "displayName": "WIR-Euro",
        "displayName-count-one": "WIR-Euro",
        "displayName-count-other": "WIR-Euro",
        "symbol": "CHE"
    },
    "CHF": {
        "displayName": "Schweizer Franken",
        "displayName-count-one": "Schweizer Franken",
        "displayName-count-other": "Schweizer Franken",
        "symbol": "CHF"
    },
    "CHW": {
        "displayName": "WIR Franken",
        "displayName-count-one": "WIR Franken",
        "displayName-count-other": "WIR Franken",
        "symbol": "CHW"
    },
    "CLE": {

```

```

        "displayName": "Chilenischer Escudo",
        "displayName-count-one": "Chilenischer Escudo",
        "displayName-count-other": "Chilenische Escudo",
        "symbol": "CLE"
    },
    "CLF": {
        "displayName": "Chilenische Unidades de Fomento",
        "displayName-count-one": "Chilenische Unidades de Fomento",
        "displayName-count-other": "Chilenische Unidades de Fomento",
        "symbol": "CLF"
    },
    "CLP": {
        "displayName": "Chilenischer Peso",
        "displayName-count-one": "Chilenischer Peso",
        "displayName-count-other": "Chilenische Pesos",
        "symbol": "CLP",
        "symbol-alt-narrow": "$"
    },
    "CNX": {
        "displayName": "Dollar der Chinesischen Volksbank",
        "displayName-count-one": "Dollar der Chinesischen Volksbank",
        "displayName-count-other": "Dollar der Chinesischen Volksbank",
        "symbol": "CNX"
    },
    "CNY": {
        "displayName": "Renminbi Yuan",
        "displayName-count-one": "Chinesischer Yuan",
        "displayName-count-other": "Renminbi Yuan",
        "symbol": "CN¥",
        "symbol-alt-narrow": "¥"
    },
    "COP": {
        "displayName": "Kolumbianischer Peso",
        "displayName-count-one": "Kolumbianischer Peso",
        "displayName-count-other": "Kolumbianische Pesos",
        "symbol": "COP",
        "symbol-alt-narrow": "$"
    },
    "COU": {
        "displayName": "Kolumbianische Unidades de valor real",
        "displayName-count-one": "Kolumbianische Unidad de valor real",
        "displayName-count-other": "Kolumbianische Unidades de valor
real",
        "symbol": "COU"
    },
    "CRC": {
        "displayName": "Costa-Rica-Colón",
        "displayName-count-one": "Costa-Rica-Colón",
        "displayName-count-other": "Costa-Rica-Colón",
        "symbol": "CRC",
        "symbol-alt-narrow": "₡"
    },
    "CSD": {
        "displayName": "Serbischer Dinar (2002-2006)",
        "displayName-count-one": "Serbischer Dinar (2002-2006)",
        "displayName-count-other": "Serbische Dinar (2002-2006)",
        "symbol": "CSD"
    }

```

```

    },
    "CSK": {
      "displayName": "Tschechoslowakische Krone",
      "displayName-count-one": "Tschechoslowakische Kronen",
      "displayName-count-other": "Tschechoslowakische Kronen",
      "symbol": "CSK"
    },
    "CUC": {
      "displayName": "Kubanischer Peso (konvertibel)",
      "displayName-count-one": "Kubanischer Peso (konvertibel)",
      "displayName-count-other": "Kubanische Pesos (konvertibel)",
      "symbol": "CUC",
      "symbol-alt-narrow": "Cub$"
    },
    "CUP": {
      "displayName": "Kubanischer Peso",
      "displayName-count-one": "Kubanischer Peso",
      "displayName-count-other": "Kubanische Pesos",
      "symbol": "CUP",
      "symbol-alt-narrow": "$"
    },
    "CVE": {
      "displayName": "Cabo-Verde-Escudo",
      "displayName-count-one": "Cabo-Verde-Escudo",
      "displayName-count-other": "Cabo-Verde-Escudos",
      "symbol": "CVE"
    },
    "CYP": {
      "displayName": "Zypern-Pfund",
      "displayName-count-one": "Zypern Pfund",
      "displayName-count-other": "Zypern Pfund",
      "symbol": "CYP"
    },
    "CZK": {
      "displayName": "Tschechische Krone",
      "displayName-count-one": "Tschechische Krone",
      "displayName-count-other": "Tschechische Kronen",
      "symbol": "CZK",
      "symbol-alt-narrow": "Kč"
    },
    "DDM": {
      "displayName": "Mark der DDR",
      "displayName-count-one": "Mark der DDR",
      "displayName-count-other": "Mark der DDR",
      "symbol": "DDM"
    },
    "DEM": {
      "displayName": "Deutsche Mark",
      "displayName-count-one": "Deutsche Mark",
      "displayName-count-other": "Deutsche Mark",
      "symbol": "DM"
    },
    "DJF": {
      "displayName": "Dschibuti-Franc",
      "displayName-count-one": "Dschibuti-Franc",
      "displayName-count-other": "Dschibuti-Franc",
      "symbol": "DJF"
    }
  }

```

```

    },
    "DKK": {
      "displayName": "Dänische Krone",
      "displayName-count-one": "Dänische Krone",
      "displayName-count-other": "Dänische Kronen",
      "symbol": "DKK",
      "symbol-alt-narrow": "kr"
    },
    "DOP": {
      "displayName": "Dominikanischer Peso",
      "displayName-count-one": "Dominikanischer Peso",
      "displayName-count-other": "Dominikanische Pesos",
      "symbol": "DOP",
      "symbol-alt-narrow": "$"
    },
    "DZD": {
      "displayName": "Algerischer Dinar",
      "displayName-count-one": "Algerischer Dinar",
      "displayName-count-other": "Algerische Dinar",
      "symbol": "DZD"
    },
    "ECS": {
      "displayName": "Ecuadorianischer Sucre",
      "displayName-count-one": "Ecuadorianischer Sucre",
      "displayName-count-other": "Ecuadorianische Sucre",
      "symbol": "ECS"
    },
    "ECV": {
      "displayName": "Verrechnungseinheit für Ecuador",
      "displayName-count-one": "Verrechnungseinheiten für Ecuador",
      "displayName-count-other": "Verrechnungseinheiten für Ecuador",
      "symbol": "ECV"
    },
    "EEK": {
      "displayName": "Estnische Krone",
      "displayName-count-one": "Estnische Krone",
      "displayName-count-other": "Estnische Kronen",
      "symbol": "EEK"
    },
    "EGP": {
      "displayName": "Ägyptisches Pfund",
      "displayName-count-one": "Ägyptisches Pfund",
      "displayName-count-other": "Ägyptische Pfund",
      "symbol": "EGP",
      "symbol-alt-narrow": "£"
    },
    "ERN": {
      "displayName": "Eritreischer Nakfa",
      "displayName-count-one": "Eritreischer Nakfa",
      "displayName-count-other": "Eritreische Nakfa",
      "symbol": "ERN"
    },
    "ESA": {
      "displayName": "Spanische Peseta (A-Konten)",
      "displayName-count-one": "Spanische Peseta (A-Konten)",
      "displayName-count-other": "Spanische Peseten (A-Konten)",
      "symbol": "ESA"
    }
  }

```

```

    },
    "ESB": {
      "displayName": "Spanische Peseta (konvertibel)",
      "displayName-count-one": "Spanische Peseta (konvertibel)",
      "displayName-count-other": "Spanische Peseten (konvertibel)",
      "symbol": "ESB"
    },
    "ESP": {
      "displayName": "Spanische Peseta",
      "displayName-count-one": "Spanische Peseta",
      "displayName-count-other": "Spanische Peseten",
      "symbol": "ESP",
      "symbol-alt-narrow": "₧"
    },
    "ETB": {
      "displayName": "Äthiopischer Birr",
      "displayName-count-one": "Äthiopischer Birr",
      "displayName-count-other": "Äthiopische Birr",
      "symbol": "ETB"
    },
    "EUR": {
      "displayName": "Euro",
      "displayName-count-one": "Euro",
      "displayName-count-other": "Euro",
      "symbol": "€",
      "symbol-alt-narrow": "€"
    },
    "FIM": {
      "displayName": "Finnische Mark",
      "displayName-count-one": "Finnische Mark",
      "displayName-count-other": "Finnische Mark",
      "symbol": "FIM"
    },
    "FJD": {
      "displayName": "Fidschi-Dollar",
      "displayName-count-one": "Fidschi-Dollar",
      "displayName-count-other": "Fidschi-Dollar",
      "symbol": "FJD",
      "symbol-alt-narrow": "$"
    },
    "FKP": {
      "displayName": "Falkland-Pfund",
      "displayName-count-one": "Falkland-Pfund",
      "displayName-count-other": "Falkland-Pfund",
      "symbol": "FKP",
      "symbol-alt-narrow": "Fl£"
    },
    "FRF": {
      "displayName": "Französischer Franc",
      "displayName-count-one": "Französischer Franc",
      "displayName-count-other": "Französische Franc",
      "symbol": "FRF"
    },
    "GBP": {
      "displayName": "Britisches Pfund",
      "displayName-count-one": "Britisches Pfund",
      "displayName-count-other": "Britische Pfund",

```

```

        "symbol": "₺",
        "symbol-alt-narrow": "₺"
    },
    "GEK": {
        "displayName": "Georgischer Kupon Larit",
        "displayName-count-one": "Georgischer Kupon Larit",
        "displayName-count-other": "Georgische Kupon Larit",
        "symbol": "GEK"
    },
    "GEL": {
        "displayName": "Georgischer Lari",
        "displayName-count-one": "Georgischer Lari",
        "displayName-count-other": "Georgische Lari",
        "symbol": "GEL",
        "symbol-alt-narrow": "ლ",
        "symbol-alt-variant": "ლ"
    },
    "GHC": {
        "displayName": "Ghanaischer Cedi (1979–2007)",
        "displayName-count-one": "Ghanaischer Cedi (1979–2007)",
        "displayName-count-other": "Ghanaische Cedi (1979–2007)",
        "symbol": "GHC"
    },
    "GHS": {
        "displayName": "Ghanaischer Cedi",
        "displayName-count-one": "Ghanaischer Cedi",
        "displayName-count-other": "Ghanaische Cedi",
        "symbol": "GHS"
    },
    "GIP": {
        "displayName": "Gibraltar-Pfund",
        "displayName-count-one": "Gibraltar-Pfund",
        "displayName-count-other": "Gibraltar Pfund",
        "symbol": "GIP",
        "symbol-alt-narrow": "£"
    },
    "GMD": {
        "displayName": "Gambia-Dalasi",
        "displayName-count-one": "Gambia-Dalasi",
        "displayName-count-other": "Gambia-Dalasi",
        "symbol": "GMD"
    },
    "GNF": {
        "displayName": "Guinea-Franc",
        "displayName-count-one": "Guinea-Franc",
        "displayName-count-other": "Guinea-Franc",
        "symbol": "GNF",
        "symbol-alt-narrow": "F.G."
    },
    "GNS": {
        "displayName": "Guineischer Syli",
        "displayName-count-one": "Guineischer Syli",
        "displayName-count-other": "Guineische Syli",
        "symbol": "GNS"
    },
    "GQE": {
        "displayName": "Äquatorialguinea-Ekwele",

```

```

    "displayName-count-one": "Äquatorialguinea-Ekwele",
    "displayName-count-other": "Äquatorialguinea-Ekwele",
    "symbol": "GQE"
  },
  "GRD": {
    "displayName": "Griechische Drachme",
    "displayName-count-one": "Griechische Drachme",
    "displayName-count-other": "Griechische Drachmen",
    "symbol": "GRD"
  },
  "GTQ": {
    "displayName": "Guatemaltekinscher Quetzal",
    "displayName-count-one": "Guatemaltekinscher Quetzal",
    "displayName-count-other": "Guatemaltekinsche Quetzales",
    "symbol": "GTQ",
    "symbol-alt-narrow": "Q"
  },
  "GWE": {
    "displayName": "Portugiesisch Guinea Escudo",
    "displayName-count-one": "Portugiesisch Guinea Escudo",
    "displayName-count-other": "Portugiesisch Guinea Escudo",
    "symbol": "GWE"
  },
  "GWP": {
    "displayName": "Guinea-Bissau Peso",
    "displayName-count-one": "Guinea-Bissau Peso",
    "displayName-count-other": "Guinea-Bissau Pesos",
    "symbol": "GWP"
  },
  "GYD": {
    "displayName": "Guyana-Dollar",
    "displayName-count-one": "Guyana-Dollar",
    "displayName-count-other": "Guyana-Dollar",
    "symbol": "GYD",
    "symbol-alt-narrow": "$"
  },
  "HKD": {
    "displayName": "Hongkong-Dollar",
    "displayName-count-one": "Hongkong-Dollar",
    "displayName-count-other": "Hongkong-Dollar",
    "symbol": "HK$",
    "symbol-alt-narrow": "$"
  },
  "HNL": {
    "displayName": "Honduras-Lempira",
    "displayName-count-one": "Honduras-Lempira",
    "displayName-count-other": "Honduras-Lempira",
    "symbol": "HNL",
    "symbol-alt-narrow": "L"
  },
  "HRD": {
    "displayName": "Kroatischer Dinar",
    "displayName-count-one": "Kroatischer Dinar",
    "displayName-count-other": "Kroatische Dinar",
    "symbol": "HRD"
  },
  "HRK": {

```



```

        "displayName": "Kroatischer Kuna",
        "displayName-count-one": "Kroatischer Kuna",
        "displayName-count-other": "Kroatische Kuna",
        "symbol": "HRK",
        "symbol-alt-narrow": "kn"
    },
    "HTG": {
        "displayName": "Haitianische Gourde",
        "displayName-count-one": "Haitianische Gourde",
        "displayName-count-other": "Haitianische Gourdes",
        "symbol": "HTG"
    },
    "HUF": {
        "displayName": "Ungarischer Forint",
        "displayName-count-one": "Ungarischer Forint",
        "displayName-count-other": "Ungarische Forint",
        "symbol": "HUF",
        "symbol-alt-narrow": "Ft"
    },
    "IDR": {
        "displayName": "Indonesische Rupiah",
        "displayName-count-one": "Indonesische Rupiah",
        "displayName-count-other": "Indonesische Rupiah",
        "symbol": "IDR",
        "symbol-alt-narrow": "Rp"
    },
    "IEP": {
        "displayName": "Irishes Pfund",
        "displayName-count-one": "Irishes Pfund",
        "displayName-count-other": "Irische Pfund",
        "symbol": "IEP"
    },
    "ILP": {
        "displayName": "Israelisches Pfund",
        "displayName-count-one": "Israelisches Pfund",
        "displayName-count-other": "Israelische Pfund",
        "symbol": "ILP"
    },
    "ILR": {
        "displayName": "Israelischer Schekel (1980-1985)",
        "displayName-count-one": "Israelischer Schekel (1980-1985)",
        "displayName-count-other": "Israelische Schekel (1980-1985)"
    },
    "ILS": {
        "displayName": "Israelischer Neuer Schekel",
        "displayName-count-one": "Israelischer Neuer Schekel",
        "displayName-count-other": "Israelische Neue Schekel",
        "symbol": "₪",
        "symbol-alt-narrow": "₪"
    },
    "INR": {
        "displayName": "Indische Rupie",
        "displayName-count-one": "Indische Rupie",
        "displayName-count-other": "Indische Rupien",
        "symbol": "₹",
        "symbol-alt-narrow": "₹"
    },
    },

```

```
"IQD": {
  "displayName": "Irakischer Dinar",
  "displayName-count-one": "Irakischer Dinar",
  "displayName-count-other": "Irakische Dinar",
  "symbol": "IQD"
},
"IRR": {
  "displayName": "Iranischer Rial",
  "displayName-count-one": "Iranischer Rial",
  "displayName-count-other": "Iranische Rial",
  "symbol": "IRR"
},
"ISJ": {
  "displayName": "Isländische Krone (1918-1981)",
  "displayName-count-one": "Isländische Krone (1918-1981)",
  "displayName-count-other": "Isländische Kronen (1918-1981)"
},
"ISK": {
  "displayName": "Isländische Krone",
  "displayName-count-one": "Isländische Krone",
  "displayName-count-other": "Isländische Kronen",
  "symbol": "ISK",
  "symbol-alt-narrow": "kr"
},
"ITL": {
  "displayName": "Italienische Lira",
  "displayName-count-one": "Italienische Lira",
  "displayName-count-other": "Italienische Lire",
  "symbol": "ITL"
},
"JMD": {
  "displayName": "Jamaika-Dollar",
  "displayName-count-one": "Jamaika-Dollar",
  "displayName-count-other": "Jamaika-Dollar",
  "symbol": "JMD",
  "symbol-alt-narrow": "$"
},
"JOD": {
  "displayName": "Jordanischer Dinar",
  "displayName-count-one": "Jordanischer Dinar",
  "displayName-count-other": "Jordanische Dinar",
  "symbol": "JOD"
},
"JPY": {
  "displayName": "Japanischer Yen",
  "displayName-count-one": "Japanischer Yen",
  "displayName-count-other": "Japanische Yen",
  "symbol": "¥",
  "symbol-alt-narrow": "¥"
},
"KES": {
  "displayName": "Kenia-Schilling",
  "displayName-count-one": "Kenia-Schilling",
  "displayName-count-other": "Kenia-Schilling",
  "symbol": "KES"
},
"KGS": {
```

```

        "displayName": "Kirgisischer Som",
        "displayName-count-one": "Kirgisischer Som",
        "displayName-count-other": "Kirgisische Som",
        "symbol": "KGS"
    },
    "KHR": {
        "displayName": "Kambodschanischer Riel",
        "displayName-count-one": "Kambodschanischer Riel",
        "displayName-count-other": "Kambodschanische Riel",
        "symbol": "KHR",
        "symbol-alt-narrow": "៛"
    },
    "KMF": {
        "displayName": "Komoren-Franc",
        "displayName-count-one": "Komoren-Franc",
        "displayName-count-other": "Komoren-Francs",
        "symbol": "KMF",
        "symbol-alt-narrow": "FC"
    },
    "KPW": {
        "displayName": "Nordkoreanischer Won",
        "displayName-count-one": "Nordkoreanischer Won",
        "displayName-count-other": "Nordkoreanische Won",
        "symbol": "KPW",
        "symbol-alt-narrow": "₩"
    },
    "KRH": {
        "displayName": "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-one": "Südkoreanischer Hwan (1953-1962)",
        "displayName-count-other": "Südkoreanischer Hwan (1953-1962)",
        "symbol": "KRH"
    },
    "KRO": {
        "displayName": "Südkoreanischer Won (1945-1953)",
        "displayName-count-one": "Südkoreanischer Won (1945-1953)",
        "displayName-count-other": "Südkoreanischer Won (1945-1953)",
        "symbol": "KRO"
    },
    "KRW": {
        "displayName": "Südkoreanischer Won",
        "displayName-count-one": "Südkoreanischer Won",
        "displayName-count-other": "Südkoreanische Won",
        "symbol": "₩",
        "symbol-alt-narrow": "₩"
    },
    "KWD": {
        "displayName": "Kuwait-Dinar",
        "displayName-count-one": "Kuwait-Dinar",
        "displayName-count-other": "Kuwait-Dinar",
        "symbol": "KWD"
    },
    "KYD": {
        "displayName": "Kaiman-Dollar",
        "displayName-count-one": "Kaiman-Dollar",
        "displayName-count-other": "Kaiman-Dollar",
        "symbol": "KYD",

```

```

        "symbol-alt-narrow": "$"
    },
    "KZT": {
        "displayName": "Kasachischer Tenge",
        "displayName-count-one": "Kasachischer Tenge",
        "displayName-count-other": "Kasachische Tenge",
        "symbol": "KZT",
        "symbol-alt-narrow": "T"
    },
    "LAK": {
        "displayName": "Laotischer Kip",
        "displayName-count-one": "Laotischer Kip",
        "displayName-count-other": "Laotische Kip",
        "symbol": "LAK",
        "symbol-alt-narrow": "₭"
    },
    "LBP": {
        "displayName": "Libanesisches Pfund",
        "displayName-count-one": "Libanesisches Pfund",
        "displayName-count-other": "Libanesische Pfund",
        "symbol": "LBP",
        "symbol-alt-narrow": "₡"
    },
    "LKR": {
        "displayName": "Sri-Lanka-Rupie",
        "displayName-count-one": "Sri-Lanka-Rupie",
        "displayName-count-other": "Sri-Lanka-Rupien",
        "symbol": "LKR",
        "symbol-alt-narrow": "Rs"
    },
    "LRD": {
        "displayName": "Liberianischer Dollar",
        "displayName-count-one": "Liberianischer Dollar",
        "displayName-count-other": "Liberianische Dollar",
        "symbol": "LRD",
        "symbol-alt-narrow": "$"
    },
    "LSL": {
        "displayName": "Loti",
        "displayName-count-one": "Loti",
        "displayName-count-other": "Loti",
        "symbol": "LSL"
    },
    "LTL": {
        "displayName": "Litauischer Litas",
        "displayName-count-one": "Litauischer Litas",
        "displayName-count-other": "Litauische Litas",
        "symbol": "LTL",
        "symbol-alt-narrow": "Lt"
    },
    "LTT": {
        "displayName": "Litauischer Talonas",
        "displayName-count-one": "Litauische Talonas",
        "displayName-count-other": "Litauische Talonas",
        "symbol": "LTT"
    },
    "LUC": {

```

```

        "displayName": "Luxemburgischer Franc (konvertibel)",
        "displayName-count-one": "Luxemburgische Franc (konvertibel)",
        "displayName-count-other": "Luxemburgische Franc (konvertibel)",
        "symbol": "LUC"
    },
    "LUF": {
        "displayName": "Luxemburgischer Franc",
        "displayName-count-one": "Luxemburgische Franc",
        "displayName-count-other": "Luxemburgische Franc",
        "symbol": "LUF"
    },
    "LUL": {
        "displayName": "Luxemburgischer Finanz-Franc",
        "displayName-count-one": "Luxemburgische Finanz-Franc",
        "displayName-count-other": "Luxemburgische Finanz-Franc",
        "symbol": "LUL"
    },
    "LVL": {
        "displayName": "Lettischer Lats",
        "displayName-count-one": "Lettischer Lats",
        "displayName-count-other": "Lettische Lats",
        "symbol": "LVL",
        "symbol-alt-narrow": "Ls"
    },
    "LVR": {
        "displayName": "Lettischer Rubel",
        "displayName-count-one": "Lettische Rubel",
        "displayName-count-other": "Lettische Rubel",
        "symbol": "LVR"
    },
    "LYD": {
        "displayName": "Libyscher Dinar",
        "displayName-count-one": "Libyscher Dinar",
        "displayName-count-other": "Libysche Dinar",
        "symbol": "LYD"
    },
    "MAD": {
        "displayName": "Marokkanischer Dirham",
        "displayName-count-one": "Marokkanischer Dirham",
        "displayName-count-other": "Marokkanische Dirham",
        "symbol": "MAD"
    },
    "MAF": {
        "displayName": "Marokkanischer Franc",
        "displayName-count-one": "Marokkanische Franc",
        "displayName-count-other": "Marokkanische Franc",
        "symbol": "MAF"
    },
    "MCF": {
        "displayName": "Monegassischer Franc",
        "displayName-count-one": "Monegassischer Franc",
        "displayName-count-other": "Monegassische Franc",
        "symbol": "MCF"
    },
    "MDC": {
        "displayName": "Moldau-Cupon",
        "displayName-count-one": "Moldau-Cupon",

```

```

        "displayName-count-other": "Moldau-Cupon",
        "symbol": "MDC"
    },
    "MDL": {
        "displayName": "Moldau-Leu",
        "displayName-count-one": "Moldau-Leu",
        "displayName-count-other": "Moldau-Leu",
        "symbol": "MDL"
    },
    "MGA": {
        "displayName": "Madagaskar-Ariary",
        "displayName-count-one": "Madagaskar-Ariary",
        "displayName-count-other": "Madagaskar-Ariary",
        "symbol": "MGA",
        "symbol-alt-narrow": "Ar"
    },
    "MGF": {
        "displayName": "Madagaskar-Franc",
        "displayName-count-one": "Madagaskar-Franc",
        "displayName-count-other": "Madagaskar-Franc",
        "symbol": "MGF"
    },
    "MKD": {
        "displayName": "Mazedonischer Denar",
        "displayName-count-one": "Mazedonischer Denar",
        "displayName-count-other": "Mazedonische Denari",
        "symbol": "MKD"
    },
    "MKN": {
        "displayName": "Mazedonischer Denar (1992-1993)",
        "displayName-count-one": "Mazedonischer Denar (1992-1993)",
        "displayName-count-other": "Mazedonische Denar (1992-1993)",
        "symbol": "MKN"
    },
    "MLF": {
        "displayName": "Malischer Franc",
        "displayName-count-one": "Malische Franc",
        "displayName-count-other": "Malische Franc",
        "symbol": "MLF"
    },
    "MMK": {
        "displayName": "Myanmarischer Kyat",
        "displayName-count-one": "Myanmarischer Kyat",
        "displayName-count-other": "Myanmarische Kyat",
        "symbol": "MMK",
        "symbol-alt-narrow": "K"
    },
    "MNT": {
        "displayName": "Mongolischer Tögrög",
        "displayName-count-one": "Mongolischer Tögrög",
        "displayName-count-other": "Mongolische Tögrög",
        "symbol": "MNT",
        "symbol-alt-narrow": "₮"
    },
    "MOP": {
        "displayName": "Macao-Pataca",
        "displayName-count-one": "Macao-Pataca",

```

```

        "displayName-count-other": "Macao-Pataca",
        "symbol": "MOP"
    },
    "MRO": {
        "displayName": "Mauretanischer Ouguiya",
        "displayName-count-one": "Mauretanischer Ouguiya",
        "displayName-count-other": "Mauretanische Ouguiya",
        "symbol": "MRO"
    },
    "MTL": {
        "displayName": "Maltesische Lira",
        "displayName-count-one": "Maltesische Lira",
        "displayName-count-other": "Maltesische Lira",
        "symbol": "MTL"
    },
    "MTP": {
        "displayName": "Maltesisches Pfund",
        "displayName-count-one": "Maltesische Pfund",
        "displayName-count-other": "Maltesische Pfund",
        "symbol": "MTP"
    },
    "MUR": {
        "displayName": "Mauritius-Rupie",
        "displayName-count-one": "Mauritius-Rupie",
        "displayName-count-other": "Mauritius-Rupien",
        "symbol": "MUR",
        "symbol-alt-narrow": "Rs"
    },
    "MVP": {
        "displayName": "Malediven-Rupie (alt)",
        "displayName-count-one": "Malediven-Rupie (alt)",
        "displayName-count-other": "Malediven-Rupien (alt)"
    },
    "MVR": {
        "displayName": "Malediven-Rufiyaa",
        "displayName-count-one": "Malediven-Rufiyaa",
        "displayName-count-other": "Malediven-Rupien",
        "symbol": "MVR"
    },
    "MWK": {
        "displayName": "Malawi-Kwacha",
        "displayName-count-one": "Malawi-Kwacha",
        "displayName-count-other": "Malawi-Kwacha",
        "symbol": "MWK"
    },
    "MXN": {
        "displayName": "Mexikanischer Peso",
        "displayName-count-one": "Mexikanischer Peso",
        "displayName-count-other": "Mexikanische Pesos",
        "symbol": "MX$",
        "symbol-alt-narrow": "$"
    },
    "MXP": {
        "displayName": "Mexikanischer Silber-Peso (1861-1992)",
        "displayName-count-one": "Mexikanische Silber-Peso (1861-1992)",
        "displayName-count-other": "Mexikanische Silber-Pesos (1861-
1992)",

```

```

        "symbol": "MXP"
      },
      "MXV": {
        "displayName": "Mexicanischer Unidad de Inversion (UDI)",
        "displayName-count-one": "Mexicanischer Unidad de Inversion
(UDI) ",
        "displayName-count-other": "Mexikanische Unidad de Inversion
(UDI) ",
        "symbol": "MXV"
      },
      "MYR": {
        "displayName": "Malaysischer Ringgit",
        "displayName-count-one": "Malaysischer Ringgit",
        "displayName-count-other": "Malaysische Ringgit",
        "symbol": "MYR",
        "symbol-alt-narrow": "RM"
      },
      "MZE": {
        "displayName": "Mosambikanischer Escudo",
        "displayName-count-one": "Mozambikanische Escudo",
        "displayName-count-other": "Mozambikanische Escudo",
        "symbol": "MZE"
      },
      "MZM": {
        "displayName": "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one": "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other": "Mosambikanische Meticais (1980-
2006) ",
        "symbol": "MZM"
      },
      "MZN": {
        "displayName": "Mosambikanischer Metical",
        "displayName-count-one": "Mosambikanischer Metical",
        "displayName-count-other": "Mosambikanische Meticais",
        "symbol": "MZN"
      },
      "NAD": {
        "displayName": "Namibia-Dollar",
        "displayName-count-one": "Namibia-Dollar",
        "displayName-count-other": "Namibia-Dollar",
        "symbol": "NAD",
        "symbol-alt-narrow": "$"
      },
      "NGN": {
        "displayName": "Nigerianischer Naira",
        "displayName-count-one": "Nigerianischer Naira",
        "displayName-count-other": "Nigerianische Naira",
        "symbol": "NGN",
        "symbol-alt-narrow": "₦"
      },
      "NIC": {
        "displayName": "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-one": "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-other": "Nicaraguanische Córdoba (1988-
1991) ",
        "symbol": "NIC"
      },
    },
  },

```



```
"NIO": {
  "displayName": "Nicaragua-Córdoba",
  "displayName-count-one": "Nicaragua-Córdoba",
  "displayName-count-other": "Nicaragua-Córdobas",
  "symbol": "NIO",
  "symbol-alt-narrow": "C$"
},
"NLG": {
  "displayName": "Niederländischer Gulden",
  "displayName-count-one": "Niederländischer Gulden",
  "displayName-count-other": "Niederländische Gulden",
  "symbol": "NLG"
},
"NOK": {
  "displayName": "Norwegische Krone",
  "displayName-count-one": "Norwegische Krone",
  "displayName-count-other": "Norwegische Kronen",
  "symbol": "NOK",
  "symbol-alt-narrow": "kr"
},
"NPR": {
  "displayName": "Nepalesische Rupie",
  "displayName-count-one": "Nepalesische Rupie",
  "displayName-count-other": "Nepalesische Rupien",
  "symbol": "NPR",
  "symbol-alt-narrow": "Rs"
},
"NZD": {
  "displayName": "Neuseeland-Dollar",
  "displayName-count-one": "Neuseeland-Dollar",
  "displayName-count-other": "Neuseeland-Dollar",
  "symbol": "NZ$",
  "symbol-alt-narrow": "$"
},
"OMR": {
  "displayName": "Omanischer Rial",
  "displayName-count-one": "Omanischer Rial",
  "displayName-count-other": "Omanische Rials",
  "symbol": "OMR"
},
"PAB": {
  "displayName": "Panamaischer Balboa",
  "displayName-count-one": "Panamaischer Balboa",
  "displayName-count-other": "Panamaische Balboas",
  "symbol": "PAB"
},
"PEI": {
  "displayName": "Peruanischer Inti",
  "displayName-count-one": "Peruanische Inti",
  "displayName-count-other": "Peruanische Inti",
  "symbol": "PEI"
},
"PEN": {
  "displayName": "Peruanischer Sol",
  "displayName-count-one": "Peruanischer Sol",
  "displayName-count-other": "Peruanische Sol",
  "symbol": "PEN"
}
```

```

    },
    "PES": {
      "displayName": "Peruanischer Sol (1863-1965)",
      "displayName-count-one": "Peruanischer Sol (1863-1965)",
      "displayName-count-other": "Peruanische Sol (1863-1965)",
      "symbol": "PES"
    },
    "PGK": {
      "displayName": "Papua-Neuguineischer Kina",
      "displayName-count-one": "Papua-Neuguineischer Kina",
      "displayName-count-other": "Papua-Neuguineische Kina",
      "symbol": "PGK"
    },
    "PHP": {
      "displayName": "Philippinischer Peso",
      "displayName-count-one": "Philippinischer Peso",
      "displayName-count-other": "Philippinische Pesos",
      "symbol": "PHP",
      "symbol-alt-narrow": "₱"
    },
    "PKR": {
      "displayName": "Pakistanische Rupie",
      "displayName-count-one": "Pakistanische Rupie",
      "displayName-count-other": "Pakistanische Rupien",
      "symbol": "PKR",
      "symbol-alt-narrow": "Rs"
    },
    "PLN": {
      "displayName": "Polnischer Złoty",
      "displayName-count-one": "Polnischer Złoty",
      "displayName-count-other": "Polnische Złoty",
      "symbol": "PLN",
      "symbol-alt-narrow": "zł"
    },
    "PLZ": {
      "displayName": "Polnischer Zloty (1950-1995)",
      "displayName-count-one": "Polnischer Zloty (1950-1995)",
      "displayName-count-other": "Polnische Zloty (1950-1995)",
      "symbol": "PLZ"
    },
    "PTE": {
      "displayName": "Portugiesischer Escudo",
      "displayName-count-one": "Portugiesische Escudo",
      "displayName-count-other": "Portugiesische Escudo",
      "symbol": "PTE"
    },
    "PYG": {
      "displayName": "Paraguayischer Guaraní",
      "displayName-count-one": "Paraguayischer Guaraní",
      "displayName-count-other": "Paraguayische Guaraníes",
      "symbol": "PYG",
      "symbol-alt-narrow": "₲"
    },
    "QAR": {
      "displayName": "Katar-Riyal",
      "displayName-count-one": "Katar-Riyal",
      "displayName-count-other": "Katar-Riyal",

```

```

        "symbol": "QAR"
    },
    "RHD": {
        "displayName": "Rhodesischer Dollar",
        "displayName-count-one": "Rhodesische Dollar",
        "displayName-count-other": "Rhodesische Dollar",
        "symbol": "RHD"
    },
    "ROL": {
        "displayName": "Rumänischer Leu (1952-2006)",
        "displayName-count-one": "Rumänischer Leu (1952-2006)",
        "displayName-count-other": "Rumänische Leu (1952-2006)",
        "symbol": "ROL"
    },
    "RON": {
        "displayName": "Rumänischer Leu",
        "displayName-count-one": "Rumänischer Leu",
        "displayName-count-other": "Rumänische Leu",
        "symbol": "RON",
        "symbol-alt-narrow": "L"
    },
    "RSD": {
        "displayName": "Serbischer Dinar",
        "displayName-count-one": "Serbischer Dinar",
        "displayName-count-other": "Serbische Dinaren",
        "symbol": "RSD"
    },
    "RUB": {
        "displayName": "Russischer Rubel",
        "displayName-count-one": "Russischer Rubel",
        "displayName-count-other": "Russische Rubel",
        "symbol": "RUB",
        "symbol-alt-narrow": "₽"
    },
    "RUR": {
        "displayName": "Russischer Rubel (1991-1998)",
        "displayName-count-one": "Russischer Rubel (1991-1998)",
        "displayName-count-other": "Russische Rubel (1991-1998)",
        "symbol": "RUR",
        "symbol-alt-narrow": "p."
    },
    "RWF": {
        "displayName": "Ruanda-Franc",
        "displayName-count-one": "Ruanda-Franc",
        "displayName-count-other": "Ruanda-Francs",
        "symbol": "RWF",
        "symbol-alt-narrow": "F.Rw"
    },
    "SAR": {
        "displayName": "Saudi-Rial",
        "displayName-count-one": "Saudi-Rial",
        "displayName-count-other": "Saudi-Rial",
        "symbol": "SAR"
    },
    "SBD": {
        "displayName": "Salomonen-Dollar",
        "displayName-count-one": "Salomonen-Dollar",

```

```

        "displayName-count-other": "Salomonen-Dollar",
        "symbol": "SBD",
        "symbol-alt-narrow": "$"
    },
    "SCR": {
        "displayName": "Seychellen-Rupie",
        "displayName-count-one": "Seychellen-Rupie",
        "displayName-count-other": "Seychellen-Rupien",
        "symbol": "SCR"
    },
    "SDD": {
        "displayName": "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one": "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other": "Sudanesische Dinar (1992-2007)",
        "symbol": "SDD"
    },
    "SDG": {
        "displayName": "Sudanesisches Pfund",
        "displayName-count-one": "Sudanesisches Pfund",
        "displayName-count-other": "Sudanesische Pfund",
        "symbol": "SDG"
    },
    "SDP": {
        "displayName": "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one": "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other": "Sudanesische Pfund (1957-1998)",
        "symbol": "SDP"
    },
    "SEK": {
        "displayName": "Schwedische Krone",
        "displayName-count-one": "Schwedische Krone",
        "displayName-count-other": "Schwedische Kronen",
        "symbol": "SEK",
        "symbol-alt-narrow": "kr"
    },
    "SGD": {
        "displayName": "Singapur-Dollar",
        "displayName-count-one": "Singapur-Dollar",
        "displayName-count-other": "Singapur-Dollar",
        "symbol": "SGD",
        "symbol-alt-narrow": "$"
    },
    "SHP": {
        "displayName": "St. Helena-Pfund",
        "displayName-count-one": "St. Helena-Pfund",
        "displayName-count-other": "St. Helena-Pfund",
        "symbol": "SHP",
        "symbol-alt-narrow": "£"
    },
    "SIT": {
        "displayName": "Slowenischer Tolar",
        "displayName-count-one": "Slowenischer Tolar",
        "displayName-count-other": "Slowenische Tolar",
        "symbol": "SIT"
    },
    "SKK": {
        "displayName": "Slowakische Krone",

```

```

        "displayName-count-one": "Slowakische Kronen",
        "displayName-count-other": "Slowakische Kronen",
        "symbol": "SKK"
    },
    "SLL": {
        "displayName": "Sierra-leonischer Leone",
        "displayName-count-one": "Sierra-leonischer Leone",
        "displayName-count-other": "Sierra-leonische Leones",
        "symbol": "SLL"
    },
    "SOS": {
        "displayName": "Somalia-Schilling",
        "displayName-count-one": "Somalia-Schilling",
        "displayName-count-other": "Somalia-Schilling",
        "symbol": "SOS"
    },
    "SRD": {
        "displayName": "Suriname-Dollar",
        "displayName-count-one": "Suriname-Dollar",
        "displayName-count-other": "Suriname-Dollar",
        "symbol": "SRD",
        "symbol-alt-narrow": "$"
    },
    "SRG": {
        "displayName": "Suriname Gulden",
        "displayName-count-one": "Suriname-Gulden",
        "displayName-count-other": "Suriname-Gulden",
        "symbol": "SRG"
    },
    "SSP": {
        "displayName": "Südsudanesisches Pfund",
        "displayName-count-one": "Südsudanesisches Pfund",
        "displayName-count-other": "Südsudanesische Pfund",
        "symbol": "SSP",
        "symbol-alt-narrow": "£"
    },
    "STD": {
        "displayName": "São-toméischer Dobra",
        "displayName-count-one": "São-toméischer Dobra",
        "displayName-count-other": "São-toméische Dobra",
        "symbol": "STD",
        "symbol-alt-narrow": "Db"
    },
    "SUR": {
        "displayName": "Sowjetischer Rubel",
        "displayName-count-one": "Sowjetische Rubel",
        "displayName-count-other": "Sowjetische Rubel",
        "symbol": "SUR"
    },
    "SVC": {
        "displayName": "El Salvador Colon",
        "displayName-count-one": "El Salvador-Colon",
        "displayName-count-other": "El Salvador-Colon",
        "symbol": "SVC"
    },
    "SYP": {
        "displayName": "Syrisches Pfund",

```

```

        "displayName-count-one": "Syrisches Pfund",
        "displayName-count-other": "Syrische Pfund",
        "symbol": "SYP",
        "symbol-alt-narrow": "SYP"
    },
    "SZL": {
        "displayName": "Swasiländischer Lilangeni",
        "displayName-count-one": "Swasiländischer Lilangeni",
        "displayName-count-other": "Swasiländische Emalangeni",
        "symbol": "SZL"
    },
    "THB": {
        "displayName": "Thailändischer Baht",
        "displayName-count-one": "Thailändischer Baht",
        "displayName-count-other": "Thailändische Baht",
        "symbol": "฿",
        "symbol-alt-narrow": "฿"
    },
    "TJR": {
        "displayName": "Tadschikistan Rubel",
        "displayName-count-one": "Tadschikistan-Rubel",
        "displayName-count-other": "Tadschikistan-Rubel",
        "symbol": "TJR"
    },
    "TJS": {
        "displayName": "Tadschikistan-Somoni",
        "displayName-count-one": "Tadschikistan-Somoni",
        "displayName-count-other": "Tadschikistan-Somoni",
        "symbol": "TJS"
    },
    "TMM": {
        "displayName": "Turkmenistan-Manat (1993-2009)",
        "displayName-count-one": "Turkmenistan-Manat (1993-2009)",
        "displayName-count-other": "Turkmenistan-Manat (1993-2009)",
        "symbol": "TMM"
    },
    "TMT": {
        "displayName": "Turkmenistan-Manat",
        "displayName-count-one": "Turkmenistan-Manat",
        "displayName-count-other": "Turkmenistan-Manat",
        "symbol": "TMT"
    },
    "TND": {
        "displayName": "Tunesischer Dinar",
        "displayName-count-one": "Tunesischer Dinar",
        "displayName-count-other": "Tunesische Dinar",
        "symbol": "TND"
    },
    "TOP": {
        "displayName": "Tongaischer Paʻanga",
        "displayName-count-one": "Tongaischer Paʻanga",
        "displayName-count-other": "Tongaische Paʻanga",
        "symbol": "TOP",
        "symbol-alt-narrow": "T$"
    },
    "TPE": {
        "displayName": "Timor-Escudo",

```

```

        "displayName-count-one": "Timor-Escudo",
        "displayName-count-other": "Timor-Escudo",
        "symbol": "TPE"
    },
    "TRL": {
        "displayName": "Türkische Lira (1922-2005)",
        "displayName-count-one": "Türkische Lira (1922-2005)",
        "displayName-count-other": "Türkische Lira (1922-2005)",
        "symbol": "TRL"
    },
    "TRY": {
        "displayName": "Türkische Lira",
        "displayName-count-one": "Türkische Lira",
        "displayName-count-other": "Türkische Lira",
        "symbol": "TRY",
        "symbol-alt-narrow": "₺",
        "symbol-alt-variant": "TL"
    },
    "TTD": {
        "displayName": "Trinidad und Tobago-Dollar",
        "displayName-count-one": "Trinidad und Tobago-Dollar",
        "displayName-count-other": "Trinidad und Tobago-Dollar",
        "symbol": "TTD",
        "symbol-alt-narrow": "$"
    },
    "TWD": {
        "displayName": "Neuer Taiwan-Dollar",
        "displayName-count-one": "Neuer Taiwan-Dollar",
        "displayName-count-other": "Neue Taiwan-Dollar",
        "symbol": "NT$",
        "symbol-alt-narrow": "NT$"
    },
    "TZS": {
        "displayName": "Tansania-Schilling",
        "displayName-count-one": "Tansania-Schilling",
        "displayName-count-other": "Tansania-Schilling",
        "symbol": "TZS"
    },
    "UAH": {
        "displayName": "Ukrainische Hrywnja",
        "displayName-count-one": "Ukrainische Hrywnja",
        "displayName-count-other": "Ukrainische Hrywen",
        "symbol": "UAH",
        "symbol-alt-narrow": "₴"
    },
    "UAK": {
        "displayName": "Ukrainischer Karbovanetz",
        "displayName-count-one": "Ukrainische Karbovanetz",
        "displayName-count-other": "Ukrainische Karbovanetz",
        "symbol": "UAK"
    },
    "UGS": {
        "displayName": "Uganda-Schilling (1966-1987)",
        "displayName-count-one": "Uganda-Schilling (1966-1987)",
        "displayName-count-other": "Uganda-Schilling (1966-1987)",
        "symbol": "UGS"
    },
    },

```

```

    "UGX": {
      "displayName": "Uganda-Schilling",
      "displayName-count-one": "Uganda-Schilling",
      "displayName-count-other": "Uganda-Schilling",
      "symbol": "UGX"
    },
    "USD": {
      "displayName": "US-Dollar",
      "displayName-count-one": "US-Dollar",
      "displayName-count-other": "US-Dollar",
      "symbol": "$",
      "symbol-alt-narrow": "$"
    },
    "USN": {
      "displayName": "US Dollar (Nächster Tag)",
      "displayName-count-one": "US-Dollar (Nächster Tag)",
      "displayName-count-other": "US-Dollar (Nächster Tag)",
      "symbol": "USN"
    },
    "USS": {
      "displayName": "US Dollar (Gleicher Tag)",
      "displayName-count-one": "US-Dollar (Gleicher Tag)",
      "displayName-count-other": "US-Dollar (Gleicher Tag)",
      "symbol": "USS"
    },
    "UYI": {
      "displayName": "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
      "displayName-count-one": "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
      "displayName-count-other": "Uruguayische Pesos (Indexierte
Rechnungseinheiten)",
      "symbol": "UYI"
    },
    "UYP": {
      "displayName": "Uruguayischer Peso (1975-1993)",
      "displayName-count-one": "Uruguayischer Peso (1975-1993)",
      "displayName-count-other": "Uruguayische Pesos (1975-1993)",
      "symbol": "UYP"
    },
    "UYU": {
      "displayName": "Uruguayischer Peso",
      "displayName-count-one": "Uruguayischer Peso",
      "displayName-count-other": "Uruguayische Pesos",
      "symbol": "UYU",
      "symbol-alt-narrow": "$"
    },
    "UZS": {
      "displayName": "Usbekistan-Sum",
      "displayName-count-one": "Usbekistan-Sum",
      "displayName-count-other": "Usbekistan-Sum",
      "symbol": "UZS"
    },
    "VEB": {
      "displayName": "Venezolanischer Bolívar (1871-2008)",
      "displayName-count-one": "Venezolanischer Bolívar (1871-2008)",

```



```

        "displayName-count-other": "Venezolanische Bolívares (1871-2008)",
        "symbol": "VEB"
    },
    "VEF": {
        "displayName": "Venezolanischer Bolívar",
        "displayName-count-one": "Venezolanischer Bolívar",
        "displayName-count-other": "Venezolanische Bolívares",
        "symbol": "VEF",
        "symbol-alt-narrow": "Bs"
    },
    "VND": {
        "displayName": "Vietnamesischer Dong",
        "displayName-count-one": "Vietnamesischer Dong",
        "displayName-count-other": "Vietnamesische Dong",
        "symbol": "₫",
        "symbol-alt-narrow": "₫"
    },
    "VNN": {
        "displayName": "Vietnamesischer Dong(1978-1985)",
        "displayName-count-one": "Vietnamesischer Dong(1978-1985)",
        "displayName-count-other": "Vietnamesische Dong(1978-1985)",
        "symbol": "VNN"
    },
    "VUV": {
        "displayName": "Vanuatu-Vatu",
        "displayName-count-one": "Vanuatu-Vatu",
        "displayName-count-other": "Vanuatu-Vatu",
        "symbol": "VUV"
    },
    "WST": {
        "displayName": "Samoanischer Tala",
        "displayName-count-one": "Samoanischer Tala",
        "displayName-count-other": "Samoanische Tala",
        "symbol": "WST"
    },
    "XAF": {
        "displayName": "CFA-Franc (BEAC)",
        "displayName-count-one": "CFA-Franc (BEAC)",
        "displayName-count-other": "CFA-Franc (BEAC)",
        "symbol": "FCFA"
    },
    "XAG": {
        "displayName": "Unze Silber",
        "displayName-count-one": "Unze Silber",
        "displayName-count-other": "Unzen Silber",
        "symbol": "XAG"
    },
    "XAU": {
        "displayName": "Unze Gold",
        "displayName-count-one": "Unze Gold",
        "displayName-count-other": "Unzen Gold",
        "symbol": "XAU"
    },
    "XBA": {
        "displayName": "Europäische Rechnungseinheit",
        "displayName-count-one": "Europäische Rechnungseinheiten",

```

```

        "displayName-count-other": "Europäische Rechnungseinheiten",
        "symbol": "XBA"
    },
    "XBB": {
        "displayName": "Europäische Währungseinheit (XBB)",
        "displayName-count-one": "Europäische Währungseinheiten (XBB)",
        "displayName-count-other": "Europäische Währungseinheiten
(XBB) ",
        "symbol": "XBB"
    },
    "XBC": {
        "displayName": "Europäische Rechnungseinheit (XBC)",
        "displayName-count-one": "Europäische Rechnungseinheiten (XBC)",
        "displayName-count-other": "Europäische Rechnungseinheiten
(XBC) ",
        "symbol": "XBC"
    },
    "XBD": {
        "displayName": "Europäische Rechnungseinheit (XBD)",
        "displayName-count-one": "Europäische Rechnungseinheiten (XBD)",
        "displayName-count-other": "Europäische Rechnungseinheiten
(XBD) ",
        "symbol": "XBD"
    },
    "XCD": {
        "displayName": "Ostkaribischer Dollar",
        "displayName-count-one": "Ostkaribischer Dollar",
        "displayName-count-other": "Ostkaribische Dollar",
        "symbol": "EC$",
        "symbol-alt-narrow": "$"
    },
    "XDR": {
        "displayName": "Sonderziehungsrechte",
        "displayName-count-one": "Sonderziehungsrechte",
        "displayName-count-other": "Sonderziehungsrechte",
        "symbol": "XDR"
    },
    "XEU": {
        "displayName": "Europäische Währungseinheit (XEU)",
        "displayName-count-one": "Europäische Währungseinheiten (XEU)",
        "displayName-count-other": "Europäische Währungseinheiten
(XEU) ",
        "symbol": "XEU"
    },
    "XFO": {
        "displayName": "Französischer Gold-Franc",
        "displayName-count-one": "Französische Gold-Franc",
        "displayName-count-other": "Französische Gold-Franc",
        "symbol": "XFO"
    },
    "XFU": {
        "displayName": "Französischer UIC-Franc",
        "displayName-count-one": "Französische UIC-Franc",
        "displayName-count-other": "Französische UIC-Franc",
        "symbol": "XFU"
    },
    "XOF": {

```

```

    "displayName": "CFA-Franc (BCEAO)",
    "displayName-count-one": "CFA-Franc (BCEAO)",
    "displayName-count-other": "CFA-Francs (BCEAO)",
    "symbol": "CFA"
  },
  "XPD": {
    "displayName": "Unze Palladium",
    "displayName-count-one": "Unze Palladium",
    "displayName-count-other": "Unzen Palladium",
    "symbol": "XPD"
  },
  "XPF": {
    "displayName": "CFP-Franc",
    "displayName-count-one": "CFP-Franc",
    "displayName-count-other": "CFP-Franc",
    "symbol": "CFPF"
  },
  "XPT": {
    "displayName": "Unze Platin",
    "displayName-count-one": "Unze Platin",
    "displayName-count-other": "Unzen Platin",
    "symbol": "XPT"
  },
  "XRE": {
    "displayName": "RINET Funds",
    "displayName-count-one": "RINET Funds",
    "displayName-count-other": "RINET Funds",
    "symbol": "XRE"
  },
  "XSU": {
    "displayName": "SUCRE",
    "displayName-count-one": "SUCRE",
    "displayName-count-other": "SUCRE",
    "symbol": "XSU"
  },
  "XTS": {
    "displayName": "Testwährung",
    "displayName-count-one": "Testwährung",
    "displayName-count-other": "Testwährung",
    "symbol": "XTS"
  },
  "XUA": {
    "displayName": "Rechnungseinheit der AfEB",
    "displayName-count-one": "Rechnungseinheit der AfEB",
    "displayName-count-other": "Rechnungseinheiten der AfEB",
    "symbol": "XUA"
  },
  "XXX": {
    "displayName": "Unbekannte Währung",
    "displayName-count-one": "(unbekannte Währung)",
    "displayName-count-other": "(unbekannte Währung)",
    "symbol": "XXX"
  },
  "YDD": {
    "displayName": "Jemen-Dinar",
    "displayName-count-one": "Jemen-Dinar",
    "displayName-count-other": "Jemen-Dinar",

```

```

        "symbol": "YDD"
    },
    "YER": {
        "displayName": "Jemen-Rial",
        "displayName-count-one": "Jemen-Rial",
        "displayName-count-other": "Jemen-Rial",
        "symbol": "YER"
    },
    "YUD": {
        "displayName": "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-one": "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-other": "Jugoslawische Dinar (1966-1990)",
        "symbol": "YUD"
    },
    "YUM": {
        "displayName": "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-one": "Jugoslawischer Neuer Dinar (1994-
2002) ",
        "displayName-count-other": "Jugoslawische Neue Dinar (1994-
2002) ",
        "symbol": "YUM"
    },
    "YUN": {
        "displayName": "Jugoslawischer Dinar (konvertibel)",
        "displayName-count-one": "Jugoslawische Dinar (konvertibel)",
        "displayName-count-other": "Jugoslawische Dinar (konvertibel)",
        "symbol": "YUN"
    },
    "YUR": {
        "displayName": "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-one": "Jugoslawischer reformierter Dinar
(1992-1993) ",
        "displayName-count-other": "Jugoslawische reformierte Dinar
(1992-1993) ",
        "symbol": "YUR"
    },
    "ZAL": {
        "displayName": "Südafrikanischer Rand (Finanz)",
        "displayName-count-one": "Südafrikanischer Rand (Finanz)",
        "displayName-count-other": "Südafrikanischer Rand (Finanz)",
        "symbol": "ZAL"
    },
    "ZAR": {
        "displayName": "Südafrikanischer Rand",
        "displayName-count-one": "Südafrikanischer Rand",
        "displayName-count-other": "Südafrikanische Rand",
        "symbol": "ZAR",
        "symbol-alt-narrow": "R"
    },
    "ZMK": {
        "displayName": "Kwacha (1968-2012)",
        "displayName-count-one": "Kwacha (1968-2012)",
        "displayName-count-other": "Kwacha (1968-2012)",
        "symbol": "ZMK"
    },
    "ZMW": {
        "displayName": "Kwacha",

```

```

    "displayName-count-one": "Kwacha",
    "displayName-count-other": "Kwacha",
    "symbol": "ZMW",
    "symbol-alt-narrow": "K"
  },
  "ZRN": {
    "displayName": "Zaire-Neuer Zaire (1993-1998)",
    "displayName-count-one": "Zaire-Neuer Zaire (1993-1998)",
    "displayName-count-other": "Zaire-Neue Zaire (1993-1998)",
    "symbol": "ZRN"
  },
  "ZRZ": {
    "displayName": "Zaire-Zaire (1971-1993)",
    "displayName-count-one": "Zaire-Zaire (1971-1993)",
    "displayName-count-other": "Zaire-Zaire (1971-1993)",
    "symbol": "ZRZ"
  },
  "ZWD": {
    "displayName": "Simbabwe-Dollar (1980-2008)",
    "displayName-count-one": "Simbabwe-Dollar (1980-2008)",
    "displayName-count-other": "Simbabwe-Dollar (1980-2008)",
    "symbol": "ZWD"
  },
  "ZWL": {
    "displayName": "Simbabwe-Dollar (2009)",
    "displayName-count-one": "Simbabwe-Dollar (2009)",
    "displayName-count-other": "Simbabwe-Dollar (2009)",
    "symbol": "ZWL"
  },
  "ZWR": {
    "displayName": "Simbabwe-Dollar (2008)",
    "displayName-count-one": "Simbabwe-Dollar (2008)",
    "displayName-count-other": "Simbabwe-Dollar (2008)",
    "symbol": "ZWR"
  }
}

```

CURRENCIES.JSX

```
{  
    "main";  
    {  
        "de";  
        {  
            "identity";  
            {  
                "version";  
                {  
                    "_number";  
                    "$Revision: 13259 $",  
                        "_cldrVersion";  
                        "31";
```

```

    }
    "language";
    "de";
  }
  "numbers";
  {
    "currencies";
    {
      "ADP";
      {
        "displayName";
        "Andorranische Pesete",
        "displayName-count-one";
        "Andorranische Pesete",
        "displayName-count-other";
        "Andorranische Peseten",
        "symbol";
        "ADP";
      }
      "AED";
      {
        "displayName";
        "VAE-Dirham",
        "displayName-count-one";
        "VAE-Dirham",
        "displayName-count-other";
        "VAE-Dirham",
        "symbol";
        "AED";
      }
      "AFA";
      {
        "displayName";
        "Afghanische Afghani (1927-2002)",
        "displayName-count-one";
        "Afghanische Afghani (1927-2002)",
        "displayName-count-other";
        "Afghanische Afghani (1927-2002)",
        "symbol";
        "AFA";
      }
      "AFN";
      {
        "displayName";
        "Afghanischer Afghani",
        "displayName-count-one";
        "Afghanischer Afghani",
        "displayName-count-other";
        "Afghanische Afghani",
        "symbol";
        "AFN";
      }
      "ALK";
      {
        "displayName";
        "Albanischer Lek (1946-1965)",
        "displayName-count-one";

```

```

        "Albanischer Lek (1946-1965)",
        "displayName-count-other";
        "Albanische Lek (1946-1965)";
    }
    "ALL";
    {
        "displayName";
        "Albanischer Lek",
        "displayName-count-one";
        "Albanischer Lek",
        "displayName-count-other";
        "Albanische Lek",
        "symbol";
        "ALL";
    }
    "AMD";
    {
        "displayName";
        "Armenischer Dram",
        "displayName-count-one";
        "Armenischer Dram",
        "displayName-count-other";
        "Armenische Dram",
        "symbol";
        "AMD";
    }
    "ANG";
    {
        "displayName";
        "Niederländische-Antillen-Gulden",
        "displayName-count-one";
        "Niederländische-Antillen-Gulden",
        "displayName-count-other";
        "Niederländische-Antillen-Gulden",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "Angolanischer Kwanza",
        "displayName-count-one";
        "Angolanischer Kwanza",
        "displayName-count-other";
        "Angolanische Kwanza",
        "symbol";
        "AOA",
        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-one";
        "Angolanischer Kwanza (1977-1990)",
        "displayName-count-other";
    }

```

```

        "Angolanische Kwanza (1977-1990)",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-one";
        "Angolanischer Neuer Kwanza (1990-2000)",
        "displayName-count-other";
        "Angolanische Neue Kwanza (1990-2000)",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-one";
        "Angolanischer Kwanza Reajustado (1995-1999)",
        "displayName-count-other";
        "Angolanische Kwanza Reajustado (1995-1999)",
        "symbol";
        "AOR";
    }
    "ARA";
    {
        "displayName";
        "Argentinischer Austral",
        "displayName-count-one";
        "Argentinischer Austral",
        "displayName-count-other";
        "Argentinische Austral",
        "symbol";
        "ARA";
    }
    "ARL";
    {
        "displayName";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-one";
        "Argentinischer Peso Ley (1970-1983)",
        "displayName-count-other";
        "Argentinische Pesos Ley (1970-1983)",
        "symbol";
        "ARL";
    }
    "ARM";
    {
        "displayName";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-one";
        "Argentinischer Peso (1881-1970)",
        "displayName-count-other";
        "Argentinische Pesos (1881-1970)",
        "symbol";
    }

```



```

        "ARM";
    }
    "ARP";
    {
        "displayName";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-one";
        "Argentinischer Peso (1983-1985)",
        "displayName-count-other";
        "Argentinische Peso (1983-1985)",
        "symbol";
        "ARP";
    }
    "ARS";
    {
        "displayName";
        "Argentinischer Peso",
        "displayName-count-one";
        "Argentinischer Peso",
        "displayName-count-other";
        "Argentinische Pesos",
        "symbol";
        "ARS",
        "symbol-alt-narrow";
        "$";
    }
    "ATS";
    {
        "displayName";
        "Österreichischer Schilling",
        "displayName-count-one";
        "Österreichischer Schilling",
        "displayName-count-other";
        "Österreichische Schilling",
        "symbol";
        "öS";
    }
    "AUD";
    {
        "displayName";
        "Australischer Dollar",
        "displayName-count-one";
        "Australischer Dollar",
        "displayName-count-other";
        "Australische Dollar",
        "symbol";
        "AU$",
        "symbol-alt-narrow";
        "$";
    }
    "AWG";
    {
        "displayName";
        "Aruba-Florin",
        "displayName-count-one";
        "Aruba-Florin",
        "displayName-count-other";
    }

```

```

        "Aruba-Florin",
        "symbol";
        "AWG";
    }
    "AZM";
    {
        "displayName";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-one";
        "Aserbaidtschan-Manat (1993-2006)",
        "displayName-count-other";
        "Aserbaidtschan-Manat (1993-2006)",
        "symbol";
        "AZM";
    }
    "AZN";
    {
        "displayName";
        "Aserbaidtschan-Manat",
        "displayName-count-one";
        "Aserbaidtschan-Manat",
        "displayName-count-other";
        "Aserbaidtschan-Manat",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-one";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "displayName-count-other";
        "Bosnien und Herzegowina Dinar (1992-1994)",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-one";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "displayName-count-other";
        "Bosnien und Herzegowina Konvertierbare Mark",
        "symbol";
        "BAM",
        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-one";
        "Bosnien und Herzegowina Neuer Dinar (1994-1997)",
        "displayName-count-other";
    }

```

```

        "Bosnien und Herzegowina Neue Dinar (1994-1997)",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "Barbados-Dollar",
        "displayName-count-one";
        "Barbados-Dollar",
        "displayName-count-other";
        "Barbados-Dollar",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "$";
    }
    "BDT";
    {
        "displayName";
        "Bangladesch-Taka",
        "displayName-count-one";
        "Bangladesch-Taka",
        "displayName-count-other";
        "Bangladesch-Taka",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }
    "BEC";
    {
        "displayName";
        "Belgischer Franc (konvertibel)",
        "displayName-count-one";
        "Belgischer Franc (konvertibel)",
        "displayName-count-other";
        "Belgische Franc (konvertibel)",
        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "Belgischer Franc",
        "displayName-count-one";
        "Belgischer Franc",
        "displayName-count-other";
        "Belgische Franc",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "Belgischer Finanz-Franc",

```

```

        "displayName-count-one";
        "Belgischer Finanz-Franc",
        "displayName-count-other";
        "Belgische Finanz-Franc",
        "symbol";
        "BEL";
    }
    "BGL";
    {
        "displayName";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-one";
        "Bulgarische Lew (1962-1999)",
        "displayName-count-other";
        "Bulgarische Lew (1962-1999)",
        "symbol";
        "BGL";
    }
    "BGM";
    {
        "displayName";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-one";
        "Bulgarischer Lew (1952-1962)",
        "displayName-count-other";
        "Bulgarische Lew (1952-1962)",
        "symbol";
        "BGK";
    }
    "BGN";
    {
        "displayName";
        "Bulgarischer Lew",
        "displayName-count-one";
        "Bulgarischer Lew",
        "displayName-count-other";
        "Bulgarische Lew",
        "symbol";
        "BGN";
    }
    "BGO";
    {
        "displayName";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-one";
        "Bulgarischer Lew (1879-1952)",
        "displayName-count-other";
        "Bulgarische Lew (1879-1952)",
        "symbol";
        "BGJ";
    }
    "BHD";
    {
        "displayName";
        "Bahrain-Dinar",
        "displayName-count-one";
        "Bahrain-Dinar",

```

```

        "displayName-count-other";
        "Bahrain-Dinar",
        "symbol";
        "BHD";
    }
    "BIF";
    {
        "displayName";
        "Burundi-Franc",
        "displayName-count-one";
        "Burundi-Franc",
        "displayName-count-other";
        "Burundi-Francs",
        "symbol";
        "BIF";
    }
    "BMD";
    {
        "displayName";
        "Bermuda-Dollar",
        "displayName-count-one";
        "Bermuda-Dollar",
        "displayName-count-other";
        "Bermuda-Dollar",
        "symbol";
        "BMD",
        "symbol-alt-narrow";
        "$";
    }
    "BND";
    {
        "displayName";
        "Brunei-Dollar",
        "displayName-count-one";
        "Brunei-Dollar",
        "displayName-count-other";
        "Brunei-Dollar",
        "symbol";
        "BND",
        "symbol-alt-narrow";
        "$";
    }
    "BOB";
    {
        "displayName";
        "Bolivanischer Boliviano",
        "displayName-count-one";
        "Bolivanischer Boliviano",
        "displayName-count-other";
        "Bolivianische Bolivianos",
        "symbol";
        "BOB",
        "symbol-alt-narrow";
        "Bs";
    }
    "BOL";
    {

```

```

        "displayName";
        "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-one";
        "Bolivianischer Boliviano (1863-1963)",
        "displayName-count-other";
        "Bolivianische Bolivianos (1863-1963)",
        "symbol";
        "BOL";
    }
    "BOP";
    {
        "displayName";
        "Bolivianischer Peso",
        "displayName-count-one";
        "Bolivianischer Peso",
        "displayName-count-other";
        "Bolivianische Peso",
        "symbol";
        "BOP";
    }
    "BOV";
    {
        "displayName";
        "Boliviansiche Mvdol",
        "displayName-count-one";
        "Boliviansiche Mvdol",
        "displayName-count-other";
        "Bolivianische Mvdol",
        "symbol";
        "BOV";
    }
    "BRB";
    {
        "displayName";
        "Brazilianischer Cruzeiro Novo (1967-1986)",
        "displayName-count-one";
        "Brazilianischer Cruzeiro Novo (1967-1986)",
        "displayName-count-other";
        "Brazilianische Cruzeiro Novo (1967-1986)",
        "symbol";
        "BRB";
    }
    "BRC";
    {
        "displayName";
        "Brazilianischer Cruzado (1986-1989)",
        "displayName-count-one";
        "Brazilianischer Cruzado (1986-1989)",
        "displayName-count-other";
        "Brazilianische Cruzado (1986-1989)",
        "symbol";
        "BRC";
    }
    "BRE";
    {
        "displayName";
        "Brazilianischer Cruzeiro (1990-1993)",

```

```

        "displayName-count-one";
        "Brazilianischer Cruzeiro (1990-1993)",
        "displayName-count-other";
        "Brasilianische Cruzeiro (1990-1993)",
        "symbol";
        "BRE";
    }
    "BRL";
    {
        "displayName";
        "Brazilianischer Real",
        "displayName-count-one";
        "Brazilianischer Real",
        "displayName-count-other";
        "Brasilianische Real",
        "symbol";
        "R$",
        "symbol-alt-narrow";
        "R$";
    }
    "BRN";
    {
        "displayName";
        "Brazilianischer Cruzado Novo (1989-1990)",
        "displayName-count-one";
        "Brazilianischer Cruzado Novo (1989-1990)",
        "displayName-count-other";
        "Brasilianische Cruzado Novo (1989-1990)",
        "symbol";
        "BRN";
    }
    "BRR";
    {
        "displayName";
        "Brazilianischer Cruzeiro (1993-1994)",
        "displayName-count-one";
        "Brazilianischer Cruzeiro (1993-1994)",
        "displayName-count-other";
        "Brasilianische Cruzeiro (1993-1994)",
        "symbol";
        "BRR";
    }
    "BRZ";
    {
        "displayName";
        "Brazilianischer Cruzeiro (1942-1967)",
        "displayName-count-one";
        "Brazilianischer Cruzeiro (1942-1967)",
        "displayName-count-other";
        "Brazilianischer Cruzeiro (1942-1967)",
        "symbol";
        "BRZ";
    }
    "BSD";
    {
        "displayName";
        "Bahamas-Dollar",

```

```

        "displayName-count-one";
        "Bahamas-Dollar",
        "displayName-count-other";
        "Bahamas-Dollar",
        "symbol";
        "BSD",
        "symbol-alt-narrow";
        "$";
    }
    "BTN";
    {
        "displayName";
        "Bhutan-Ngultrum",
        "displayName-count-one";
        "Bhutan-Ngultrum",
        "displayName-count-other";
        "Bhutan-Ngultrum",
        "symbol";
        "BTN";
    }
    "BUK";
    {
        "displayName";
        "Birmanischer Kyat",
        "displayName-count-one";
        "Birmanischer Kyat",
        "displayName-count-other";
        "Birmanische Kyat",
        "symbol";
        "BUK";
    }
    "BWP";
    {
        "displayName";
        "Botswanischer Pula",
        "displayName-count-one";
        "Botswanischer Pula",
        "displayName-count-other";
        "Botswanische Pula",
        "symbol";
        "BWP",
        "symbol-alt-narrow";
        "P";
    }
    "BYB";
    {
        "displayName";
        "Belarus-Rubel (1994-1999)",
        "displayName-count-one";
        "Belarus-Rubel (1994-1999)",
        "displayName-count-other";
        "Belarus-Rubel (1994-1999)",
        "symbol";
        "BYB";
    }
    "BYN";
    {

```



```

        "displayName";
        "Weißrussischer Rubel",
        "displayName-count-one";
        "Weißrussischer Rubel",
        "displayName-count-other";
        "Weißrussische Rubel",
        "symbol";
        "BYN",
        "symbol-alt-narrow";
        "p.";
    }
    "BYR";
    {
        "displayName";
        "Weißrussischer Rubel (2000-2016)",
        "displayName-count-one";
        "Weißrussischer Rubel (2000-2016)",
        "displayName-count-other";
        "Weißrussische Rubel (2000-2016)",
        "symbol";
        "BYR";
    }
    "BZD";
    {
        "displayName";
        "Belize-Dollar",
        "displayName-count-one";
        "Belize-Dollar",
        "displayName-count-other";
        "Belize-Dollar",
        "symbol";
        "BZD",
        "symbol-alt-narrow";
        "$";
    }
    "CAD";
    {
        "displayName";
        "Kanadischer Dollar",
        "displayName-count-one";
        "Kanadischer Dollar",
        "displayName-count-other";
        "Kanadische Dollar",
        "symbol";
        "CA$",
        "symbol-alt-narrow";
        "$";
    }
    "CDF";
    {
        "displayName";
        "Kongo-Franc",
        "displayName-count-one";
        "Kongo-Franc",
        "displayName-count-other";
        "Kongo-Francs",
        "symbol";
    }

```

```

        "CDF";
    }
    "CHE";
    {
        "displayName";
        "WIR-Euro",
        "displayName-count-one";
        "WIR-Euro",
        "displayName-count-other";
        "WIR-Euro",
        "symbol";
        "CHE";
    }
    "CHF";
    {
        "displayName";
        "Schweizer Franken",
        "displayName-count-one";
        "Schweizer Franken",
        "displayName-count-other";
        "Schweizer Franken",
        "symbol";
        "CHF";
    }
    "CHW";
    {
        "displayName";
        "WIR Franken",
        "displayName-count-one";
        "WIR Franken",
        "displayName-count-other";
        "WIR Franken",
        "symbol";
        "CHW";
    }
    "CLE";
    {
        "displayName";
        "Chilenischer Escudo",
        "displayName-count-one";
        "Chilenischer Escudo",
        "displayName-count-other";
        "Chilenische Escudo",
        "symbol";
        "CLE";
    }
    "CLF";
    {
        "displayName";
        "Chilenische Unidades de Fomento",
        "displayName-count-one";
        "Chilenische Unidades de Fomento",
        "displayName-count-other";
        "Chilenische Unidades de Fomento",
        "symbol";
        "CLF";
    }
}

```

```

"CLP";
{
  "displayName";
  "Chilenischer Peso",
    "displayName-count-one";
  "Chilenischer Peso",
    "displayName-count-other";
  "Chilenische Pesos",
    "symbol";
  "CLP",
    "symbol-alt-narrow";
  "$";
}
"CNX";
{
  "displayName";
  "Dollar der Chinesischen Volksbank",
    "displayName-count-one";
  "Dollar der Chinesischen Volksbank",
    "displayName-count-other";
  "Dollar der Chinesischen Volksbank",
    "symbol";
  "CNX";
}
"CNY";
{
  "displayName";
  "Renminbi Yuan",
    "displayName-count-one";
  "Chinesischer Yuan",
    "displayName-count-other";
  "Renminbi Yuan",
    "symbol";
  "CN¥",
    "symbol-alt-narrow";
  "¥";
}
"COP";
{
  "displayName";
  "Kolumbianischer Peso",
    "displayName-count-one";
  "Kolumbianischer Peso",
    "displayName-count-other";
  "Kolumbianische Pesos",
    "symbol";
  "COP",
    "symbol-alt-narrow";
  "$";
}
"COU";
{
  "displayName";
  "Kolumbianische Unidades de valor real",
    "displayName-count-one";
  "Kolumbianische Unidad de valor real",
    "displayName-count-other";
}

```

```

        "Kolumbianische Unidades de valor real",
        "symbol";
        "COU";
    }
    "CRC";
    {
        "displayName";
        "Costa-Rica-Colón",
        "displayName-count-one";
        "Costa-Rica-Colón",
        "displayName-count-other";
        "Costa-Rica-Colón",
        "symbol";
        "CRC",
        "symbol-alt-narrow";
        "₡";
    }
    "CSD";
    {
        "displayName";
        "Serbischer Dinar (2002-2006)",
        "displayName-count-one";
        "Serbischer Dinar (2002-2006)",
        "displayName-count-other";
        "Serbische Dinar (2002-2006)",
        "symbol";
        "CSD";
    }
    "CSK";
    {
        "displayName";
        "Tschechoslowakische Krone",
        "displayName-count-one";
        "Tschechoslowakische Kronen",
        "displayName-count-other";
        "Tschechoslowakische Kronen",
        "symbol";
        "CSK";
    }
    "CUC";
    {
        "displayName";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-one";
        "Kubanischer Peso (konvertibel)",
        "displayName-count-other";
        "Kubanische Pesos (konvertibel)",
        "symbol";
        "CUC",
        "symbol-alt-narrow";
        "Cub$";
    }
    "CUP";
    {
        "displayName";
        "Kubanischer Peso",
        "displayName-count-one";

```

```

        "Kubanischer Peso",
        "displayName-count-other";
    "Kubanische Pesos",
        "symbol";
    "CUP",
        "symbol-alt-narrow";
    "$";
}
"CVE";
{
    "displayName";
    "Cabo-Verde-Escudo",
        "displayName-count-one";
    "Cabo-Verde-Escudo",
        "displayName-count-other";
    "Cabo-Verde-Escudos",
        "symbol";
    "CVE";
}
"CYP";
{
    "displayName";
    "Zypern-Pfund",
        "displayName-count-one";
    "Zypern Pfund",
        "displayName-count-other";
    "Zypern Pfund",
        "symbol";
    "CYP";
}
"CZK";
{
    "displayName";
    "Tschechische Krone",
        "displayName-count-one";
    "Tschechische Krone",
        "displayName-count-other";
    "Tschechische Kronen",
        "symbol";
    "CZK",
        "symbol-alt-narrow";
    "Kč";
}
"DDM";
{
    "displayName";
    "Mark der DDR",
        "displayName-count-one";
    "Mark der DDR",
        "displayName-count-other";
    "Mark der DDR",
        "symbol";
    "DDM";
}
"DEM";
{
    "displayName";

```

```

        "Deutsche Mark",
        "displayName-count-one";
        "Deutsche Mark",
        "displayName-count-other";
        "Deutsche Mark",
        "symbol";
        "DM";
    }
    "DJF";
    {
        "displayName";
        "Dschibuti-Franc",
        "displayName-count-one";
        "Dschibuti-Franc",
        "displayName-count-other";
        "Dschibuti-Franc",
        "symbol";
        "DJF";
    }
    "DKK";
    {
        "displayName";
        "Dänische Krone",
        "displayName-count-one";
        "Dänische Krone",
        "displayName-count-other";
        "Dänische Kronen",
        "symbol";
        "DKK",
        "symbol-alt-narrow";
        "kr";
    }
    "DOP";
    {
        "displayName";
        "Dominikanischer Peso",
        "displayName-count-one";
        "Dominikanischer Peso",
        "displayName-count-other";
        "Dominikanische Pesos",
        "symbol";
        "DOP",
        "symbol-alt-narrow";
        "$";
    }
    "DZD";
    {
        "displayName";
        "Algerischer Dinar",
        "displayName-count-one";
        "Algerischer Dinar",
        "displayName-count-other";
        "Algerische Dinar",
        "symbol";
        "DZD";
    }
    "ECS";

```

```

        {
            "displayName";
            "Ecuadorianischer Sucre",
            "displayName-count-one";
            "Ecuadorianischer Sucre",
            "displayName-count-other";
            "Ecuadorianische Sucre",
            "symbol";
            "ECS";
        }
        "ECV";
        {
            "displayName";
            "Verrechnungseinheit für Ecuador",
            "displayName-count-one";
            "Verrechnungseinheiten für Ecuador",
            "displayName-count-other";
            "Verrechnungseinheiten für Ecuador",
            "symbol";
            "ECV";
        }
        "EEK";
        {
            "displayName";
            "Estnische Krone",
            "displayName-count-one";
            "Estnische Krone",
            "displayName-count-other";
            "Estnische Kronen",
            "symbol";
            "EEK";
        }
        "EGP";
        {
            "displayName";
            "Ägyptisches Pfund",
            "displayName-count-one";
            "Ägyptisches Pfund",
            "displayName-count-other";
            "Ägyptische Pfund",
            "symbol";
            "EGP",
            "symbol-alt-narrow";
            "E£";
        }
        "ERN";
        {
            "displayName";
            "Eritreischer Nakfa",
            "displayName-count-one";
            "Eritreischer Nakfa",
            "displayName-count-other";
            "Eritreische Nakfa",
            "symbol";
            "ERN";
        }
        "ESA";

```

```

    {
      "displayName";
      "Spanische Peseta (A-Konten)",
      "displayName-count-one";
      "Spanische Peseta (A-Konten)",
      "displayName-count-other";
      "Spanische Peseten (A-Konten)",
      "symbol";
      "ESA";
    }
    "ESB";
    {
      "displayName";
      "Spanische Peseta (konvertibel)",
      "displayName-count-one";
      "Spanische Peseta (konvertibel)",
      "displayName-count-other";
      "Spanische Peseten (konvertibel)",
      "symbol";
      "ESB";
    }
    "ESP";
    {
      "displayName";
      "Spanische Peseta",
      "displayName-count-one";
      "Spanische Peseta",
      "displayName-count-other";
      "Spanische Peseten",
      "symbol";
      "ESP",
      "symbol-alt-narrow";
      "₧";
    }
    "ETB";
    {
      "displayName";
      "Äthiopischer Birr",
      "displayName-count-one";
      "Äthiopischer Birr",
      "displayName-count-other";
      "Äthiopische Birr",
      "symbol";
      "ETB";
    }
    "EUR";
    {
      "displayName";
      "Euro",
      "displayName-count-one";
      "Euro",
      "displayName-count-other";
      "Euro",
      "symbol";
      "€",
      "symbol-alt-narrow";
      "€";
    }

```



```

    }
    "FIM";
    {
        "displayName";
        "Finnische Mark",
        "displayName-count-one";
        "Finnische Mark",
        "displayName-count-other";
        "Finnische Mark",
        "symbol";
        "FIM";
    }
    "FJD";
    {
        "displayName";
        "Fidschi-Dollar",
        "displayName-count-one";
        "Fidschi-Dollar",
        "displayName-count-other";
        "Fidschi-Dollar",
        "symbol";
        "FJD",
        "symbol-alt-narrow";
        "$";
    }
    "FKP";
    {
        "displayName";
        "Falkland-Pfund",
        "displayName-count-one";
        "Falkland-Pfund",
        "displayName-count-other";
        "Falkland-Pfund",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "Fl£";
    }
    "FRF";
    {
        "displayName";
        "Französischer Franc",
        "displayName-count-one";
        "Französischer Franc",
        "displayName-count-other";
        "Französische Franc",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "Britisches Pfund",
        "displayName-count-one";
        "Britisches Pfund",
        "displayName-count-other";
        "Britische Pfund",

```

```

        "symbol";
        "₾",
        "symbol-alt-narrow";
        "₾";
    }
    "GEEK";
    {
        "displayName";
        "Georgischer Kupon Larit",
        "displayName-count-one";
        "Georgischer Kupon Larit",
        "displayName-count-other";
        "Georgische Kupon Larit",
        "symbol";
        "GEEK";
    }
    "GEL";
    {
        "displayName";
        "Georgischer Lari",
        "displayName-count-one";
        "Georgischer Lari",
        "displayName-count-other";
        "Georgische Lari",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }
    "GHC";
    {
        "displayName";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-one";
        "Ghanaischer Cedi (1979-2007)",
        "displayName-count-other";
        "Ghanaische Cedi (1979-2007)",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "Ghanaischer Cedi",
        "displayName-count-one";
        "Ghanaischer Cedi",
        "displayName-count-other";
        "Ghanaische Cedi",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "Gibraltar-Pfund",

```

```

        "displayName-count-one";
        "Gibraltar-Pfund",
        "displayName-count-other";
        "Gibraltar Pfund",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
    "GMD";
    {
        "displayName";
        "Gambia-Dalasi",
        "displayName-count-one";
        "Gambia-Dalasi",
        "displayName-count-other";
        "Gambia-Dalasi",
        "symbol";
        "GMD";
    }
    "GNF";
    {
        "displayName";
        "Guinea-Franc",
        "displayName-count-one";
        "Guinea-Franc",
        "displayName-count-other";
        "Guinea-Franc",
        "symbol";
        "GNF",
        "symbol-alt-narrow";
        "F.G.";
    }
    "GNS";
    {
        "displayName";
        "Guineischer Syli",
        "displayName-count-one";
        "Guineischer Syli",
        "displayName-count-other";
        "Guineische Syli",
        "symbol";
        "GNS";
    }
    "GQE";
    {
        "displayName";
        "Äquatorialguinea-Ekwele",
        "displayName-count-one";
        "Äquatorialguinea-Ekwele",
        "displayName-count-other";
        "Äquatorialguinea-Ekwele",
        "symbol";
        "GQE";
    }
    "GRD";
    {

```

```

        "displayName";
        "Griechische Drachme",
        "displayName-count-one";
        "Griechische Drachme",
        "displayName-count-other";
        "Griechische Drachmen",
        "symbol";
        "GRD";
    }
    "GTQ";
    {
        "displayName";
        "Guatemaltekinscher Quetzal",
        "displayName-count-one";
        "Guatemaltekinscher Quetzal",
        "displayName-count-other";
        "Guatemaltekinsche Quetzales",
        "symbol";
        "GTQ",
        "symbol-alt-narrow";
        "Q";
    }
    "GWE";
    {
        "displayName";
        "Portugiesisch Guinea Escudo",
        "displayName-count-one";
        "Portugiesisch Guinea Escudo",
        "displayName-count-other";
        "Portugiesisch Guinea Escudo",
        "symbol";
        "GWE";
    }
    "GWP";
    {
        "displayName";
        "Guinea-Bissau Peso",
        "displayName-count-one";
        "Guinea-Bissau Peso",
        "displayName-count-other";
        "Guinea-Bissau Pesos",
        "symbol";
        "GWP";
    }
    "GYD";
    {
        "displayName";
        "Guyana-Dollar",
        "displayName-count-one";
        "Guyana-Dollar",
        "displayName-count-other";
        "Guyana-Dollar",
        "symbol";
        "GYD",
        "symbol-alt-narrow";
        "$";
    }
}

```

```

"HKD";
{
  "displayName";
  "Hongkong-Dollar",
    "displayName-count-one";
  "Hongkong-Dollar",
    "displayName-count-other";
  "Hongkong-Dollar",
    "symbol";
  "HK$";
    "symbol-alt-narrow";
  "$";
}
"HNL";
{
  "displayName";
  "Honduras-Lempira",
    "displayName-count-one";
  "Honduras-Lempira",
    "displayName-count-other";
  "Honduras-Lempira",
    "symbol";
  "HNL";
    "symbol-alt-narrow";
  "L";
}
"HRD";
{
  "displayName";
  "Kroatischer Dinar",
    "displayName-count-one";
  "Kroatischer Dinar",
    "displayName-count-other";
  "Kroatische Dinar",
    "symbol";
  "HRD";
}
"HRK";
{
  "displayName";
  "Kroatischer Kuna",
    "displayName-count-one";
  "Kroatischer Kuna",
    "displayName-count-other";
  "Kroatische Kuna",
    "symbol";
  "HRK";
    "symbol-alt-narrow";
  "kn";
}
"HTG";
{
  "displayName";
  "Haitianische Gourde",
    "displayName-count-one";
  "Haitianische Gourde",
    "displayName-count-other";
}

```

```

        "Haitianische Gourdes",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "Ungarischer Forint",
        "displayName-count-one";
        "Ungarischer Forint",
        "displayName-count-other";
        "Ungarische Forint",
        "symbol";
        "HUF",
        "symbol-alt-narrow";
        "Ft";
    }
    "IDR";
    {
        "displayName";
        "Indonesische Rupiah",
        "displayName-count-one";
        "Indonesische Rupiah",
        "displayName-count-other";
        "Indonesische Rupiah",
        "symbol";
        "IDR",
        "symbol-alt-narrow";
        "Rp";
    }
    "IEP";
    {
        "displayName";
        "Irisches Pfund",
        "displayName-count-one";
        "Irisches Pfund",
        "displayName-count-other";
        "Irische Pfund",
        "symbol";
        "IEP";
    }
    "ILP";
    {
        "displayName";
        "Israelisches Pfund",
        "displayName-count-one";
        "Israelisches Pfund",
        "displayName-count-other";
        "Israelische Pfund",
        "symbol";
        "ILP";
    }
    "ILR";
    {
        "displayName";
        "Israelischer Schekel (1980-1985)",
        "displayName-count-one";

```

```

        "Israelischer Schekel (1980-1985)",
        "displayName-count-other";
        "Israelische Schekel (1980-1985)";
    }
    "ILS";
    {
        "displayName";
        "Israelischer Neuer Schekel",
        "displayName-count-one";
        "Israelischer Neuer Schekel",
        "displayName-count-other";
        "Israelische Neue Schekel",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "Indische Rupie",
        "displayName-count-one";
        "Indische Rupie",
        "displayName-count-other";
        "Indische Rupien",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
    "IQD";
    {
        "displayName";
        "Irakischer Dinar",
        "displayName-count-one";
        "Irakischer Dinar",
        "displayName-count-other";
        "Irakische Dinar",
        "symbol";
        "IQD";
    }
    "IRR";
    {
        "displayName";
        "Iranischer Rial",
        "displayName-count-one";
        "Iranischer Rial",
        "displayName-count-other";
        "Iranische Rial",
        "symbol";
        "IRR";
    }
    "ISJ";
    {
        "displayName";
        "Isländische Krone (1918-1981)",
        "displayName-count-one";
    }

```

```

        "Isländische Krone (1918-1981)",
        "displayName-count-other";
        "Isländische Kronen (1918-1981)";
    }
    "ISK";
    {
        "displayName";
        "Isländische Krone",
        "displayName-count-one";
        "Isländische Krone",
        "displayName-count-other";
        "Isländische Kronen",
        "symbol";
        "ISK",
        "symbol-alt-narrow";
        "kr";
    }
    "ITL";
    {
        "displayName";
        "Italienische Lira",
        "displayName-count-one";
        "Italienische Lira",
        "displayName-count-other";
        "Italienische Lire",
        "symbol";
        "ITL";
    }
    "JMD";
    {
        "displayName";
        "Jamaika-Dollar",
        "displayName-count-one";
        "Jamaika-Dollar",
        "displayName-count-other";
        "Jamaika-Dollar",
        "symbol";
        "JMD",
        "symbol-alt-narrow";
        "$";
    }
    "JOD";
    {
        "displayName";
        "Jordanischer Dinar",
        "displayName-count-one";
        "Jordanischer Dinar",
        "displayName-count-other";
        "Jordanische Dinar",
        "symbol";
        "JOD";
    }
    "JPY";
    {
        "displayName";
        "Japanischer Yen",
        "displayName-count-one";

```



```

        "Japanischer Yen",
        "displayName-count-other";
    "Japanische Yen",
    "symbol";
    "¥",
    "symbol-alt-narrow";
    "¥";
}
"KES";
{
    "displayName";
    "Kenia-Schilling",
    "displayName-count-one";
    "Kenia-Schilling",
    "displayName-count-other";
    "Kenia-Schilling",
    "symbol";
    "KES";
}
"KGS";
{
    "displayName";
    "Kirgisischer Som",
    "displayName-count-one";
    "Kirgisischer Som",
    "displayName-count-other";
    "Kirgisische Som",
    "symbol";
    "KGS";
}
"KHR";
{
    "displayName";
    "Kambodschanischer Riel",
    "displayName-count-one";
    "Kambodschanischer Riel",
    "displayName-count-other";
    "Kambodschanische Riel",
    "symbol";
    "KHR",
    "symbol-alt-narrow";
    "៛";
}
"KMF";
{
    "displayName";
    "Komoren-Franc",
    "displayName-count-one";
    "Komoren-Franc",
    "displayName-count-other";
    "Komoren-Francs",
    "symbol";
    "KMF",
    "symbol-alt-narrow";
    "FC";
}

```

```

"KPW";
{
  "displayName";
  "Nordkoreanischer Won",
    "displayName-count-one";
  "Nordkoreanischer Won",
    "displayName-count-other";
  "Nordkoreanische Won",
    "symbol";
  "KPW",
    "symbol-alt-narrow";
  "₩";
}
"KRH";
{
  "displayName";
  "Südkoreanischer Hwan (1953-1962)",
    "displayName-count-one";
  "Südkoreanischer Hwan (1953-1962)",
    "displayName-count-other";
  "Südkoreanischer Hwan (1953-1962)",
    "symbol";
  "KRH";
}
"KRO";
{
  "displayName";
  "Südkoreanischer Won (1945-1953)",
    "displayName-count-one";
  "Südkoreanischer Won (1945-1953)",
    "displayName-count-other";
  "Südkoreanischer Won (1945-1953)",
    "symbol";
  "KRO";
}
"KRW";
{
  "displayName";
  "Südkoreanischer Won",
    "displayName-count-one";
  "Südkoreanischer Won",
    "displayName-count-other";
  "Südkoreanische Won",
    "symbol";
  "₩",
    "symbol-alt-narrow";
  "₩";
}
"KWD";
{
  "displayName";
  "Kuwait-Dinar",
    "displayName-count-one";
  "Kuwait-Dinar",
    "displayName-count-other";
  "Kuwait-Dinar",
    "symbol";
}

```

```

        "KWD";
    }
    "KYD";
    {
        "displayName";
        "Kaiman-Dollar",
        "displayName-count-one";
        "Kaiman-Dollar",
        "displayName-count-other";
        "Kaiman-Dollar",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
        "$";
    }
    "KZT";
    {
        "displayName";
        "Kasachischer Tenge",
        "displayName-count-one";
        "Kasachischer Tenge",
        "displayName-count-other";
        "Kasachische Tenge",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "T";
    }
    "LAK";
    {
        "displayName";
        "Laotischer Kip",
        "displayName-count-one";
        "Laotischer Kip",
        "displayName-count-other";
        "Laotische Kip",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "Libanesisches Pfund",
        "displayName-count-one";
        "Libanesisches Pfund",
        "displayName-count-other";
        "Libanesische Pfund",
        "symbol";
        "LBP",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";

```

```

        "Sri-Lanka-Rupie",
        "displayName-count-one";
        "Sri-Lanka-Rupie",
        "displayName-count-other";
        "Sri-Lanka-Rupien",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "LRD";
    {
        "displayName";
        "Liberianischer Dollar",
        "displayName-count-one";
        "Liberianischer Dollar",
        "displayName-count-other";
        "Liberianische Dollar",
        "symbol";
        "LRD",
        "symbol-alt-narrow";
        "$";
    }
    "LSL";
    {
        "displayName";
        "Loti",
        "displayName-count-one";
        "Loti",
        "displayName-count-other";
        "Loti",
        "symbol";
        "LSL";
    }
    "LTL";
    {
        "displayName";
        "Litauischer Litas",
        "displayName-count-one";
        "Litauischer Litas",
        "displayName-count-other";
        "Litauische Litas",
        "symbol";
        "LTL",
        "symbol-alt-narrow";
        "Lt";
    }
    "LTT";
    {
        "displayName";
        "Litauischer Talonas",
        "displayName-count-one";
        "Litauische Talonas",
        "displayName-count-other";
        "Litauische Talonas",
        "symbol";
        "LTT";
    }

```

```

    }
    "LUC";
    {
        "displayName";
        "Luxemburgischer Franc (konvertibel)",
        "displayName-count-one";
        "Luxemburgische Franc (konvertibel)",
        "displayName-count-other";
        "Luxemburgische Franc (konvertibel)",
        "symbol";
        "LUC";
    }
    "LUF";
    {
        "displayName";
        "Luxemburgischer Franc",
        "displayName-count-one";
        "Luxemburgische Franc",
        "displayName-count-other";
        "Luxemburgische Franc",
        "symbol";
        "LUF";
    }
    "LUL";
    {
        "displayName";
        "Luxemburgischer Finanz-Franc",
        "displayName-count-one";
        "Luxemburgische Finanz-Franc",
        "displayName-count-other";
        "Luxemburgische Finanz-Franc",
        "symbol";
        "LUL";
    }
    "LVL";
    {
        "displayName";
        "Lettischer Lats",
        "displayName-count-one";
        "Lettischer Lats",
        "displayName-count-other";
        "Lettische Lats",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "Lettischer Rubel",
        "displayName-count-one";
        "Lettische Rubel",
        "displayName-count-other";
        "Lettische Rubel",
        "symbol";
        "LVR";
    }

```

```

    }
    "LYD";
    {
        "displayName";
        "Libyscher Dinar",
        "displayName-count-one";
        "Libyscher Dinar",
        "displayName-count-other";
        "Libysche Dinar",
        "symbol";
        "LYD";
    }
    "MAD";
    {
        "displayName";
        "Marokkanischer Dirham",
        "displayName-count-one";
        "Marokkanischer Dirham",
        "displayName-count-other";
        "Marokkanische Dirham",
        "symbol";
        "MAD";
    }
    "MAF";
    {
        "displayName";
        "Marokkanischer Franc",
        "displayName-count-one";
        "Marokkanische Franc",
        "displayName-count-other";
        "Marokkanische Franc",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "Monegassischer Franc",
        "displayName-count-one";
        "Monegassischer Franc",
        "displayName-count-other";
        "Monegassische Franc",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "Moldau-Cupon",
        "displayName-count-one";
        "Moldau-Cupon",
        "displayName-count-other";
        "Moldau-Cupon",
        "symbol";
        "MDC";
    }
    "MDL";

```

```

    {
      "displayName";
      "Moldau-Leu",
      "displayName-count-one";
      "Moldau-Leu",
      "displayName-count-other";
      "Moldau-Leu",
      "symbol";
      "MDL";
    }
    "MGA";
    {
      "displayName";
      "Madagaskar-Ariary",
      "displayName-count-one";
      "Madagaskar-Ariary",
      "displayName-count-other";
      "Madagaskar-Ariary",
      "symbol";
      "MGA",
      "symbol-alt-narrow";
      "Ar";
    }
    "MGF";
    {
      "displayName";
      "Madagaskar-Franc",
      "displayName-count-one";
      "Madagaskar-Franc",
      "displayName-count-other";
      "Madagaskar-Franc",
      "symbol";
      "MGF";
    }
    "MKD";
    {
      "displayName";
      "Mazedonischer Denar",
      "displayName-count-one";
      "Mazedonischer Denar",
      "displayName-count-other";
      "Mazedonische Denari",
      "symbol";
      "MKD";
    }
    "MKN";
    {
      "displayName";
      "Mazedonischer Denar (1992-1993)",
      "displayName-count-one";
      "Mazedonischer Denar (1992-1993)",
      "displayName-count-other";
      "Mazedonische Denar (1992-1993)",
      "symbol";
      "MKN";
    }
    "MLF";

```

```

    {
      "displayName";
      "Malischer Franc",
        "displayName-count-one";
      "Malische Franc",
        "displayName-count-other";
      "Malische Franc",
        "symbol";
      "MLF";
    }
    "MMK";
    {
      "displayName";
      "Myanmarischer Kyat",
        "displayName-count-one";
      "Myanmarischer Kyat",
        "displayName-count-other";
      "Myanmarische Kyat",
        "symbol";
      "MMK",
        "symbol-alt-narrow";
      "K";
    }
    "MNT";
    {
      "displayName";
      "Mongolischer Tögrög",
        "displayName-count-one";
      "Mongolischer Tögrög",
        "displayName-count-other";
      "Mongolische Tögrög",
        "symbol";
      "MNT",
        "symbol-alt-narrow";
      "₮";
    }
    "MOP";
    {
      "displayName";
      "Macao-Pataca",
        "displayName-count-one";
      "Macao-Pataca",
        "displayName-count-other";
      "Macao-Pataca",
        "symbol";
      "MOP";
    }
    "MRO";
    {
      "displayName";
      "Mauretanischer Ouguiya",
        "displayName-count-one";
      "Mauretanischer Ouguiya",
        "displayName-count-other";
      "Mauretanische Ouguiya",
        "symbol";
      "MRO";
    }

```



```

    }
    "MTL";
    {
        "displayName";
        "Maltesische Lira",
        "displayName-count-one";
        "Maltesische Lira",
        "displayName-count-other";
        "Maltesische Lira",
        "symbol";
        "MTL";
    }
    "MTP";
    {
        "displayName";
        "Maltesisches Pfund",
        "displayName-count-one";
        "Maltesische Pfund",
        "displayName-count-other";
        "Maltesische Pfund",
        "symbol";
        "MTP";
    }
    "MUR";
    {
        "displayName";
        "Mauritius-Rupie",
        "displayName-count-one";
        "Mauritius-Rupie",
        "displayName-count-other";
        "Mauritius-Rupien",
        "symbol";
        "MUR",
        "symbol-alt-narrow";
        "Rs";
    }
    "MVP";
    {
        "displayName";
        "Malediven-Rupie (alt)",
        "displayName-count-one";
        "Malediven-Rupie (alt)",
        "displayName-count-other";
        "Malediven-Rupien (alt)";
    }
    "MVR";
    {
        "displayName";
        "Malediven-Rufiyaa",
        "displayName-count-one";
        "Malediven-Rufiyaa",
        "displayName-count-other";
        "Malediven-Rupien",
        "symbol";
        "MVR";
    }
    "MWK";

```

```

    {
      "displayName";
      "Malawi-Kwacha",
      "displayName-count-one";
      "Malawi-Kwacha",
      "displayName-count-other";
      "Malawi-Kwacha",
      "symbol";
      "MWK";
    }
    "MXN";
    {
      "displayName";
      "Mexikanischer Peso",
      "displayName-count-one";
      "Mexikanischer Peso",
      "displayName-count-other";
      "Mexikanische Pesos",
      "symbol";
      "MX$";
      "symbol-alt-narrow";
      "$";
    }
    "MXP";
    {
      "displayName";
      "Mexikanischer Silber-Peso (1861-1992)",
      "displayName-count-one";
      "Mexikanische Silber-Peso (1861-1992)",
      "displayName-count-other";
      "Mexikanische Silber-Pesos (1861-1992)",
      "symbol";
      "MXP";
    }
    "MXV";
    {
      "displayName";
      "Mexicanischer Unidad de Inversion (UDI)",
      "displayName-count-one";
      "Mexicanischer Unidad de Inversion (UDI)",
      "displayName-count-other";
      "Mexikanische Unidad de Inversion (UDI)",
      "symbol";
      "MXV";
    }
    "MYR";
    {
      "displayName";
      "Malaysischer Ringgit",
      "displayName-count-one";
      "Malaysischer Ringgit",
      "displayName-count-other";
      "Malaysische Ringgit",
      "symbol";
      "MYR",
      "symbol-alt-narrow";
      "RM";
    }
  }

```

```

    }
    "MZE";
    {
        "displayName";
        "Mosambikanischer Escudo",
        "displayName-count-one";
        "Mozambikanische Escudo",
        "displayName-count-other";
        "Mozambikanische Escudo",
        "symbol";
        "MZE";
    }
    "MZM";
    {
        "displayName";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-one";
        "Mosambikanischer Metical (1980-2006)",
        "displayName-count-other";
        "Mosambikanische Meticaïs (1980-2006)",
        "symbol";
        "MZM";
    }
    "MZN";
    {
        "displayName";
        "Mosambikanischer Metical",
        "displayName-count-one";
        "Mosambikanischer Metical",
        "displayName-count-other";
        "Mosambikanische Meticaïs",
        "symbol";
        "MZN";
    }
    "NAD";
    {
        "displayName";
        "Namibia-Dollar",
        "displayName-count-one";
        "Namibia-Dollar",
        "displayName-count-other";
        "Namibia-Dollar",
        "symbol";
        "NAD",
        "symbol-alt-narrow";
        "$";
    }
    "NGN";
    {
        "displayName";
        "Nigerianischer Naira",
        "displayName-count-one";
        "Nigerianischer Naira",
        "displayName-count-other";
        "Nigerianische Naira",
        "symbol";
        "NGN",

```

```

        "symbol-alt-narrow";
        "₡";
    }
    "NIC";
    {
        "displayName";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-one";
        "Nicaraguanischer Córdoba (1988-1991)",
        "displayName-count-other";
        "Nicaraguanische Córdoba (1988-1991)",
        "symbol";
        "NIC";
    }
    "NIO";
    {
        "displayName";
        "Nicaragua-Córdoba",
        "displayName-count-one";
        "Nicaragua-Córdoba",
        "displayName-count-other";
        "Nicaragua-Córdobas",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";
        "Niederländischer Gulden",
        "displayName-count-one";
        "Niederländischer Gulden",
        "displayName-count-other";
        "Niederländische Gulden",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "Norwegische Krone",
        "displayName-count-one";
        "Norwegische Krone",
        "displayName-count-other";
        "Norwegische Kronen",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "Nepalesische Rupie",
        "displayName-count-one";
        "Nepalesische Rupie",

```

```

        "displayName-count-other";
        "Nepalesische Rupien",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {
        "displayName";
        "Neuseeland-Dollar",
        "displayName-count-one";
        "Neuseeland-Dollar",
        "displayName-count-other";
        "Neuseeland-Dollar",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
        "$";
    }
    "OMR";
    {
        "displayName";
        "Omanischer Rial",
        "displayName-count-one";
        "Omanischer Rial",
        "displayName-count-other";
        "Omanische Rials",
        "symbol";
        "OMR";
    }
    "PAB";
    {
        "displayName";
        "Panamaischer Balboa",
        "displayName-count-one";
        "Panamaischer Balboa",
        "displayName-count-other";
        "Panamaische Balboas",
        "symbol";
        "PAB";
    }
    "PEI";
    {
        "displayName";
        "Peruanischer Inti",
        "displayName-count-one";
        "Peruanische Inti",
        "displayName-count-other";
        "Peruanische Inti",
        "symbol";
        "PEI";
    }
    "PEN";
    {
        "displayName";
        "Peruanischer Sol",

```

```

        "displayName-count-one";
        "Peruanischer Sol",
        "displayName-count-other";
        "Peruanische Sol",
        "symbol";
        "PEN";
    }
    "PES";
    {
        "displayName";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-one";
        "Peruanischer Sol (1863-1965)",
        "displayName-count-other";
        "Peruanische Sol (1863-1965)",
        "symbol";
        "PES";
    }
    "PGK";
    {
        "displayName";
        "Papua-Neuguineischer Kina",
        "displayName-count-one";
        "Papua-Neuguineischer Kina",
        "displayName-count-other";
        "Papua-Neuguineische Kina",
        "symbol";
        "PGK";
    }
    "PHP";
    {
        "displayName";
        "Philippinischer Peso",
        "displayName-count-one";
        "Philippinischer Peso",
        "displayName-count-other";
        "Philippinische Pesos",
        "symbol";
        "PHP",
        "symbol-alt-narrow";
        "₱";
    }
    "PKR";
    {
        "displayName";
        "Pakistanische Rupie",
        "displayName-count-one";
        "Pakistanische Rupie",
        "displayName-count-other";
        "Pakistanische Rupien",
        "symbol";
        "PKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "PLN";
    {

```

```

        "displayName";
        "Polnischer Złoty",
        "displayName-count-one";
        "Polnischer Złoty",
        "displayName-count-other";
        "Polnische Złoty",
        "symbol";
        "PLN",
        "symbol-alt-narrow";
        "zł";
    }
    "PLZ";
    {
        "displayName";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-one";
        "Polnischer Zloty (1950-1995)",
        "displayName-count-other";
        "Polnische Zloty (1950-1995)",
        "symbol";
        "PLZ";
    }
    "PTE";
    {
        "displayName";
        "Portugiesischer Escudo",
        "displayName-count-one";
        "Portugiesische Escudo",
        "displayName-count-other";
        "Portugiesische Escudo",
        "symbol";
        "PTE";
    }
    "PYG";
    {
        "displayName";
        "Paraguayischer Guaraní",
        "displayName-count-one";
        "Paraguayischer Guaraní",
        "displayName-count-other";
        "Paraguayische Guaraníes",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "Katar-Riyal",
        "displayName-count-one";
        "Katar-Riyal",
        "displayName-count-other";
        "Katar-Riyal",
        "symbol";
        "QAR";
    }
}

```

```
"RHD";
{
  "displayName";
  "Rhodesischer Dollar",
    "displayName-count-one";
  "Rhodesische Dollar",
    "displayName-count-other";
  "Rhodesische Dollar",
    "symbol";
  "RHD";
}
"ROL";
{
  "displayName";
  "Rumänischer Leu (1952-2006)",
    "displayName-count-one";
  "Rumänischer Leu (1952-2006)",
    "displayName-count-other";
  "Rumänische Leu (1952-2006)",
    "symbol";
  "ROL";
}
"RON";
{
  "displayName";
  "Rumänischer Leu",
    "displayName-count-one";
  "Rumänischer Leu",
    "displayName-count-other";
  "Rumänische Leu",
    "symbol";
  "RON",
    "symbol-alt-narrow";
  "L";
}
"RSD";
{
  "displayName";
  "Serbischer Dinar",
    "displayName-count-one";
  "Serbischer Dinar",
    "displayName-count-other";
  "Serbische Dinaren",
    "symbol";
  "RSD";
}
"RUB";
{
  "displayName";
  "Russischer Rubel",
    "displayName-count-one";
  "Russischer Rubel",
    "displayName-count-other";
  "Russische Rubel",
    "symbol";
  "RUB",
    "symbol-alt-narrow";
```



```

        "F";
    }
    "RUR";
    {
        "displayName";
        "Russischer Rubel (1991-1998)",
        "displayName-count-one";
        "Russischer Rubel (1991-1998)",
        "displayName-count-other";
        "Russische Rubel (1991-1998)",
        "symbol";
        "RUR",
        "symbol-alt-narrow";
        "p.";
    }
    "RWF";
    {
        "displayName";
        "Ruanda-Franc",
        "displayName-count-one";
        "Ruanda-Franc",
        "displayName-count-other";
        "Ruanda-Francs",
        "symbol";
        "RWF",
        "symbol-alt-narrow";
        "F.Rw";
    }
    "SAR";
    {
        "displayName";
        "Saudi-Rial",
        "displayName-count-one";
        "Saudi-Rial",
        "displayName-count-other";
        "Saudi-Rial",
        "symbol";
        "SAR";
    }
    "SBD";
    {
        "displayName";
        "Salomonen-Dollar",
        "displayName-count-one";
        "Salomonen-Dollar",
        "displayName-count-other";
        "Salomonen-Dollar",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "$";
    }
    "SCR";
    {
        "displayName";
        "Seychellen-Rupie",
        "displayName-count-one";

```

```

        "Seychellen-Rupie",
        "displayName-count-other";
        "Seychellen-Rupien",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-one";
        "Sudanesischer Dinar (1992-2007)",
        "displayName-count-other";
        "Sudanesische Dinar (1992-2007)",
        "symbol";
        "SDD";
    }
    "SDG";
    {
        "displayName";
        "Sudanesisches Pfund",
        "displayName-count-one";
        "Sudanesisches Pfund",
        "displayName-count-other";
        "Sudanesische Pfund",
        "symbol";
        "SDG";
    }
    "SDP";
    {
        "displayName";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-one";
        "Sudanesisches Pfund (1957-1998)",
        "displayName-count-other";
        "Sudanesische Pfund (1957-1998)",
        "symbol";
        "SDP";
    }
    "SEK";
    {
        "displayName";
        "Schwedische Krone",
        "displayName-count-one";
        "Schwedische Krone",
        "displayName-count-other";
        "Schwedische Kronen",
        "symbol";
        "SEK",
        "symbol-alt-narrow";
        "kr";
    }
    "SGD";
    {
        "displayName";
        "Singapur-Dollar",
        "displayName-count-one";

```

```

        "Singapur-Dollar",
        "displayName-count-other";
    "Singapur-Dollar",
        "symbol";
    "SGD",
        "symbol-alt-narrow";
    "$";
}
"SHP";
{
    "displayName";
    "St. Helena-Pfund",
        "displayName-count-one";
    "St. Helena-Pfund",
        "displayName-count-other";
    "St. Helena-Pfund",
        "symbol";
    "SHP",
        "symbol-alt-narrow";
    "£";
}
"SIT";
{
    "displayName";
    "Slowenischer Tolar",
        "displayName-count-one";
    "Slowenischer Tolar",
        "displayName-count-other";
    "Slowenische Tolar",
        "symbol";
    "SIT";
}
"SKK";
{
    "displayName";
    "Slowakische Krone",
        "displayName-count-one";
    "Slowakische Kronen",
        "displayName-count-other";
    "Slowakische Kronen",
        "symbol";
    "SKK";
}
"SLL";
{
    "displayName";
    "Sierra-leonischer Leone",
        "displayName-count-one";
    "Sierra-leonischer Leone",
        "displayName-count-other";
    "Sierra-leonische Leones",
        "symbol";
    "SLL";
}
"SOS";
{
    "displayName";

```

```

        "Somalia-Schilling",
        "displayName-count-one";
        "Somalia-Schilling",
        "displayName-count-other";
        "Somalia-Schilling",
        "symbol";
        "SOS";
    }
    "SRD";
    {
        "displayName";
        "Suriname-Dollar",
        "displayName-count-one";
        "Suriname-Dollar",
        "displayName-count-other";
        "Suriname-Dollar",
        "symbol";
        "SRD",
        "symbol-alt-narrow";
        "$";
    }
    "SRG";
    {
        "displayName";
        "Suriname Gulden",
        "displayName-count-one";
        "Suriname-Gulden",
        "displayName-count-other";
        "Suriname-Gulden",
        "symbol";
        "SRG";
    }
    "SSP";
    {
        "displayName";
        "Südsudanesisches Pfund",
        "displayName-count-one";
        "Südsudanesisches Pfund",
        "displayName-count-other";
        "Südsudanesische Pfund",
        "symbol";
        "SSP",
        "symbol-alt-narrow";
        "£";
    }
    "STD";
    {
        "displayName";
        "São-toméischer Dobra",
        "displayName-count-one";
        "São-toméischer Dobra",
        "displayName-count-other";
        "São-toméische Dobra",
        "symbol";
        "STD",
        "symbol-alt-narrow";
        "Db";
    }

```

```

    }
    "SUR";
    {
        "displayName";
        "Sowjetischer Rubel",
        "displayName-count-one";
        "Sowjetische Rubel",
        "displayName-count-other";
        "Sowjetische Rubel",
        "symbol";
        "SUR";
    }
    "SVC";
    {
        "displayName";
        "El Salvador Colon",
        "displayName-count-one";
        "El Salvador-Colon",
        "displayName-count-other";
        "El Salvador-Colon",
        "symbol";
        "SVC";
    }
    "SYP";
    {
        "displayName";
        "Syrisches Pfund",
        "displayName-count-one";
        "Syrisches Pfund",
        "displayName-count-other";
        "Syrische Pfund",
        "symbol";
        "SYP",
        "symbol-alt-narrow";
        "SYP";
    }
    "SZL";
    {
        "displayName";
        "Swasiländischer Lilangeni",
        "displayName-count-one";
        "Swasiländischer Lilangeni",
        "displayName-count-other";
        "Swasiländische Emalangeni",
        "symbol";
        "SZL";
    }
    "THB";
    {
        "displayName";
        "Thailändischer Baht",
        "displayName-count-one";
        "Thailändischer Baht",
        "displayName-count-other";
        "Thailändische Baht",
        "symbol";
        "฿";
    }

```

```

        "symbol-alt-narrow";
        "B";
    }
    "TJR";
    {
        "displayName";
        "Tadschikistan Rubel",
        "displayName-count-one";
        "Tadschikistan-Rubel",
        "displayName-count-other";
        "Tadschikistan-Rubel",
        "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "Tadschikistan-Somoni",
        "displayName-count-one";
        "Tadschikistan-Somoni",
        "displayName-count-other";
        "Tadschikistan-Somoni",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-one";
        "Turkmenistan-Manat (1993-2009)",
        "displayName-count-other";
        "Turkmenistan-Manat (1993-2009)",
        "symbol";
        "TMM";
    }
    "TMT";
    {
        "displayName";
        "Turkmenistan-Manat",
        "displayName-count-one";
        "Turkmenistan-Manat",
        "displayName-count-other";
        "Turkmenistan-Manat",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "Tunesischer Dinar",
        "displayName-count-one";
        "Tunesischer Dinar",
        "displayName-count-other";
        "Tunesische Dinar",
        "symbol";
        "TND";
    }

```

```

    }
    "TOP";
    {
        "displayName";
        "Tongaischer Pa'anga",
        "displayName-count-one";
        "Tongaischer Pa'anga",
        "displayName-count-other";
        "Tongaische Pa'anga",
        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
    "TPE";
    {
        "displayName";
        "Timor-Escudo",
        "displayName-count-one";
        "Timor-Escudo",
        "displayName-count-other";
        "Timor-Escudo",
        "symbol";
        "TPE";
    }
    "TRL";
    {
        "displayName";
        "Türkische Lira (1922-2005)",
        "displayName-count-one";
        "Türkische Lira (1922-2005)",
        "displayName-count-other";
        "Türkische Lira (1922-2005)",
        "symbol";
        "TRL";
    }
    "TRY";
    {
        "displayName";
        "Türkische Lira",
        "displayName-count-one";
        "Türkische Lira",
        "displayName-count-other";
        "Türkische Lira",
        "symbol";
        "TRY",
        "symbol-alt-narrow";
        "₺",
        "symbol-alt-variant";
        "TL";
    }
    "TTD";
    {
        "displayName";
        "Trinidad und Tobago-Dollar",
        "displayName-count-one";
        "Trinidad und Tobago-Dollar",

```

```

        "displayName-count-other";
        "Trinidad und Tobago-Dollar",
        "symbol";
        "TTD",
        "symbol-alt-narrow";
        "$";
    }
    "TWD";
    {
        "displayName";
        "Neuer Taiwan-Dollar",
        "displayName-count-one";
        "Neuer Taiwan-Dollar",
        "displayName-count-other";
        "Neue Taiwan-Dollar",
        "symbol";
        "NT$";
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";
    {
        "displayName";
        "Tansania-Schilling",
        "displayName-count-one";
        "Tansania-Schilling",
        "displayName-count-other";
        "Tansania-Schilling",
        "symbol";
        "TZS";
    }
    "UAH";
    {
        "displayName";
        "Ukrainische Hrywnja",
        "displayName-count-one";
        "Ukrainische Hrywnja",
        "displayName-count-other";
        "Ukrainische Hrywen",
        "symbol";
        "UAH";
        "symbol-alt-narrow";
        "₴";
    }
    "UAK";
    {
        "displayName";
        "Ukrainischer Karbovanetz",
        "displayName-count-one";
        "Ukrainische Karbovanetz",
        "displayName-count-other";
        "Ukrainische Karbovanetz",
        "symbol";
        "UAK";
    }
    "UGS";
    {

```



```

        "displayName";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-one";
        "Uganda-Schilling (1966-1987)",
        "displayName-count-other";
        "Uganda-Schilling (1966-1987)",
        "symbol";
        "UGS";
    }
    "UGX";
    {
        "displayName";
        "Uganda-Schilling",
        "displayName-count-one";
        "Uganda-Schilling",
        "displayName-count-other";
        "Uganda-Schilling",
        "symbol";
        "UGX";
    }
    "USD";
    {
        "displayName";
        "US-Dollar",
        "displayName-count-one";
        "US-Dollar",
        "displayName-count-other";
        "US-Dollar",
        "symbol";
        "$",
        "symbol-alt-narrow";
        "$";
    }
    "USN";
    {
        "displayName";
        "US Dollar (Nächster Tag)",
        "displayName-count-one";
        "US-Dollar (Nächster Tag)",
        "displayName-count-other";
        "US-Dollar (Nächster Tag)",
        "symbol";
        "USN";
    }
    "USS";
    {
        "displayName";
        "US Dollar (Gleicher Tag)",
        "displayName-count-one";
        "US-Dollar (Gleicher Tag)",
        "displayName-count-other";
        "US-Dollar (Gleicher Tag)",
        "symbol";
        "USS";
    }
    "UYI";
    {

```

```

        "displayName";
        "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
        "displayName-count-one";
        "Uruguayischer Peso (Indexierte
Rechnungseinheiten)",
        "displayName-count-other";
        "Uruguayische Pesos (Indexierte
Rechnungseinheiten)",
        "symbol";
        "UYI";
    }
    "UYP";
    {
        "displayName";
        "Uruguayischer Peso (1975-1993)",
        "displayName-count-one";
        "Uruguayischer Peso (1975-1993)",
        "displayName-count-other";
        "Uruguayische Pesos (1975-1993)",
        "symbol";
        "UYU";
    }
    "UYU";
    {
        "displayName";
        "Uruguayischer Peso",
        "displayName-count-one";
        "Uruguayischer Peso",
        "displayName-count-other";
        "Uruguayische Pesos",
        "symbol";
        "UYU",
        "symbol-alt-narrow";
        "$";
    }
    "UZS";
    {
        "displayName";
        "Usbekistan-Sum",
        "displayName-count-one";
        "Usbekistan-Sum",
        "displayName-count-other";
        "Usbekistan-Sum",
        "symbol";
        "UZS";
    }
    "VEB";
    {
        "displayName";
        "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-one";
        "Venezolanischer Bolívar (1871-2008)",
        "displayName-count-other";
        "Venezolanische Bolívares (1871-2008)",
        "symbol";
        "VEB";
    }

```

```

    }
    "VEF";
    {
        "displayName";
        "Venezolanischer Bolívar",
        "displayName-count-one";
        "Venezolanischer Bolívar",
        "displayName-count-other";
        "Venezolanische Bolívares",
        "symbol";
        "VEF",
        "symbol-alt-narrow";
        "Bs";
    }
    "VND";
    {
        "displayName";
        "Vietnamesischer Dong",
        "displayName-count-one";
        "Vietnamesischer Dong",
        "displayName-count-other";
        "Vietnamesische Dong",
        "symbol";
        "₫",
        "symbol-alt-narrow";
        "₫";
    }
    "VNN";
    {
        "displayName";
        "Vietnamesischer Dong (1978–1985) ",
        "displayName-count-one";
        "Vietnamesischer Dong (1978–1985) ",
        "displayName-count-other";
        "Vietnamesische Dong (1978–1985) ",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "Vanuatu-Vatu",
        "displayName-count-one";
        "Vanuatu-Vatu",
        "displayName-count-other";
        "Vanuatu-Vatu",
        "symbol";
        "VUV";
    }
    "WST";
    {
        "displayName";
        "Samoanischer Tala",
        "displayName-count-one";
        "Samoanischer Tala",
        "displayName-count-other";
        "Samoanische Tala",

```

```

        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "CFA-Franc (BEAC) ",
        "displayName-count-one";
        "CFA-Franc (BEAC) ",
        "displayName-count-other";
        "CFA-Franc (BEAC) ",
        "symbol";
        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "Unze Silber",
        "displayName-count-one";
        "Unze Silber",
        "displayName-count-other";
        "Unzen Silber",
        "symbol";
        "XAG";
    }
    "XAU";
    {
        "displayName";
        "Unze Gold",
        "displayName-count-one";
        "Unze Gold",
        "displayName-count-other";
        "Unzen Gold",
        "symbol";
        "XAU";
    }
    "XBA";
    {
        "displayName";
        "Europäische Rechnungseinheit",
        "displayName-count-one";
        "Europäische Rechnungseinheiten",
        "displayName-count-other";
        "Europäische Rechnungseinheiten",
        "symbol";
        "XBA";
    }
    "XBB";
    {
        "displayName";
        "Europäische Währungseinheit (XBB) ",
        "displayName-count-one";
        "Europäische Währungseinheiten (XBB) ",
        "displayName-count-other";
        "Europäische Währungseinheiten (XBB) ",
        "symbol";
        "XBB";
    }

```

```

    }
    "XBC";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBC)",
        "displayName-count-one";
        "Europäische Rechnungseinheiten (XBC)",
        "displayName-count-other";
        "Europäische Rechnungseinheiten (XBC)",
        "symbol";
        "XBC";
    }
    "XBD";
    {
        "displayName";
        "Europäische Rechnungseinheit (XBD)",
        "displayName-count-one";
        "Europäische Rechnungseinheiten (XBD)",
        "displayName-count-other";
        "Europäische Rechnungseinheiten (XBD)",
        "symbol";
        "XBD";
    }
    "XCD";
    {
        "displayName";
        "Ostkaribischer Dollar",
        "displayName-count-one";
        "Ostkaribischer Dollar",
        "displayName-count-other";
        "Ostkaribische Dollar",
        "symbol";
        "EC$",
        "symbol-alt-narrow";
        "$";
    }
    "XDR";
    {
        "displayName";
        "Sonderziehungsrechte",
        "displayName-count-one";
        "Sonderziehungsrechte",
        "displayName-count-other";
        "Sonderziehungsrechte",
        "symbol";
        "XDR";
    }
    "XEU";
    {
        "displayName";
        "Europäische Währungseinheit (XEU)",
        "displayName-count-one";
        "Europäische Währungseinheiten (XEU)",
        "displayName-count-other";
        "Europäische Währungseinheiten (XEU)",
        "symbol";
        "XEU";
    }

```

```

    }
    "XFO";
    {
        "displayName";
        "Französischer Gold-Franc",
        "displayName-count-one";
        "Französische Gold-Franc",
        "displayName-count-other";
        "Französische Gold-Franc",
        "symbol";
        "XFO";
    }
    "XFU";
    {
        "displayName";
        "Französischer UIC-Franc",
        "displayName-count-one";
        "Französische UIC-Franc",
        "displayName-count-other";
        "Französische UIC-Franc",
        "symbol";
        "XFU";
    }
    "XOF";
    {
        "displayName";
        "CFA-Franc (BCEAO)",
        "displayName-count-one";
        "CFA-Franc (BCEAO)",
        "displayName-count-other";
        "CFA-Francs (BCEAO)",
        "symbol";
        "CFA";
    }
    "XPD";
    {
        "displayName";
        "Unze Palladium",
        "displayName-count-one";
        "Unze Palladium",
        "displayName-count-other";
        "Unzen Palladium",
        "symbol";
        "XPD";
    }
    "XPF";
    {
        "displayName";
        "CFP-Franc",
        "displayName-count-one";
        "CFP-Franc",
        "displayName-count-other";
        "CFP-Franc",
        "symbol";
        "CFPF";
    }
    "XPT";

```

```

        {
            "displayName";
            "Unze Platin",
            "displayName-count-one";
            "Unze Platin",
            "displayName-count-other";
            "Unzen Platin",
            "symbol";
            "XPT";
        }
        "XRE";
        {
            "displayName";
            "RINET Funds",
            "displayName-count-one";
            "RINET Funds",
            "displayName-count-other";
            "RINET Funds",
            "symbol";
            "XRE";
        }
        "XSU";
        {
            "displayName";
            "SUCRE",
            "displayName-count-one";
            "SUCRE",
            "displayName-count-other";
            "SUCRE",
            "symbol";
            "XSU";
        }
        "XTS";
        {
            "displayName";
            "Testwährung",
            "displayName-count-one";
            "Testwährung",
            "displayName-count-other";
            "Testwährung",
            "symbol";
            "XTS";
        }
        "XUA";
        {
            "displayName";
            "Rechnungseinheit der AfEB",
            "displayName-count-one";
            "Rechnungseinheit der AfEB",
            "displayName-count-other";
            "Rechnungseinheiten der AfEB",
            "symbol";
            "XUA";
        }
        "XXX";
        {
            "displayName";

```

```

        "Unbekannte Währung",
        "displayName-count-one";
    "(unbekannte Währung)",
        "displayName-count-other";
    "(unbekannte Währung)",
        "symbol";
    "XXX";
}
"YDD";
{
    "displayName";
    "Jemen-Dinar",
        "displayName-count-one";
    "Jemen-Dinar",
        "displayName-count-other";
    "Jemen-Dinar",
        "symbol";
    "YDD";
}
"YER";
{
    "displayName";
    "Jemen-Rial",
        "displayName-count-one";
    "Jemen-Rial",
        "displayName-count-other";
    "Jemen-Rial",
        "symbol";
    "YER";
}
"YUD";
{
    "displayName";
    "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-one";
    "Jugoslawischer Dinar (1966-1990)",
        "displayName-count-other";
    "Jugoslawische Dinar (1966-1990)",
        "symbol";
    "YUD";
}
"YUM";
{
    "displayName";
    "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-one";
    "Jugoslawischer Neuer Dinar (1994-2002)",
        "displayName-count-other";
    "Jugoslawische Neue Dinar (1994-2002)",
        "symbol";
    "YUM";
}
"YUN";
{
    "displayName";
    "Jugoslawischer Dinar (konvertibel)",
        "displayName-count-one";

```



```

        "Jugoslawische Dinar (konvertibel)",
        "displayName-count-other";
        "Jugoslawische Dinar (konvertibel)",
        "symbol";
        "YUN";
    }
    "YUR";
    {
        "displayName";
        "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-one";
        "Jugoslawischer reformierter Dinar (1992-1993)",
        "displayName-count-other";
        "Jugoslawische reformierte Dinar (1992-1993)",
        "symbol";
        "YUR";
    }
    "ZAL";
    {
        "displayName";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-one";
        "Südafrikanischer Rand (Finanz)",
        "displayName-count-other";
        "Südafrikanischer Rand (Finanz)",
        "symbol";
        "ZAL";
    }
    "ZAR";
    {
        "displayName";
        "Südafrikanischer Rand",
        "displayName-count-one";
        "Südafrikanischer Rand",
        "displayName-count-other";
        "Südafrikanische Rand",
        "symbol";
        "ZAR",
        "symbol-alt-narrow";
        "R";
    }
    "ZMK";
    {
        "displayName";
        "Kwacha (1968-2012)",
        "displayName-count-one";
        "Kwacha (1968-2012)",
        "displayName-count-other";
        "Kwacha (1968-2012)",
        "symbol";
        "ZMK";
    }
    "ZMW";
    {
        "displayName";
        "Kwacha",
        "displayName-count-one";

```

```

        "Kwacha",
        "displayName-count-other";
    "Kwacha",
    "symbol";
    "ZMW",
    "symbol-alt-narrow";
    "K";
}
"ZRN";
{
    "displayName";
    "Zaire-Neuer Zaïre (1993-1998)",
    "displayName-count-one";
    "Zaire-Neuer Zaïre (1993-1998)",
    "displayName-count-other";
    "Zaire-Neue Zaïre (1993-1998)",
    "symbol";
    "ZRN";
}
"ZRZ";
{
    "displayName";
    "Zaire-Zaïre (1971-1993)",
    "displayName-count-one";
    "Zaire-Zaïre (1971-1993)",
    "displayName-count-other";
    "Zaire-Zaïre (1971-1993)",
    "symbol";
    "ZRZ";
}
"ZWD";
{
    "displayName";
    "Simbabwe-Dollar (1980-2008)",
    "displayName-count-one";
    "Simbabwe-Dollar (1980-2008)",
    "displayName-count-other";
    "Simbabwe-Dollar (1980-2008)",
    "symbol";
    "ZWD";
}
"ZWL";
{
    "displayName";
    "Simbabwe-Dollar (2009)",
    "displayName-count-one";
    "Simbabwe-Dollar (2009)",
    "displayName-count-other";
    "Simbabwe-Dollar (2009)",
    "symbol";
    "ZWL";
}
"ZWR";
{
    "displayName";
    "Simbabwe-Dollar (2008)",
    "displayName-count-one";

```

```
        "Simbabwe-Dollar (2008)",  
          "displayName-count-other";  
      "Simbabwe-Dollar (2008)",  
        "symbol";  
    "ZWR";  
  }  
}  
}  
}
```

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
    'de': {
        'datetimepicker': {
            placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
            today: 'heute'
        }
    }
});
function App() {
    const dateValue = new Date("12/11/2017 1:00 AM");
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
    locale='de' />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load();
```

```

    'de': {
      'datetimepicker': {
        placeholder: 'Wählen Sie ein Datum und eine Uhrzeit aus',
        today: 'heute'
      }
    }
  });
function App() {
  const dateValue: Date = new Date("12/11/2017 1:00 AM");
  return <DateTimePickerComponent id="datetimepicker"
value={dateValue} locale='de' />;
}
ReactDOM.render(<App />, document.getElementById('element'));

```

NUMBERINGSYSTEMS.JSON

```

{
  "supplemental": {
    "version": {
      "_number": "$Revision: 12732 $",
      "_unicodeVersion": "9.0.0",
      "_cldrVersion": "31"
    },
    "numberingSystems": {
      "adlm": {
        "_digits": "ᲐᲑᲒᲓᲔᲕᲖᲗᲘᲙᲚᲛᲜᲝᲞᲟᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩᲪᲫᲬᲭᲮᲯᲰᲱᲲᲳᲴᲵᲶᲷᲸᲹᲺ᲻᲼ᲽᲾᲿ",
        "_type": "numeric"
      },
      "ahom": {
        "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱱᱲᱴᱶᱰᱵᱽ᱾᱿",
        "_type": "numeric"
      },
      "arab": {
        "_digits": "٠١٢٣٤٥٦٧٨٩",
        "_type": "numeric"
      },
      "arabext": {
        "_digits": "٠١٢٣٤٥٦٧٨٩",
        "_type": "numeric"
      },
      "armn": {
        "_rules": "armenian-upper",
        "_type": "algorithmic"
      },
      "armnlow": {
        "_rules": "armenian-lower",
        "_type": "algorithmic"
      },
      "bali": {
        "_digits": "ᬀᬁᬂᬃᬄᬅᬆᬇᬈᬉᬊᬋᬌᬍᬎᬏᬐᬑᬒᬓᬔᬕᬖᬗᬘᬙᬚᬛᬜᬝᬞᬟᬠᬡᬢᬣᬤᬥᬦᬧᬨᬩᬪᬫᬬᬭᬮᬯᬰᬱᬲᬳ᬴ᬵᬶᬷᬸᬹᬺᬻᬼᬽᬾᬿ",
        "_type": "numeric"
      },
      "beng": {
        "_digits": "০১২৩৪৫৬৭৮৯",
        "_type": "numeric"
      }
    }
  },

```

```
"bhks": {
  "_digits": "□□□□□□□□",
  "_type": "numeric"
},
"brah": {
  "_digits": "·\\۳۳۸۶۱۷۹",
  "_type": "numeric"
},
"cakm": {
  "_digits": "ᐐᐘᐃᐞᐞᐞᐞᐞᐞ",
  "_type": "numeric"
},
"cham": {
  "_digits": "□□□□□□□□",
  "_type": "numeric"
},
"cyrl": {
  "_rules": "cyrillic-lower",
  "_type": "algorithmic"
},
"deva": {
  "_digits": "ॐ३४५६७८९",
  "_type": "numeric"
},
"ethi": {
  "_rules": "ethiopic",
  "_type": "algorithmic"
},
"fullwide": {
  "_digits": "0 1 2 3 4 5 6 7 8 9",
  "_type": "numeric"
},
"geor": {
  "_rules": "georgian",
  "_type": "algorithmic"
},
"grek": {
  "_rules": "greek-upper",
  "_type": "algorithmic"
},
"greklow": {
  "_rules": "greek-lower",
  "_type": "algorithmic"
},
"gujr": {
  "_digits": "૦૧૨૩૪૫૬૭૮૯",
  "_type": "numeric"
},
"guru": {
  "_digits": "ᐐᐑᐃᐃᐞᐞᐞᐞᐞᐞ",
  "_type": "numeric"
},
"hanidays": {
  "_rules": "zh/SpelloutRules/spellout-numbering-days",
  "_type": "algorithmic"
}
```

```

    },
    "hanidec": {
      "_digits": "〇一二三四五六七八九",
      "_type": "numeric"
    },
    "hans": {
      "_rules": "zh/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hansfin": {
      "_rules": "zh/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hant": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hantfin": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hebr": {
      "_rules": "hebrew",
      "_type": "algorithmic"
    },
    "hmng": {
      "_digits": "□□□□□□□□□□",
      "_type": "numeric"
    },
    "java": {
      "_digits": "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉᐊᐋᐌᐍᐎᐏᐐᐑᐒᐓᐔᐕᐖᐗᐘᐙᐚᐛᐜᐝᐞᐟᐠᐡᐢᐣᐤᐥᐦᐧᐨᐩᐪᐫᐬᐭᐮᐯᐰᐱᐲᐳᐴᐵᐶᐷᐸᐹᐺᐻᐼᐽᐾᐿ",
      "_type": "numeric"
    },
    "jpan": {
      "_rules": "ja/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "jpanfin": {
      "_rules": "ja/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "kali": {
      "_digits": "□□□□□□□□□□",
      "_type": "numeric"
    },
    "khmr": {
      "_digits": "០១២៣៤៥៦៧៨៩",
      "_type": "numeric"
    },
    "knda": {
      "_digits": "೦೧೨೩೪೫೬೭೮೯",
      "_type": "numeric"
    },
    "lana": {
      "_digits": "□□□□□□□□□□",

```

```

    "_type": "numeric"
  },
  "lanatham": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "laoo": {
    "_digits": "໐໑໒໓໔໕໖໗໘໑",
    "_type": "numeric"
  },
  "latn": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "lepc": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "limb": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mathbold": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathdbl": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathmono": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsanb": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsans": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mlym": {
    "_digits": "൦൧൨൩൪൫൬൭൮൯",
    "_type": "numeric"
  },
  "modi": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mong": {
    "_digits": "᠐᠑᠒᠓᠐ᠠᠨᠨᠠᠭᠤᠯᠤᠰ",
    "_type": "numeric"
  },
  "mroo": {
    "_digits": "□□□□□□□□□□",

```

```

    "_type": "numeric"
  },
  "mtei": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "mymr": {
    "_digits": "၀၁၂၃၄၅၆၇၈",
    "_type": "numeric"
  },
  "mymrshan": {
    "_digits": "၀၁၂၃၄၅၆၇၈",
    "_type": "numeric"
  },
  "mymrtlng": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "newa": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "nkoo": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "olck": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "orya": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "osma": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "roman": {
    "_rules": "roman-upper",
    "_type": "algorithmic"
  },
  "romanlow": {
    "_rules": "roman-lower",
    "_type": "algorithmic"
  },
  "saur": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },
  "shrd": {
    "_digits": "ᐐᐐᐐᐐᐐᐐᐐᐐᐐᐐ",
    "_type": "numeric"
  },

```



```

"sind": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"sinh": {
  "_digits": "𑌀𑌁𑌂𑌃𑌄𑌅𑌆𑌇𑌈𑌉",
  "_type": "numeric"
},
"sora": {
  "_digits": "ᱚᱛᱟᱨᱛᱟᱱ",
  "_type": "numeric"
},
"sund": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"takr": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"taluk": {
  "_digits": "ᱠᱤᱨᱤᱰᱤᱦᱚᱱ",
  "_type": "numeric"
},
"taml": {
  "_rules": "tamil",
  "_type": "algorithmic"
},
"tamldec": {
  "_digits": "௦௧௨௩௪௫௬௭௮௯",
  "_type": "numeric"
},
"telu": {
  "_digits": "౦౧౨౩౪౫౬౭౮౯",
  "_type": "numeric"
},
"thai": {
  "_digits": "๐๑๒๓๔๕๖๗๘๙",
  "_type": "numeric"
},
"tibet": {
  "_digits": "༠༡༢༣༤༥༦༧༨༩",
  "_type": "numeric"
},
"tirh": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"vaii": {
  "_digits": "𞤵𞤶𞤷𞤸𞤹𞥀𞥁𞥂𞥃𞥄",
  "_type": "numeric"
},
"wara": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
}

```

```

    }
  }
}

```

NUMBERINGSYSTEMS.JSX

```

{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
    "numberingSystems";
    {
      "adlm";
      {
        "_digits";
        "ᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩ",
        "_type";
        "numeric";
      }
      "ahom";
      {
        "_digits";
        "ᩌᩍᩎᩏᩐᩑᩒᩓᩔᩕᩖ",
        "_type";
        "numeric";
      }
      "arab";
      {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
      }
      "arabext";
      {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
      }
      "armn";
      {
        "_rules";
        "armenian-upper",
        "_type";
        "algorithmic";
      }
    }
  }
}

```

```

    "armnlow";
    {
      "_rules";
      "armenian-lower",
      "_type";
      "algorithmic";
    }
    "bali";
    {
      "_digits";
      "ᮀᮁᮂᮃᮄᮅᮆᮇᮈᮉ",
      "_type";
      "numeric";
    }
    "beng";
    {
      "_digits";
      "০১২৩৪৫৬৭৮৯",
      "_type";
      "numeric";
    }
    "bhks";
    {
      "_digits";
      "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
      "_type";
      "numeric";
    }
    "brah";
    {
      "_digits";
      "ᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱴᱶᱟ",
      "_type";
      "numeric";
    }
    "cakm";
    {
      "_digits";
      "ᨀᨁᨂᨃᨄᨅᨆᨇᨈᨉᨐᨑᨒᨓᨔᨕᨖᨘᨗᨙᨚᨛ᨜᨝᨞᨟ᨠᨡᨢᨣᨤᨥᨦᨧᨨᨩᨪᨫᨬᨭᨮᨯᨰᨱᨲᨳᨴᨵᨶᨷᨸᨹᨺᨻᨼᨽᨾᨿ",
      "_type";
      "numeric";
    }
    "cham";
    {
      "_digits";
      "ᦅᦆᦇᦈᦉᦊᦍᦏᦐᦑᦒᦓᦔᦕᦖᦗᦘᦙᦚᦛᦞᦟᦠᦡᦢᦣᦤᦥᦦᦧᦨᦩᦪᦫ᦬᦭᦮᦯ᦰᦱᦲᦳᦴᦵᦶᦷᦸᦹᦺᦻᦼᦽᦾᦿ",
      "_type";
      "numeric";
    }
    "cyr1";
    {
      "_rules";
      "cyrillic-lower",
      "_type";
      "algorithmic";
    }
  }

```

```
"deva";
{
  "_digits";
  "୦୧୨୩୪୫୬୭୮୯",
  "_type";
  "numeric";
}
"ethi";
{
  "_rules";
  "ethiopic",
  "_type";
  "algorithmic";
}
"fullwide";
{
  "_digits";
  "0 1 2 3 4 5 6 7 8 9",
  "_type";
  "numeric";
}
"geor";
{
  "_rules";
  "georgian",
  "_type";
  "algorithmic";
}
"grek";
{
  "_rules";
  "greek-upper",
  "_type";
  "algorithmic";
}
"greklow";
{
  "_rules";
  "greek-lower",
  "_type";
  "algorithmic";
}
"gujr";
{
  "_digits";
  "୦୧୨୩୪୫୬୭୮୯",
  "_type";
  "numeric";
}
"guru";
{
  "_digits";
  "୦୧୨୩୪୫୬୭୮୯",
  "_type";
  "numeric";
}
```

```

"hanidays";
{
  "_rules";
  "zh/SpelloutRules/spellout-numbering-days",
  "_type";
  "algorithmic";
}
"hanidec";
{
  "_digits";
  "〇一二三四五六七八九",
  "_type";
  "numeric";
}
"hans";
{
  "_rules";
  "zh/SpelloutRules/spellout-cardinal",
  "_type";
  "algorithmic";
}
"hansfin";
{
  "_rules";
  "zh/SpelloutRules/spellout-cardinal-financial",
  "_type";
  "algorithmic";
}
"hant";
{
  "_rules";
  "zh_Hant/SpelloutRules/spellout-cardinal",
  "_type";
  "algorithmic";
}
"hantfin";
{
  "_rules";
  "zh_Hant/SpelloutRules/spellout-cardinal-financial",
  "_type";
  "algorithmic";
}
"hebr";
{
  "_rules";
  "hebrew",
  "_type";
  "algorithmic";
}
"hmng";
{
  "_digits";
  "□□□□□□□□□□",
  "_type";
  "numeric";
}

```

```

"java";
{
  "_digits";
  "௦௧௨௩௪௫௬௭௮௯௦",
  "_type";
  "numeric";
}
"jpan";
{
  "_rules";
  "ja/SpelloutRules/spellout-cardinal",
  "_type";
  "algorithmic";
}
"jpanfin";
{
  "_rules";
  "ja/SpelloutRules/spellout-cardinal-financial",
  "_type";
  "algorithmic";
}
"kali";
{
  "_digits";
  "௦௧௨௩௪௫௬௭௮௯௦",
  "_type";
  "numeric";
}
"khmr";
{
  "_digits";
  "០១២៣៤៥៦៧៨៩",
  "_type";
  "numeric";
}
"knda";
{
  "_digits";
  "೦೧೨೩೪೫೬೭೮೯",
  "_type";
  "numeric";
}
"lana";
{
  "_digits";
  "௦௧௨௩௪௫௬௭௮௯௦",
  "_type";
  "numeric";
}
"lanatham";
{
  "_digits";
  "௦௧௨௩௪௫௬௭௮௯௦",
  "_type";
  "numeric";
}

```

```
}
"laoo";
{
  "_digits";
  "໐໑໒໓໔໕໖໗໘໑",
  "_type";
  "numeric";
}
"latn";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"lepc";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"limb";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"mathbold";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathdbl";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathmono";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mathsanb";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
```

```
}
"mathsans";
{
  "_digits";
  "0123456789",
  "_type";
  "numeric";
}
"mlym";
{
  "_digits";
  "ഘറനർത്ഥനവുൻ",
  "_type";
  "numeric";
}
"modi";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"mong";
{
  "_digits";
  "᠐᠑᠒᠓᠐ᠠᠨᠨᠠᠭ",
  "_type";
  "numeric";
}
"mroo";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"mtei";
{
  "_digits";
  "᠐᠑ᠶᠢᠨᠵᠢᠨᠵᠢᠨᠵᠢ",
  "_type";
  "numeric";
}
"mymr";
{
  "_digits";
  "၀၁၂၃၄၅၆၇၈၉",
  "_type";
  "numeric";
}
"mymrshan";
{
  "_digits";
  "01၇၈၉၀၁၂၃၄၅၆၇၈",
  "_type";
```



```

    "numeric";
}
"mymrtlng";
{
    "_digits";
    "0୫୪3୩6୨୧୬୦",
    "_type";
    "numeric";
}
"newa";
{
    "_digits";
    "୦୦୦୦୦୦୦୦୦୦",
    "_type";
    "numeric";
}
"nkoo";
{
    "_digits";
    "୨୯୯୯୯୯୯୯୯୯୦",
    "_type";
    "numeric";
}
"olck";
{
    "_digits";
    "୦୯୯୯୯୯୯୯୯୯୯",
    "_type";
    "numeric";
}
"orya";
{
    "_digits";
    "୦୧୨୩୪୫୬୭୮୯",
    "_type";
    "numeric";
}
"osma";
{
    "_digits";
    "୦୧୨୩୪୫୬୭୮୯",
    "_type";
    "numeric";
}
"roman";
{
    "_rules";
    "roman-upper",
    "_type";
    "algorithmic";
}
"romanlow";
{
    "_rules";
    "roman-lower"

```

```

        "_type";
        "algorithmic";
    }
    "saur";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "shrd";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sind";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sinh";
    {
        "_digits";
        "⁂௧௧௧௧௧௧௧௧௧",
        "_type";
        "numeric";
    }
    "sora";
    {
        "_digits";
        "0௧௧௧௧௧௧௧௧",
        "_type";
        "numeric";
    }
    "sund";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "takr";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "talv";
    {
        "_digits";
        "⁂௧௧௧௧௧௧௧௧",

```

```

        "_type";
        "numeric";
    }
    "taml";
    {
        "_rules";
        "tamil",
        "_type";
        "algorithmic";
    }
    "tamldec";
    {
        "_digits";
        "0௧௨௩௪௫௬௭௮௯",
        "_type";
        "numeric";
    }
    "telu";
    {
        "_digits";
        "౦౧౨౩౪౫౬౭౮౯",
        "_type";
        "numeric";
    }
    "thai";
    {
        "_digits";
        "๐๑๒๓๔๕๖๗๘๙",
        "_type";
        "numeric";
    }
    "tibb";
    {
        "_digits";
        "༠༡༢༣༤༥༦༧༨༩",
        "_type";
        "numeric";
    }
    "tirh";
    {
        "_digits";
        "౦౧౨౩౪౫౬౭౮౯",
        "_type";
        "numeric";
    }
    "vaih";
    {
        "_digits";
        "౦౧౨౩౪౫౬౭౮౯",
        "_type";
        "numeric";
    }
    "wara";
    {
        "_digits";
        "౦౧౨౩౪౫౬౭౮౯",

```

```

        "_type":
        "numeric";
    }
}
}
}

```

NUMBERS.JSON

```

{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 13259 $",
          "_cldrVersion": "31"
        },
        "language": "de"
      },
      "numbers": {
        "defaultNumberingSystem": "latn",
        "otherNumberingSystems": {
          "native": "latn"
        },
        "minimumGroupingDigits": "1",
        "symbols-numberSystem-latn": {
          "decimal": ",",
          "group": ".",
          "list": ";",
          "percentSign": "%",
          "plusSign": "+",
          "minusSign": "-",
          "exponential": "E",
          "superscriptingExponent": "·",
          "perMille": "‰",
          "infinity": "∞",
          "nan": "NaN",
          "timeSeparator": ":"
        },
        "decimalFormats-numberSystem-latn": {
          "standard": "#,##0.###",
          "long": {
            "decimalFormat": {
              "1000-count-one": "0 Tausend",
              "1000-count-other": "0 Tausend",
              "10000-count-one": "00 Tausend",
              "10000-count-other": "00 Tausend",
              "100000-count-one": "000 Tausend",
              "100000-count-other": "000 Tausend",
              "1000000-count-one": "0 Million",
              "1000000-count-other": "0 Millionen",
              "10000000-count-one": "00 Millionen",
              "10000000-count-other": "00 Millionen",
              "100000000-count-one": "000 Millionen",
              "100000000-count-other": "000 Millionen",
              "1000000000-count-one": "0 Milliarde",

```

```

        "1000000000-count-other": "0 Milliarden",
        "1000000000-count-one": "00 Milliarden",
        "1000000000-count-other": "00 Milliarden",
        "1000000000-count-one": "000 Milliarden",
        "10000000000-count-other": "000 Milliarden",
        "10000000000-count-one": "0 Billion",
        "100000000000-count-other": "0 Billionen",
        "100000000000-count-one": "00 Billionen",
        "1000000000000-count-other": "00 Billionen",
        "1000000000000-count-one": "000 Billionen",
        "10000000000000-count-other": "000 Billionen"
    }
},
"short": {
    "decimalFormat": {
        "1000-count-one": "0",
        "1000-count-other": "0",
        "10000-count-one": "0",
        "10000-count-other": "0",
        "100000-count-one": "0",
        "100000-count-other": "0",
        "1000000-count-one": "0 Mio'.'",
        "1000000-count-other": "0 Mio'.'",
        "10000000-count-one": "00 Mio'.'",
        "10000000-count-other": "00 Mio'.'",
        "100000000-count-one": "000 Mio'.'",
        "100000000-count-other": "000 Mio'.'",
        "1000000000-count-one": "0 Mrd'.'",
        "1000000000-count-other": "0 Mrd'.'",
        "10000000000-count-one": "00 Mrd'.'",
        "10000000000-count-other": "00 Mrd'.'",
        "100000000000-count-one": "000 Mrd'.'",
        "100000000000-count-other": "000 Mrd'.'",
        "1000000000000-count-one": "0 Bio'.'",
        "1000000000000-count-other": "0 Bio'.'",
        "10000000000000-count-one": "00 Bio'.'",
        "10000000000000-count-other": "00 Bio'.'",
        "100000000000000-count-one": "000 Bio'.'",
        "100000000000000-count-other": "000 Bio'.'"
    }
}
},
"scientificFormats-numberSystem-latn": {
    "standard": "#E0"
},
"percentFormats-numberSystem-latn": {
    "standard": "#,##0 %"
},
"currencyFormats-numberSystem-latn": {
    "currencySpacing": {
        "beforeCurrency": {
            "currencyMatch": "[:^S:]",
            "surroundingMatch": "[:digit:]",
            "insertBetween": " "
        },
        "afterCurrency": {
            "currencyMatch": "[:^S:]",

```

```

    "surroundingMatch": "[[:digit:]]",
    "insertBetween": " "
  }
},
"standard": "#,##0.00 ¤",
"accounting": "#,##0.00 ¤",
"short": {
  "standard": {
    "1000-count-one": "0 Tsd'.' ¤",
    "1000-count-other": "0 Tsd'.' ¤",
    "10000-count-one": "00 Tsd'.' ¤",
    "10000-count-other": "00 Tsd'.' ¤",
    "100000-count-one": "000 Tsd'.' ¤",
    "100000-count-other": "000 Tsd'.' ¤",
    "1000000-count-one": "0 Mio'.' ¤",
    "1000000-count-other": "0 Mio'.' ¤",
    "10000000-count-one": "00 Mio'.' ¤",
    "10000000-count-other": "00 Mio'.' ¤",
    "100000000-count-one": "000 Mio'.' ¤",
    "100000000-count-other": "000 Mio'.' ¤",
    "1000000000-count-one": "0 Mrd'.' ¤",
    "1000000000-count-other": "0 Mrd'.' ¤",
    "10000000000-count-one": "00 Mrd'.' ¤",
    "10000000000-count-other": "00 Mrd'.' ¤",
    "100000000000-count-one": "000 Mrd'.' ¤",
    "100000000000-count-other": "000 Mrd'.' ¤",
    "1000000000000-count-one": "0 Bio'.' ¤",
    "1000000000000-count-other": "0 Bio'.' ¤",
    "10000000000000-count-one": "00 Bio'.' ¤",
    "10000000000000-count-other": "00 Bio'.' ¤",
    "100000000000000-count-one": "000 Bio'.' ¤",
    "100000000000000-count-other": "000 Bio'.' ¤"
  },
  "unitPattern-count-one": "{0} {1}",
  "unitPattern-count-other": "{0} {1}"
},
"miscPatterns-numberSystem-latn": {
  "atLeast": "{0}+",
  "range": "{0}-{1}"
},
"minimalPairs": {
  "pluralMinimalPairs": "{0} Tag",
  "pluralMinimalPairs": "{0} Tage",
  "other": "{0}. Abzweigung nach rechts nehmen"
}
}
}
}
}

```

NUMBERS.JSX

```
{
    "main";
}
```

```

    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13259 $",
          "_cldrVersion";
          "31";
        }
        "language";
        "de";
      }
    }
    "numbers";
    {
      "defaultNumberingSystem";
      "latn",
      "otherNumberingSystems";
      {
        "native";
        "latn";
      }
      "minimumGroupingDigits";
      "1",
      "symbols-numberSystem-latn";
      {
        "decimal";
        ",",
        "group";
        ".",
        "list";
        ";",
        "percentSign";
        "%",
        "plusSign";
        "+",
        "minusSign";
        "-",
        "exponential";
        "E",
        "superscriptingExponent";
        ". ",
        "perMille";
        "‰",
        "infinity";
        "∞",
        "nan";
        "NaN",
        "timeSeparator";
        ":";
      }
      "decimalFormats-numberSystem-latn";
      {
        "standard";
        "#,##0.###",
        "long";
      }
    }
  }
}

```

```

    {
      "decimalFormat";
      {
        "1000-count-one";
        "0 Tausend",
        "1000-count-other";
        "0 Tausend",
        "10000-count-one";
        "00 Tausend",
        "10000-count-other";
        "00 Tausend",
        "100000-count-one";
        "000 Tausend",
        "100000-count-other";
        "000 Tausend",
        "1000000-count-one";
        "0 Million",
        "1000000-count-other";
        "0 Millionen",
        "10000000-count-one";
        "00 Millionen",
        "10000000-count-other";
        "00 Millionen",
        "100000000-count-one";
        "000 Millionen",
        "100000000-count-other";
        "000 Millionen",
        "1000000000-count-one";
        "0 Milliarde",
        "1000000000-count-other";
        "0 Milliarden",
        "10000000000-count-one";
        "00 Milliarden",
        "10000000000-count-other";
        "00 Milliarden",
        "100000000000-count-one";
        "000 Milliarden",
        "100000000000-count-other";
        "000 Milliarden",
        "1000000000000-count-one";
        "0 Billion",
        "1000000000000-count-other";
        "0 Billionen",
        "10000000000000-count-one";
        "00 Billionen",
        "10000000000000-count-other";
        "00 Billionen",
        "100000000000000-count-one";
        "000 Billionen",
        "100000000000000-count-other";
        "000 Billionen";
      }
    }
    "short";
    {
      "decimalFormat";
      {

```



```

        "1000-count-one";
        "0",
        "1000-count-other";
        "0",
        "10000-count-one";
        "0",
        "10000-count-other";
        "0",
        "100000-count-one";
        "0",
        "100000-count-other";
        "0",
        "1000000-count-one";
        "0 Mio'.'",
        "1000000-count-other";
        "0 Mio'.'",
        "10000000-count-one";
        "00 Mio'.'",
        "10000000-count-other";
        "00 Mio'.'",
        "100000000-count-one";
        "000 Mio'.'",
        "100000000-count-other";
        "000 Mio'.'",
        "1000000000-count-one";
        "0 Mrd'.'",
        "1000000000-count-other";
        "0 Mrd'.'",
        "10000000000-count-one";
        "00 Mrd'.'",
        "10000000000-count-other";
        "00 Mrd'.'",
        "100000000000-count-one";
        "000 Mrd'.'",
        "100000000000-count-other";
        "000 Mrd'.'",
        "1000000000000-count-one";
        "0 Bio'.'",
        "1000000000000-count-other";
        "0 Bio'.'",
        "10000000000000-count-one";
        "00 Bio'.'",
        "10000000000000-count-other";
        "00 Bio'.'",
        "100000000000000-count-one";
        "000 Bio'.'",
        "100000000000000-count-other";
        "000 Bio'.'";
    }
}
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}
"percentFormats-numberSystem-latn";

```

```

    {
      "standard";
      "#,##0 %";
    }
    "currencyFormats-numberSystem-latn";
    {
      "currencySpacing";
      {
        "beforeCurrency";
        {
          "currencyMatch";
          "[:^S:]",
          "surroundingMatch";
          "[:digit:]",
          "insertBetween";
          " ";
        }
        "afterCurrency";
        {
          "currencyMatch";
          "[:^S:]",
          "surroundingMatch";
          "[:digit:]",
          "insertBetween";
          " ";
        }
      }
    }
    "standard";
    "#,##0.00 ¤",
    "accounting";
    "#,##0.00 ¤",
    "short";
    {
      "standard";
      {
        "1000-count-one";
        "0 Tsd'.' ¤",
        "1000-count-other";
        "0 Tsd'.' ¤",
        "10000-count-one";
        "00 Tsd'.' ¤",
        "10000-count-other";
        "00 Tsd'.' ¤",
        "100000-count-one";
        "000 Tsd'.' ¤",
        "100000-count-other";
        "000 Tsd'.' ¤",
        "1000000-count-one";
        "0 Mio'.' ¤",
        "1000000-count-other";
        "0 Mio'.' ¤",
        "10000000-count-one";
        "00 Mio'.' ¤",
        "10000000-count-other";
        "00 Mio'.' ¤",
        "100000000-count-one";
        "000 Mio'.' ¤",
      }
    }
  }

```

```

        "100000000-count-other";
        "000 Mio'.' s",
        "1000000000-count-one";
        "0 Mrd'.' s",
        "1000000000-count-other";
        "0 Mrd'.' s",
        "10000000000-count-one";
        "00 Mrd'.' s",
        "10000000000-count-other";
        "00 Mrd'.' s",
        "100000000000-count-one";
        "000 Mrd'.' s",
        "100000000000-count-other";
        "000 Mrd'.' s",
        "1000000000000-count-one";
        "0 Bio'.' s",
        "1000000000000-count-other";
        "0 Bio'.' s",
        "10000000000000-count-one";
        "00 Bio'.' s",
        "10000000000000-count-other";
        "00 Bio'.' s",
        "100000000000000-count-one";
        "000 Bio'.' s",
        "100000000000000-count-other";
        "000 Bio'.' s";
    }
}
"unitPattern-count-one";
"{0} {1}",
    "unitPattern-count-other";
"{0} {1}";
}
"miscPatterns-numberSystem-latn";
{
    "atLeast";
    "{0}+",
        "range";
    "{0}-{1}";
}
"minimalPairs";
{
    "pluralMinimalPairs";
    "{0} Tag",
        "pluralMinimalPairs";
    "{0} Tage",
        "other";
    "{0}. Abzweigung nach rechts nehmen";
}
}
}
}

```

TIMEZONENAMES.JSON

```

{
  "main": {
    "de": {
      "identity": {
        "version": {
          "_number": "$Revision: 12879 $",
          "_cldrVersion": "30.0.3"
        },
        "language": "de"
      },
      "dates": {
        "timeZoneNames": {
          "hourFormat": "+HH:mm;-HH:mm",
          "gmtFormat": "GMT{0}",
          "gmtZeroFormat": "GMT",
          "regionFormat": "{0} Zeit",
          "regionFormat-type-daylight": "{0} Sommerzeit",
          "regionFormat-type-standard": "{0} Normalzeit",
          "fallbackFormat": "{1} ({0})",
          "zone": {
            "America": {
              "Adak": {
                "exemplarCity": "Adak"
              },
              "Anchorage": {
                "exemplarCity": "Anchorage"
              },
              "Anguilla": {
                "exemplarCity": "Anguilla"
              },
              "Antigua": {
                "exemplarCity": "Antigua"
              },
              "Araguaina": {
                "exemplarCity": "Araguaina"
              },
              "Argentina": {
                "Rio_Gallegos": {
                  "exemplarCity": "Rio Gallegos"
                },
                "San_Juan": {
                  "exemplarCity": "San Juan"
                },
                "Ushuaia": {
                  "exemplarCity": "Ushuaia"
                },
                "La_Rioja": {
                  "exemplarCity": "La Rioja"
                },
                "San_Luis": {
                  "exemplarCity": "San Luis"
                },
                "Salta": {
                  "exemplarCity": "Salta"
                },
                "Tucuman": {
                  "exemplarCity": "Tucuman"
                }
              }
            }
          }
        }
      }
    }
  }
}

```

```
    }  
  },  
  "Aruba": {  
    "exemplarCity": "Aruba"  
  },  
  "Asuncion": {  
    "exemplarCity": "Asunción"  
  },  
  "Bahia": {  
    "exemplarCity": "Bahia"  
  },  
  "Bahia_Banderas": {  
    "exemplarCity": "Bahia Banderas"  
  },  
  "Barbados": {  
    "exemplarCity": "Barbados"  
  },  
  "Belem": {  
    "exemplarCity": "Belem"  
  },  
  "Belize": {  
    "exemplarCity": "Belize"  
  },  
  "Blanc-Sablon": {  
    "exemplarCity": "Blanc-Sablon"  
  },  
  "Boa_Vista": {  
    "exemplarCity": "Boa Vista"  
  },  
  "Bogota": {  
    "exemplarCity": "Bogotá"  
  },  
  "Boise": {  
    "exemplarCity": "Boise"  
  },  
  "Buenos_Aires": {  
    "exemplarCity": "Buenos Aires"  
  },  
  "Cambridge_Bay": {  
    "exemplarCity": "Cambridge Bay"  
  },  
  "Campo_Grande": {  
    "exemplarCity": "Campo Grande"  
  },  
  "Cancun": {  
    "exemplarCity": "Cancún"  
  },  
  "Caracas": {  
    "exemplarCity": "Caracas"  
  },  
  "Catamarca": {  
    "exemplarCity": "Catamarca"  
  },  
  "Cayenne": {  
    "exemplarCity": "Cayenne"  
  },  
  "Cayman": {
```

```
    "exemplarCity": "Kaimaninseln"
  },
  "Chicago": {
    "exemplarCity": "Chicago"
  },
  "Chihuahua": {
    "exemplarCity": "Chihuahua"
  },
  "Coral_Harbour": {
    "exemplarCity": "Atikokan"
  },
  "Cordoba": {
    "exemplarCity": "Córdoba"
  },
  "Costa_Rica": {
    "exemplarCity": "Costa Rica"
  },
  "Creston": {
    "exemplarCity": "Creston"
  },
  "Cuiaba": {
    "exemplarCity": "Cuiaba"
  },
  "Curacao": {
    "exemplarCity": "Curaçao"
  },
  "Danmarkshavn": {
    "exemplarCity": "Danmarkshavn"
  },
  "Dawson": {
    "exemplarCity": "Dawson"
  },
  "Dawson_Creek": {
    "exemplarCity": "Dawson Creek"
  },
  "Denver": {
    "exemplarCity": "Denver"
  },
  "Detroit": {
    "exemplarCity": "Detroit"
  },
  "Dominica": {
    "exemplarCity": "Dominica"
  },
  "Edmonton": {
    "exemplarCity": "Edmonton"
  },
  "Eirunepe": {
    "exemplarCity": "Eirunepe"
  },
  "El_Salvador": {
    "exemplarCity": "El Salvador"
  },
  "Fort_Nelson": {
    "exemplarCity": "Fort Nelson"
  },
  "Fortaleza": {
```

```
    "exemplarCity": "Fortaleza"
  },
  "Glace_Bay": {
    "exemplarCity": "Glace Bay"
  },
  "Godthab": {
    "exemplarCity": "Nuuk"
  },
  "Goose_Bay": {
    "exemplarCity": "Goose Bay"
  },
  "Grand_Turk": {
    "exemplarCity": "Grand Turk"
  },
  "Grenada": {
    "exemplarCity": "Grenada"
  },
  "Guadeloupe": {
    "exemplarCity": "Guadeloupe"
  },
  "Guatemala": {
    "exemplarCity": "Guatemala"
  },
  "Guayaquil": {
    "exemplarCity": "Guayaquil"
  },
  "Guyana": {
    "exemplarCity": "Guyana"
  },
  "Halifax": {
    "exemplarCity": "Halifax"
  },
  "Havana": {
    "exemplarCity": "Havanna"
  },
  "Hermosillo": {
    "exemplarCity": "Hermosillo"
  },
  "Indiana": {
    "Vincennes": {
      "exemplarCity": "Vincennes, Indiana"
    },
    "Petersburg": {
      "exemplarCity": "Petersburg, Indiana"
    },
    "Tell_City": {
      "exemplarCity": "Tell City, Indiana"
    },
    "Knox": {
      "exemplarCity": "Knox, Indiana"
    },
    "Winamac": {
      "exemplarCity": "Winamac, Indiana"
    },
    "Marengo": {
      "exemplarCity": "Marengo, Indiana"
    }
  },
}
```

```
    "Vevay": {
      "exemplarCity": "Vevay, Indiana"
    },
    "Indianapolis": {
      "exemplarCity": "Indianapolis"
    },
    "Inuvik": {
      "exemplarCity": "Inuvik"
    },
    "Iqaluit": {
      "exemplarCity": "Iqaluit"
    },
    "Jamaica": {
      "exemplarCity": "Jamaika"
    },
    "Jujuy": {
      "exemplarCity": "Jujuy"
    },
    "Juneau": {
      "exemplarCity": "Juneau"
    },
    "Kentucky": {
      "Monticello": {
        "exemplarCity": "Monticello, Kentucky"
      }
    },
    "Kralendijk": {
      "exemplarCity": "Kralendijk"
    },
    "La_Paz": {
      "exemplarCity": "La Paz"
    },
    "Lima": {
      "exemplarCity": "Lima"
    },
    "Los_Angeles": {
      "exemplarCity": "Los Angeles"
    },
    "Louisville": {
      "exemplarCity": "Louisville"
    },
    "Lower_Princes": {
      "exemplarCity": "Lower Prince's Quarter"
    },
    "Maceio": {
      "exemplarCity": "Maceio"
    },
    "Managua": {
      "exemplarCity": "Managua"
    },
    "Manaus": {
      "exemplarCity": "Manaus"
    },
    "Marigot": {
      "exemplarCity": "Marigot"
    },
  },
```



```
"Martinique": {
  "exemplarCity": "Martinique"
},
"Matamoros": {
  "exemplarCity": "Matamoros"
},
"Mazatlan": {
  "exemplarCity": "Mazatlan"
},
"Mendoza": {
  "exemplarCity": "Mendoza"
},
"Menominee": {
  "exemplarCity": "Menominee"
},
"Merida": {
  "exemplarCity": "Merida"
},
"Metlakatla": {
  "exemplarCity": "Metlakatla"
},
"Mexico_City": {
  "exemplarCity": "Mexiko-Stadt"
},
"Miquelon": {
  "exemplarCity": "Miquelon"
},
"Moncton": {
  "exemplarCity": "Moncton"
},
"Monterrey": {
  "exemplarCity": "Monterrey"
},
"Montevideo": {
  "exemplarCity": "Montevideo"
},
"Montserrat": {
  "exemplarCity": "Montserrat"
},
"Nassau": {
  "exemplarCity": "Nassau"
},
"New_York": {
  "exemplarCity": "New York"
},
"Nipigon": {
  "exemplarCity": "Nipigon"
},
"Nome": {
  "exemplarCity": "Nome"
},
"Noronha": {
  "exemplarCity": "Noronha"
},
"North_Dakota": {
  "Beulah": {
    "exemplarCity": "Beulah, North Dakota"
  }
}
```

```
    },  
    "New_Salem": {  
      "exemplarCity": "New Salem, North Dakota"  
    },  
    "Center": {  
      "exemplarCity": "Center, North Dakota"  
    }  
  },  
  "Ojinaga": {  
    "exemplarCity": "Ojinaga"  
  },  
  "Panama": {  
    "exemplarCity": "Panama"  
  },  
  "Pangnirtung": {  
    "exemplarCity": "Pangnirtung"  
  },  
  "Paramaribo": {  
    "exemplarCity": "Paramaribo"  
  },  
  "Phoenix": {  
    "exemplarCity": "Phoenix"  
  },  
  "Port-au-Prince": {  
    "exemplarCity": "Port-au-Prince"  
  },  
  "Port_of_Spain": {  
    "exemplarCity": "Port of Spain"  
  },  
  "Porto_Velho": {  
    "exemplarCity": "Porto Velho"  
  },  
  "Puerto_Rico": {  
    "exemplarCity": "Puerto Rico"  
  },  
  "Rainy_River": {  
    "exemplarCity": "Rainy River"  
  },  
  "Rankin_Inlet": {  
    "exemplarCity": "Rankin Inlet"  
  },  
  "Recife": {  
    "exemplarCity": "Recife"  
  },  
  "Regina": {  
    "exemplarCity": "Regina"  
  },  
  "Resolute": {  
    "exemplarCity": "Resolute"  
  },  
  "Rio_Branco": {  
    "exemplarCity": "Rio Branco"  
  },  
  "Santa_Isabel": {  
    "exemplarCity": "Santa Isabel"  
  },  
  "Santarem": {
```

```
        "exemplarCity": "Santarem"
      },
      "Santiago": {
        "exemplarCity": "Santiago"
      },
      "Santo_Domingo": {
        "exemplarCity": "Santo Domingo"
      },
      "Sao_Paulo": {
        "exemplarCity": "São Paulo"
      },
      "Scoresbysund": {
        "exemplarCity": "Ittoqqortoormiit"
      },
      "Sitka": {
        "exemplarCity": "Sitka"
      },
      "St_Barthelemy": {
        "exemplarCity": "Saint-Barthélemy"
      },
      "St_Johns": {
        "exemplarCity": "St. John's"
      },
      "St_Kitts": {
        "exemplarCity": "St. Kitts"
      },
      "St_Lucia": {
        "exemplarCity": "St. Lucia"
      },
      "St_Thomas": {
        "exemplarCity": "St. Thomas"
      },
      "St_Vincent": {
        "exemplarCity": "St. Vincent"
      },
      "Swift_Current": {
        "exemplarCity": "Swift Current"
      },
      "Tegucigalpa": {
        "exemplarCity": "Tegucigalpa"
      },
      "Thule": {
        "exemplarCity": "Thule"
      },
      "Thunder_Bay": {
        "exemplarCity": "Thunder Bay"
      },
      "Tijuana": {
        "exemplarCity": "Tijuana"
      },
      "Toronto": {
        "exemplarCity": "Toronto"
      },
      "Tortola": {
        "exemplarCity": "Tortola"
      },
      "Vancouver": {
```

```

        "exemplarCity": "Vancouver"
    },
    "Whitehorse": {
        "exemplarCity": "Whitehorse"
    },
    "Winnipeg": {
        "exemplarCity": "Winnipeg"
    },
    "Yakutat": {
        "exemplarCity": "Yakutat"
    },
    "Yellowknife": {
        "exemplarCity": "Yellowknife"
    }
},
"Atlantic": {
    "Azores": {
        "exemplarCity": "Azoren"
    },
    "Bermuda": {
        "exemplarCity": "Bermudas"
    },
    "Canary": {
        "exemplarCity": "Kanaren"
    },
    "Cape_Verde": {
        "exemplarCity": "Cabo Verde"
    },
    "Faeroe": {
        "exemplarCity": "Färöer"
    },
    "Madeira": {
        "exemplarCity": "Madeira"
    },
    "Reykjavik": {
        "exemplarCity": "Reykjavík"
    },
    "South_Georgia": {
        "exemplarCity": "Südgeorgien"
    },
    "St_Helena": {
        "exemplarCity": "St. Helena"
    },
    "Stanley": {
        "exemplarCity": "Stanley"
    }
},
"Europe": {
    "Amsterdam": {
        "exemplarCity": "Amsterdam"
    },
    "Andorra": {
        "exemplarCity": "Andorra"
    },
    "Astrakhan": {
        "exemplarCity": "Astrachan"
    }
},

```

```
"Athens": {
  "exemplarCity": "Athen"
},
"Belgrade": {
  "exemplarCity": "Belgrad"
},
"Berlin": {
  "exemplarCity": "Berlin"
},
"Bratislava": {
  "exemplarCity": "Bratislava"
},
"Brussels": {
  "exemplarCity": "Brüssel"
},
"Bucharest": {
  "exemplarCity": "Bukarest"
},
"Budapest": {
  "exemplarCity": "Budapest"
},
"Busingen": {
  "exemplarCity": "Büsingen"
},
"Chisinau": {
  "exemplarCity": "Kischinau"
},
"Copenhagen": {
  "exemplarCity": "Kopenhagen"
},
"Dublin": {
  "long": {
    "daylight": "Irische Sommerzeit"
  },
  "exemplarCity": "Dublin"
},
"Gibraltar": {
  "exemplarCity": "Gibraltar"
},
"Guernsey": {
  "exemplarCity": "Guernsey"
},
"Helsinki": {
  "exemplarCity": "Helsinki"
},
"Isle_of_Man": {
  "exemplarCity": "Isle of Man"
},
"Istanbul": {
  "exemplarCity": "Istanbul"
},
"Jersey": {
  "exemplarCity": "Jersey"
},
"Kaliningrad": {
  "exemplarCity": "Kaliningrad"
},
```

```
"Kiev": {
  "exemplarCity": "Kiew"
},
"Kirov": {
  "exemplarCity": "Kirow"
},
"Lisbon": {
  "exemplarCity": "Lissabon"
},
"Ljubljana": {
  "exemplarCity": "Ljubljana"
},
"London": {
  "long": {
    "daylight": "Britische Sommerzeit"
  },
  "exemplarCity": "London"
},
"Luxembourg": {
  "exemplarCity": "Luxemburg"
},
"Madrid": {
  "exemplarCity": "Madrid"
},
"Malta": {
  "exemplarCity": "Malta"
},
"Mariehamn": {
  "exemplarCity": "Mariehamn"
},
"Minsk": {
  "exemplarCity": "Minsk"
},
"Monaco": {
  "exemplarCity": "Monaco"
},
"Moscow": {
  "exemplarCity": "Moskau"
},
"Oslo": {
  "exemplarCity": "Oslo"
},
"Paris": {
  "exemplarCity": "Paris"
},
"Podgorica": {
  "exemplarCity": "Podgorica"
},
"Prague": {
  "exemplarCity": "Prag"
},
"Riga": {
  "exemplarCity": "Riga"
},
"Rome": {
  "exemplarCity": "Rom"
},
```

```
"Samara": {
  "exemplarCity": "Samara"
},
"San_Marino": {
  "exemplarCity": "San Marino"
},
"Sarajevo": {
  "exemplarCity": "Sarajevo"
},
"Simferopol": {
  "exemplarCity": "Simferopol"
},
"Skopje": {
  "exemplarCity": "Skopje"
},
"Sofia": {
  "exemplarCity": "Sofia"
},
"Stockholm": {
  "exemplarCity": "Stockholm"
},
"Tallinn": {
  "exemplarCity": "Tallinn"
},
"Tirane": {
  "exemplarCity": "Tirana"
},
"Ulyanovsk": {
  "exemplarCity": "Uljanowsk"
},
"Uzhgorod": {
  "exemplarCity": "Uschgorod"
},
"Vaduz": {
  "exemplarCity": "Vaduz"
},
"Vatican": {
  "exemplarCity": "Vatikan"
},
"Vienna": {
  "exemplarCity": "Wien"
},
"Vilnius": {
  "exemplarCity": "Vilnius"
},
"Volgograd": {
  "exemplarCity": "Wolgograd"
},
"Warsaw": {
  "exemplarCity": "Warschau"
},
"Zagreb": {
  "exemplarCity": "Zagreb"
},
"Zaporozhye": {
  "exemplarCity": "Saporischja"
},
```

```
    "Zurich": {
      "exemplarCity": "Zürich"
    },
    "Africa": {
      "Abidjan": {
        "exemplarCity": "Abidjan"
      },
      "Accra": {
        "exemplarCity": "Accra"
      },
      "Addis_Ababa": {
        "exemplarCity": "Addis Abeba"
      },
      "Algiers": {
        "exemplarCity": "Algier"
      },
      "Asmera": {
        "exemplarCity": "Asmara"
      },
      "Bamako": {
        "exemplarCity": "Bamako"
      },
      "Bangui": {
        "exemplarCity": "Bangui"
      },
      "Banjul": {
        "exemplarCity": "Banjul"
      },
      "Bissau": {
        "exemplarCity": "Bissau"
      },
      "Blantyre": {
        "exemplarCity": "Blantyre"
      },
      "Brazzaville": {
        "exemplarCity": "Brazzaville"
      },
      "Bujumbura": {
        "exemplarCity": "Bujumbura"
      },
      "Cairo": {
        "exemplarCity": "Kairo"
      },
      "Casablanca": {
        "exemplarCity": "Casablanca"
      },
      "Ceuta": {
        "exemplarCity": "Ceuta"
      },
      "Conakry": {
        "exemplarCity": "Conakry"
      },
      "Dakar": {
        "exemplarCity": "Dakar"
      },
      "Dar_es_Salaam": {
```



```
        "exemplarCity": "Daressalam"
    },
    "Djibouti": {
        "exemplarCity": "Dschibuti"
    },
    "Douala": {
        "exemplarCity": "Douala"
    },
    "El_Aaiun": {
        "exemplarCity": "El Aaiún"
    },
    "Freetown": {
        "exemplarCity": "Freetown"
    },
    "Gaborone": {
        "exemplarCity": "Gaborone"
    },
    "Harare": {
        "exemplarCity": "Harare"
    },
    "Johannesburg": {
        "exemplarCity": "Johannesburg"
    },
    "Juba": {
        "exemplarCity": "Juba"
    },
    "Kampala": {
        "exemplarCity": "Kampala"
    },
    "Khartoum": {
        "exemplarCity": "Khartum"
    },
    "Kigali": {
        "exemplarCity": "Kigali"
    },
    "Kinshasa": {
        "exemplarCity": "Kinshasa"
    },
    "Lagos": {
        "exemplarCity": "Lagos"
    },
    "Libreville": {
        "exemplarCity": "Libreville"
    },
    "Lome": {
        "exemplarCity": "Lomé"
    },
    "Luanda": {
        "exemplarCity": "Luanda"
    },
    "Lubumbashi": {
        "exemplarCity": "Lubumbashi"
    },
    "Lusaka": {
        "exemplarCity": "Lusaka"
    },
    "Malabo": {
```

```
        "exemplarCity": "Malabo"
      },
      "Maputo": {
        "exemplarCity": "Maputo"
      },
      "Maseru": {
        "exemplarCity": "Maseru"
      },
      "Mbabane": {
        "exemplarCity": "Mbabane"
      },
      "Mogadishu": {
        "exemplarCity": "Mogadischu"
      },
      "Monrovia": {
        "exemplarCity": "Monrovia"
      },
      "Nairobi": {
        "exemplarCity": "Nairobi"
      },
      "Ndjamena": {
        "exemplarCity": "N'Djamena"
      },
      "Niamey": {
        "exemplarCity": "Niamey"
      },
      "Nouakchott": {
        "exemplarCity": "Nouakchott"
      },
      "Ouagadougou": {
        "exemplarCity": "Ouagadougou"
      },
      "Porto-Novo": {
        "exemplarCity": "Porto Novo"
      },
      "Sao_Tome": {
        "exemplarCity": "São Tomé"
      },
      "Tripoli": {
        "exemplarCity": "Tripolis"
      },
      "Tunis": {
        "exemplarCity": "Tunis"
      },
      "Windhoek": {
        "exemplarCity": "Windhoek"
      }
    },
    "Asia": {
      "Aden": {
        "exemplarCity": "Aden"
      },
      "Almaty": {
        "exemplarCity": "Almaty"
      },
      "Amman": {
        "exemplarCity": "Amman"
      }
    }
  }
}
```

```
    },  
    "Anadyr": {  
      "exemplarCity": "Anadyr"  
    },  
    "Aqtau": {  
      "exemplarCity": "Aqtau"  
    },  
    "Aqtobe": {  
      "exemplarCity": "Aktobe"  
    },  
    "Ashgabat": {  
      "exemplarCity": "Aşgabat"  
    },  
    "Baghdad": {  
      "exemplarCity": "Bagdad"  
    },  
    "Bahrain": {  
      "exemplarCity": "Bahrain"  
    },  
    "Baku": {  
      "exemplarCity": "Baku"  
    },  
    "Bangkok": {  
      "exemplarCity": "Bangkok"  
    },  
    "Barnaul": {  
      "exemplarCity": "Barnaul"  
    },  
    "Beirut": {  
      "exemplarCity": "Beirut"  
    },  
    "Bishkek": {  
      "exemplarCity": "Bischkek"  
    },  
    "Brunei": {  
      "exemplarCity": "Brunei"  
    },  
    "Calcutta": {  
      "exemplarCity": "Kalkutta"  
    },  
    "Chita": {  
      "exemplarCity": "Tschita"  
    },  
    "Choibalsan": {  
      "exemplarCity": "Tschoibalsan"  
    },  
    "Colombo": {  
      "exemplarCity": "Colombo"  
    },  
    "Damascus": {  
      "exemplarCity": "Damaskus"  
    },  
    "Dhaka": {  
      "exemplarCity": "Dhaka"  
    },  
    "Dili": {  
      "exemplarCity": "Dili"
```

```
    },  
    "Dubai": {  
      "exemplarCity": "Dubai"  
    },  
    "Dushanbe": {  
      "exemplarCity": "Duschanbe"  
    },  
    "Gaza": {  
      "exemplarCity": "Gaza"  
    },  
    "Hebron": {  
      "exemplarCity": "Hebron"  
    },  
    "Hong_Kong": {  
      "exemplarCity": "Hongkong"  
    },  
    "Hovd": {  
      "exemplarCity": "Chowd"  
    },  
    "Irkutsk": {  
      "exemplarCity": "Irkutsk"  
    },  
    "Jakarta": {  
      "exemplarCity": "Jakarta"  
    },  
    "Jayapura": {  
      "exemplarCity": "Jayapura"  
    },  
    "Jerusalem": {  
      "exemplarCity": "Jerusalem"  
    },  
    "Kabul": {  
      "exemplarCity": "Kabul"  
    },  
    "Kamchatka": {  
      "exemplarCity": "Kamtschatka"  
    },  
    "Karachi": {  
      "exemplarCity": "Karatschi"  
    },  
    "Katmandu": {  
      "exemplarCity": "Kathmandu"  
    },  
    "Khandyga": {  
      "exemplarCity": "Chandyga"  
    },  
    "Krasnoyarsk": {  
      "exemplarCity": "Krasnojarsk"  
    },  
    "Kuala_Lumpur": {  
      "exemplarCity": "Kuala Lumpur"  
    },  
    "Kuching": {  
      "exemplarCity": "Kuching"  
    },  
    "Kuwait": {  
      "exemplarCity": "Kuwait"
```

```
    },  
    "Macau": {  
      "exemplarCity": "Macao"  
    },  
    "Magadan": {  
      "exemplarCity": "Magadan"  
    },  
    "Makassar": {  
      "exemplarCity": "Makassar"  
    },  
    "Manila": {  
      "exemplarCity": "Manila"  
    },  
    "Muscat": {  
      "exemplarCity": "Maskat"  
    },  
    "Nicosia": {  
      "exemplarCity": "Nikosia"  
    },  
    "Novokuznetsk": {  
      "exemplarCity": "Nowokuznetsk"  
    },  
    "Novosibirsk": {  
      "exemplarCity": "Nowosibirsk"  
    },  
    "Omsk": {  
      "exemplarCity": "Omsk"  
    },  
    "Oral": {  
      "exemplarCity": "Oral"  
    },  
    "Phnom_Penh": {  
      "exemplarCity": "Phnom Penh"  
    },  
    "Pontianak": {  
      "exemplarCity": "Pontianak"  
    },  
    "Pyongyang": {  
      "exemplarCity": "Pjöngjang"  
    },  
    "Qatar": {  
      "exemplarCity": "Katar"  
    },  
    "Qyzylorda": {  
      "exemplarCity": "Qysylorda"  
    },  
    "Rangoon": {  
      "exemplarCity": "Rangun"  
    },  
    "Riyadh": {  
      "exemplarCity": "Riad"  
    },  
    "Saigon": {  
      "exemplarCity": "Ho-Chi-Minh-Stadt"  
    },  
    "Sakhalin": {  
      "exemplarCity": "Sachalin"  
    },
```

```
    },
    "Samarkand": {
      "exemplarCity": "Samarkand"
    },
    "Seoul": {
      "exemplarCity": "Seoul"
    },
    "Shanghai": {
      "exemplarCity": "Shanghai"
    },
    "Singapore": {
      "exemplarCity": "Singapur"
    },
    "Srednekolymsk": {
      "exemplarCity": "Srednekolymsk"
    },
    "Taipei": {
      "exemplarCity": "Taipeh"
    },
    "Tashkent": {
      "exemplarCity": "Taschkent"
    },
    "Tbilisi": {
      "exemplarCity": "Tiflis"
    },
    "Tehran": {
      "exemplarCity": "Teheran"
    },
    "Thimphu": {
      "exemplarCity": "Thimphu"
    },
    "Tokyo": {
      "exemplarCity": "Tokio"
    },
    "Tomsk": {
      "exemplarCity": "Tomsk"
    },
    "Ulaanbaatar": {
      "exemplarCity": "Ulaanbaatar"
    },
    "Urumqi": {
      "exemplarCity": "Ürümqi"
    },
    "Ust-Nera": {
      "exemplarCity": "Ust-Nera"
    },
    "Vientiane": {
      "exemplarCity": "Vientiane"
    },
    "Vladivostok": {
      "exemplarCity": "Wladiwostok"
    },
    "Yakutsk": {
      "exemplarCity": "Jakutsk"
    },
    "Yekaterinburg": {
      "exemplarCity": "Jekaterinburg"
    }
  }
```

```

    },
    "Yerevan": {
      "exemplarCity": "Eriwan"
    }
  },
  "Indian": {
    "Antananarivo": {
      "exemplarCity": "Antananarivo"
    },
    "Chagos": {
      "exemplarCity": "Chagos"
    },
    "Christmas": {
      "exemplarCity": "Weihnachtsinsel"
    },
    "Cocos": {
      "exemplarCity": "Cocos"
    },
    "Comoro": {
      "exemplarCity": "Komoren"
    },
    "Kerguelen": {
      "exemplarCity": "Kerguelen"
    },
    "Mahe": {
      "exemplarCity": "Mahe"
    },
    "Maldives": {
      "exemplarCity": "Malediven"
    },
    "Mauritius": {
      "exemplarCity": "Mauritius"
    },
    "Mayotte": {
      "exemplarCity": "Mayotte"
    },
    "Reunion": {
      "exemplarCity": "Réunion"
    }
  },
  "Australia": {
    "Adelaide": {
      "exemplarCity": "Adelaide"
    },
    "Brisbane": {
      "exemplarCity": "Brisbane"
    },
    "Broken_Hill": {
      "exemplarCity": "Broken Hill"
    },
    "Currie": {
      "exemplarCity": "Currie"
    },
    "Darwin": {
      "exemplarCity": "Darwin"
    },
    "Eucla": {

```

```
        "exemplarCity": "Eucla"
      },
      "Hobart": {
        "exemplarCity": "Hobart"
      },
      "Lindeman": {
        "exemplarCity": "Lindeman"
      },
      "Lord_Howe": {
        "exemplarCity": "Lord Howe"
      },
      "Melbourne": {
        "exemplarCity": "Melbourne"
      },
      "Perth": {
        "exemplarCity": "Perth"
      },
      "Sydney": {
        "exemplarCity": "Sydney"
      }
    },
    "Pacific": {
      "Apia": {
        "exemplarCity": "Apia"
      },
      "Auckland": {
        "exemplarCity": "Auckland"
      },
      "Bougainville": {
        "exemplarCity": "Bougainville"
      },
      "Chatham": {
        "exemplarCity": "Chatham"
      },
      "Easter": {
        "exemplarCity": "Osterinsel"
      },
      "Efate": {
        "exemplarCity": "Efate"
      },
      "Enderbury": {
        "exemplarCity": "Enderbury"
      },
      "Fakaofu": {
        "exemplarCity": "Fakaofu"
      },
      "Fiji": {
        "exemplarCity": "Fidschi"
      },
      "Funafuti": {
        "exemplarCity": "Funafuti"
      },
      "Galapagos": {
        "exemplarCity": "Galapagos"
      },
      "Gambier": {
        "exemplarCity": "Gambier"
      }
    }
  }
}
```



```
    },
    "Guadalcanal": {
      "exemplarCity": "Guadalcanal"
    },
    "Guam": {
      "exemplarCity": "Guam"
    },
    "Honolulu": {
      "exemplarCity": "Honolulu"
    },
    "Johnston": {
      "exemplarCity": "Johnston"
    },
    "Kiritimati": {
      "exemplarCity": "Kiritimati"
    },
    "Kosrae": {
      "exemplarCity": "Kosrae"
    },
    "Kwajalein": {
      "exemplarCity": "Kwajalein"
    },
    "Majuro": {
      "exemplarCity": "Majuro"
    },
    "Marquesas": {
      "exemplarCity": "Marquesas"
    },
    "Midway": {
      "exemplarCity": "Midway"
    },
    "Nauru": {
      "exemplarCity": "Nauru"
    },
    "Niue": {
      "exemplarCity": "Niue"
    },
    "Norfolk": {
      "exemplarCity": "Norfolk"
    },
    "Noumea": {
      "exemplarCity": "Noumea"
    },
    "Pago_Pago": {
      "exemplarCity": "Pago Pago"
    },
    "Palau": {
      "exemplarCity": "Palau"
    },
    "Pitcairn": {
      "exemplarCity": "Pitcairn"
    },
    "Ponape": {
      "exemplarCity": "Pohnpei"
    },
    "Port_Moresby": {
      "exemplarCity": "Port Moresby"
    }
  }
```

```
    },  
    "Rarotonga": {  
      "exemplarCity": "Rarotonga"  
    },  
    "Saipan": {  
      "exemplarCity": "Saipan"  
    },  
    "Tahiti": {  
      "exemplarCity": "Tahiti"  
    },  
    "Tarawa": {  
      "exemplarCity": "Tarawa"  
    },  
    "Tongatapu": {  
      "exemplarCity": "Tongatapu"  
    },  
    "Truk": {  
      "exemplarCity": "Chuuk"  
    },  
    "Wake": {  
      "exemplarCity": "Wake"  
    },  
    "Wallis": {  
      "exemplarCity": "Wallis"  
    }  
  },  
  "Arctic": {  
    "Longyearbyen": {  
      "exemplarCity": "Longyearbyen"  
    }  
  },  
  "Antarctica": {  
    "Casey": {  
      "exemplarCity": "Casey"  
    },  
    "Davis": {  
      "exemplarCity": "Davis"  
    },  
    "DumontDUrville": {  
      "exemplarCity": "Dumont d'Urville"  
    },  
    "Macquarie": {  
      "exemplarCity": "Macquarie"  
    },  
    "Mawson": {  
      "exemplarCity": "Mawson"  
    },  
    "McMurdo": {  
      "exemplarCity": "McMurdo"  
    },  
    "Palmer": {  
      "exemplarCity": "Palmer"  
    },  
    "Rothera": {  
      "exemplarCity": "Rothera"  
    },  
    "Syowa": {
```

```
        "exemplarCity": "Syowa"
      },
      "Troll": {
        "exemplarCity": "Troll"
      },
      "Vostok": {
        "exemplarCity": "Wostok"
      }
    },
    "Etc": {
      "GMT": {
        "exemplarCity": "GMT"
      },
      "GMT1": {
        "exemplarCity": "GMT+1"
      },
      "GMT10": {
        "exemplarCity": "GMT+10"
      },
      "GMT11": {
        "exemplarCity": "GMT+11"
      },
      "GMT12": {
        "exemplarCity": "GMT+12"
      },
      "GMT2": {
        "exemplarCity": "GMT+2"
      },
      "GMT3": {
        "exemplarCity": "GMT+3"
      },
      "GMT4": {
        "exemplarCity": "GMT+4"
      },
      "GMT5": {
        "exemplarCity": "GMT+5"
      },
      "GMT6": {
        "exemplarCity": "GMT+6"
      },
      "GMT7": {
        "exemplarCity": "GMT+7"
      },
      "GMT8": {
        "exemplarCity": "GMT+8"
      },
      "GMT9": {
        "exemplarCity": "GMT+9"
      },
      "GMT-1": {
        "exemplarCity": "GMT-1"
      },
      "GMT-10": {
        "exemplarCity": "GMT-10"
      },
      "GMT-11": {
        "exemplarCity": "GMT-11"
      }
    }
  }
}
```

```

    },
    "GMT-12": {
      "exemplarCity": "GMT-12"
    },
    "GMT-13": {
      "exemplarCity": "GMT-13"
    },
    "GMT-14": {
      "exemplarCity": "GMT-14"
    },
    "GMT-2": {
      "exemplarCity": "GMT-2"
    },
    "GMT-3": {
      "exemplarCity": "GMT-3"
    },
    "GMT-4": {
      "exemplarCity": "GMT-4"
    },
    "GMT-5": {
      "exemplarCity": "GMT-5"
    },
    "GMT-6": {
      "exemplarCity": "GMT-6"
    },
    "GMT-7": {
      "exemplarCity": "GMT-7"
    },
    "GMT-8": {
      "exemplarCity": "GMT-8"
    },
    "GMT-9": {
      "exemplarCity": "GMT-9"
    },
    "Unknown": {
      "exemplarCity": "Unbekannt"
    }
  },
  "metazone": {
    "Acre": {
      "long": {
        "generic": "Acre-Zeit",
        "standard": "Acre-Normalzeit",
        "daylight": "Acre-Sommerzeit"
      }
    },
    "Afghanistan": {
      "long": {
        "standard": "Afghanistan-Zeit"
      }
    },
    "Africa_Central": {
      "long": {
        "standard": "Zentralafrikanische Zeit"
      }
    }
  },

```

```
"Africa_Eastern": {
  "long": {
    "standard": "Ostafrikanische Zeit"
  }
},
"Africa_Southern": {
  "long": {
    "standard": "Südafrikanische Zeit"
  }
},
"Africa_Western": {
  "long": {
    "generic": "Westafrikanische Zeit",
    "standard": "Westafrikanische Normalzeit",
    "daylight": "Westafrikanische Sommerzeit"
  }
},
"Alaska": {
  "long": {
    "generic": "Alaska-Zeit",
    "standard": "Alaska-Normalzeit",
    "daylight": "Alaska-Sommerzeit"
  }
},
"Almaty": {
  "long": {
    "generic": "Almaty-Zeit",
    "standard": "Almaty-Normalzeit",
    "daylight": "Almaty-Sommerzeit"
  }
},
"Amazon": {
  "long": {
    "generic": "Amazonas-Zeit",
    "standard": "Amazonas-Normalzeit",
    "daylight": "Amazonas-Sommerzeit"
  }
},
"America_Central": {
  "long": {
    "generic": "Nordamerikanische Inlandzeit",
    "standard": "Nordamerikanische Inland-Normalzeit",
    "daylight": "Nordamerikanische Inland-Sommerzeit"
  }
},
"America_Eastern": {
  "long": {
    "generic": "Nordamerikanische Ostküstenzeit",
    "standard": "Nordamerikanische Ostküsten-Normalzeit",
    "daylight": "Nordamerikanische Ostküsten-Sommerzeit"
  }
},
"America_Mountain": {
  "long": {
    "generic": "Rocky-Mountain-Zeit",
    "standard": "Rocky Mountain-Normalzeit",
    "daylight": "Rocky-Mountain-Sommerzeit"
  }
}
```

```

    }
  },
  "America_Pacific": {
    "long": {
      "generic": "Nordamerikanische Westküstenzeit",
      "standard": "Nordamerikanische Westküsten-Normalzeit",
      "daylight": "Nordamerikanische Westküsten-Sommerzeit"
    }
  },
  "Anadyr": {
    "long": {
      "generic": "Anadyr Zeit",
      "standard": "Anadyr Normalzeit",
      "daylight": "Anadyr Sommerzeit"
    }
  },
  "Apia": {
    "long": {
      "generic": "Apia-Zeit",
      "standard": "Apia-Normalzeit",
      "daylight": "Apia-Sommerzeit"
    }
  },
  "Aqtau": {
    "long": {
      "generic": "Aqtau-Zeit",
      "standard": "Aqtau-Normalzeit",
      "daylight": "Aqtau-Sommerzeit"
    }
  },
  "Aqtobe": {
    "long": {
      "generic": "Aqtöbe-Zeit",
      "standard": "Aqtöbe-Normalzeit",
      "daylight": "Aqtöbe-Sommerzeit"
    }
  },
  "Arabian": {
    "long": {
      "generic": "Arabische Zeit",
      "standard": "Arabische Normalzeit",
      "daylight": "Arabische Sommerzeit"
    }
  },
  "Argentina": {
    "long": {
      "generic": "Argentinische Zeit",
      "standard": "Argentinische Normalzeit",
      "daylight": "Argentinische Sommerzeit"
    }
  },
  "Argentina_Western": {
    "long": {
      "generic": "Westargentinische Zeit",
      "standard": "Westargentinische Normalzeit",
      "daylight": "Westargentinische Sommerzeit"
    }
  }
}

```

```

    },
    "Armenia": {
      "long": {
        "generic": "Armenische Zeit",
        "standard": "Armenische Normalzeit",
        "daylight": "Armenische Sommerzeit"
      }
    },
    "Atlantic": {
      "long": {
        "generic": "Atlantik-Zeit",
        "standard": "Atlantik-Normalzeit",
        "daylight": "Atlantik-Sommerzeit"
      }
    },
    "Australia_Central": {
      "long": {
        "generic": "Zentralaustralische Zeit",
        "standard": "Zentralaustralische Normalzeit",
        "daylight": "Zentralaustralische Sommerzeit"
      }
    },
    "Australia_CentralWestern": {
      "long": {
        "generic": "Zentral-/Westaustralische Zeit",
        "standard": "Zentral-/Westaustralische Normalzeit",
        "daylight": "Zentral-/Westaustralische Sommerzeit"
      }
    },
    "Australia_Eastern": {
      "long": {
        "generic": "Ostaustralische Zeit",
        "standard": "Ostaustralische Normalzeit",
        "daylight": "Ostaustralische Sommerzeit"
      }
    },
    "Australia_Western": {
      "long": {
        "generic": "Westaustralische Zeit",
        "standard": "Westaustralische Normalzeit",
        "daylight": "Westaustralische Sommerzeit"
      }
    },
    "Azerbaijan": {
      "long": {
        "generic": "Aserbaidsschanische Zeit",
        "standard": "Aserbeidschanische Normalzeit",
        "daylight": "Aserbaidsschanische Sommerzeit"
      }
    },
    "Azores": {
      "long": {
        "generic": "Azoren-Zeit",
        "standard": "Azoren-Normalzeit",
        "daylight": "Azoren-Sommerzeit"
      }
    }
  },

```

```
"Bangladesh": {
  "long": {
    "generic": "Bangladesch-Zeit",
    "standard": "Bangladesch-Normalzeit",
    "daylight": "Bangladesch-Sommerzeit"
  }
},
"Bhutan": {
  "long": {
    "standard": "Bhutan-Zeit"
  }
},
"Bolivia": {
  "long": {
    "standard": "Bolivianische Zeit"
  }
},
"Brasilia": {
  "long": {
    "generic": "Brasília-Zeit",
    "standard": "Brasília-Normalzeit",
    "daylight": "Brasília-Sommerzeit"
  }
},
"Brunei": {
  "long": {
    "standard": "Brunei-Zeit"
  }
},
"Cape_Verde": {
  "long": {
    "generic": "Cabo-Verde-Zeit",
    "standard": "Cabo-Verde-Normalzeit",
    "daylight": "Cabo-Verde-Sommerzeit"
  }
},
"Casey": {
  "long": {
    "standard": "Casey-Zeit"
  }
},
"Chamorro": {
  "long": {
    "standard": "Chamorro-Zeit"
  }
},
"Chatham": {
  "long": {
    "generic": "Chatham-Zeit",
    "standard": "Chatham-Normalzeit",
    "daylight": "Chatham-Sommerzeit"
  }
},
"Chile": {
  "long": {
    "generic": "Chilenische Zeit",
    "standard": "Chilenische Normalzeit",
```



```

        "daylight": "Chilenische Sommerzeit"
    },
    },
    "China": {
        "long": {
            "generic": "Chinesische Zeit",
            "standard": "Chinesische Normalzeit",
            "daylight": "Chinesische Sommerzeit"
        }
    },
    "Choibalsan": {
        "long": {
            "generic": "Tschoibalsan-Zeit",
            "standard": "Tschoibalsan-Normalzeit",
            "daylight": "Tschoibalsan-Sommerzeit"
        }
    },
    "Christmas": {
        "long": {
            "standard": "Weihnachtsinsel-Zeit"
        }
    },
    "Cocos": {
        "long": {
            "standard": "Kokosinseln-Zeit"
        }
    },
    "Colombia": {
        "long": {
            "generic": "Kolumbianische Zeit",
            "standard": "Kolumbianische Normalzeit",
            "daylight": "Kolumbianische Sommerzeit"
        }
    },
    "Cook": {
        "long": {
            "generic": "Cookinseln-Zeit",
            "standard": "Cookinseln-Normalzeit",
            "daylight": "Cookinseln-Sommerzeit"
        }
    },
    "Cuba": {
        "long": {
            "generic": "Kubanische Zeit",
            "standard": "Kubanische Normalzeit",
            "daylight": "Kubanische Sommerzeit"
        }
    },
    "Davis": {
        "long": {
            "standard": "Davis-Zeit"
        }
    },
    "DumontDUrville": {
        "long": {
            "standard": "Dumont-d'Urville-Zeit"
        }
    }
}

```

```

    },
    "East_Timor": {
      "long": {
        "standard": "Osttimor-Zeit"
      }
    },
    "Easter": {
      "long": {
        "generic": "Osterinsel-Zeit",
        "standard": "Osterinsel-Normalzeit",
        "daylight": "Osterinsel-Sommerzeit"
      }
    },
    "Ecuador": {
      "long": {
        "standard": "Ecuadorianische Zeit"
      }
    },
    "Europe_Central": {
      "long": {
        "generic": "Mitteleuropäische Zeit",
        "standard": "Mitteleuropäische Normalzeit",
        "daylight": "Mitteleuropäische Sommerzeit"
      },
      "short": {
        "generic": "MEZ",
        "standard": "MEZ",
        "daylight": "MESZ"
      }
    },
    "Europe_Eastern": {
      "long": {
        "generic": "Osteuropäische Zeit",
        "standard": "Osteuropäische Normalzeit",
        "daylight": "Osteuropäische Sommerzeit"
      },
      "short": {
        "generic": "OEZ",
        "standard": "OEZ",
        "daylight": "OESZ"
      }
    },
    "Europe_Further_Eastern": {
      "long": {
        "standard": "Kaliningrader Zeit"
      }
    },
    "Europe_Western": {
      "long": {
        "generic": "Westeuropäische Zeit",
        "standard": "Westeuropäische Normalzeit",
        "daylight": "Westeuropäische Sommerzeit"
      },
      "short": {
        "generic": "WEZ",
        "standard": "WEZ",
        "daylight": "WESZ"
      }
    }
  }

```

```

    }
  },
  "Falkland": {
    "long": {
      "generic": "Falklandinseln-Zeit",
      "standard": "Falklandinseln-Normalzeit",
      "daylight": "Falklandinseln-Sommerzeit"
    }
  },
  "Fiji": {
    "long": {
      "generic": "Fidschi-Zeit",
      "standard": "Fidschi-Normalzeit",
      "daylight": "Fidschi-Sommerzeit"
    }
  },
  "French_Guiana": {
    "long": {
      "standard": "Französisch-Guayana-Zeit"
    }
  },
  "French_Southern": {
    "long": {
      "standard": "Französische Süd- und Antarktisgebiete-Zeit"
    }
  },
  "Galapagos": {
    "long": {
      "standard": "Galapagos-Zeit"
    }
  },
  "Gambier": {
    "long": {
      "standard": "Gambier-Zeit"
    }
  },
  "Georgia": {
    "long": {
      "generic": "Georgische Zeit",
      "standard": "Georgische Normalzeit",
      "daylight": "Georgische Sommerzeit"
    }
  },
  "Gilbert_Islands": {
    "long": {
      "standard": "Gilbert-Inseln-Zeit"
    }
  },
  "GMT": {
    "long": {
      "standard": "Mittlere Greenwich-Zeit"
    }
  },
  "Greenland_Eastern": {
    "long": {
      "generic": "Ostgrönland-Zeit",
      "standard": "Ostgrönland-Normalzeit",

```

```

        "daylight": "Ostgrönland-Sommerzeit"
    },
    },
    "Greenland_Western": {
        "long": {
            "generic": "Westgrönland-Zeit",
            "standard": "Westgrönland-Normalzeit",
            "daylight": "Westgrönland-Sommerzeit"
        }
    },
    "Guam": {
        "long": {
            "standard": "Guam-Zeit"
        }
    },
    "Gulf": {
        "long": {
            "standard": "Golf-Zeit"
        }
    },
    "Guyana": {
        "long": {
            "standard": "Guyana-Zeit"
        }
    },
    "Hawaii_Aleutian": {
        "long": {
            "generic": "Hawaii-Aleuten-Zeit",
            "standard": "Hawaii-Aleuten-Normalzeit",
            "daylight": "Hawaii-Aleuten-Sommerzeit"
        }
    },
    "Hong_Kong": {
        "long": {
            "generic": "Hongkong-Zeit",
            "standard": "Hongkong-Normalzeit",
            "daylight": "Hongkong-Sommerzeit"
        }
    },
    "Hovd": {
        "long": {
            "generic": "Chowd-Zeit",
            "standard": "Chowd-Normalzeit",
            "daylight": "Chowd-Sommerzeit"
        }
    },
    "India": {
        "long": {
            "standard": "Indische Zeit"
        }
    },
    "Indian_Ocean": {
        "long": {
            "standard": "Indischer Ozean-Zeit"
        }
    },
    "Indochina": {

```

```
        "long": {
          "standard": "Indochina-Zeit"
        }
      },
      "Indonesia_Central": {
        "long": {
          "standard": "Zentralindonesische Zeit"
        }
      },
      "Indonesia_Eastern": {
        "long": {
          "standard": "Ostindonesische Zeit"
        }
      },
      "Indonesia_Western": {
        "long": {
          "standard": "Westindonesische Zeit"
        }
      },
      "Iran": {
        "long": {
          "generic": "Iranische Zeit",
          "standard": "Iranische Normalzeit",
          "daylight": "Iranische Sommerzeit"
        }
      },
      "Irkutsk": {
        "long": {
          "generic": "Irkutsk-Zeit",
          "standard": "Irkutsk-Normalzeit",
          "daylight": "Irkutsk-Sommerzeit"
        }
      },
      "Israel": {
        "long": {
          "generic": "Israelische Zeit",
          "standard": "Israelische Normalzeit",
          "daylight": "Israelische Sommerzeit"
        }
      },
      "Japan": {
        "long": {
          "generic": "Japanische Zeit",
          "standard": "Japanische Normalzeit",
          "daylight": "Japanische Sommerzeit"
        }
      },
      "Kamchatka": {
        "long": {
          "generic": "Kamtschatka-Zeit",
          "standard": "Kamtschatka-Normalzeit",
          "daylight": "Kamtschatka-Sommerzeit"
        }
      },
      "Kazakhstan_Eastern": {
        "long": {
          "standard": "Ostkasachische Zeit"
```

```

    }
  },
  "Kazakhstan_Western": {
    "long": {
      "standard": "Westkasachische Zeit"
    }
  },
  "Korea": {
    "long": {
      "generic": "Koreanische Zeit",
      "standard": "Koreanische Normalzeit",
      "daylight": "Koreanische Sommerzeit"
    }
  },
  "Kosrae": {
    "long": {
      "standard": "Kosrae-Zeit"
    }
  },
  "Krasnoyarsk": {
    "long": {
      "generic": "Krasnojarsk-Zeit",
      "standard": "Krasnojarsk-Normalzeit",
      "daylight": "Krasnojarsk-Sommerzeit"
    }
  },
  "Kyrgystan": {
    "long": {
      "standard": "Kirgisistan-Zeit"
    }
  },
  "Lanka": {
    "long": {
      "standard": "Sri-Lanka-Zeit"
    }
  },
  "Line_Islands": {
    "long": {
      "standard": "Linieninseln-Zeit"
    }
  },
  "Lord_Howe": {
    "long": {
      "generic": "Lord-Howe-Zeit",
      "standard": "Lord-Howe-Normalzeit",
      "daylight": "Lord-Howe-Sommerzeit"
    }
  },
  "Macau": {
    "long": {
      "generic": "Macau-Zeit",
      "standard": "Macau-Normalzeit",
      "daylight": "Macau-Sommerzeit"
    }
  },
  "Macquarie": {
    "long": {

```

```

        "standard": "Macquarieinsel-Zeit"
    },
    },
    "Magadan": {
        "long": {
            "generic": "Magadan-Zeit",
            "standard": "Magadan-Normalzeit",
            "daylight": "Magadan-Sommerzeit"
        }
    },
    "Malaysia": {
        "long": {
            "standard": "Malaysische Zeit"
        }
    },
    "Maldives": {
        "long": {
            "standard": "Malediven-Zeit"
        }
    },
    "Marquesas": {
        "long": {
            "standard": "Marquesas-Zeit"
        }
    },
    "Marshall_Islands": {
        "long": {
            "standard": "Marshallinseln-Zeit"
        }
    },
    "Mauritius": {
        "long": {
            "generic": "Mauritius-Zeit",
            "standard": "Mauritius-Normalzeit",
            "daylight": "Mauritius-Sommerzeit"
        }
    },
    "Mawson": {
        "long": {
            "standard": "Mawson-Zeit"
        }
    },
    "Mexico_Northwest": {
        "long": {
            "generic": "Mexiko Nordwestliche Zone-Zeit",
            "standard": "Mexiko Nordwestliche Zone-Normalzeit",
            "daylight": "Mexiko Nordwestliche Zone-Sommerzeit"
        }
    },
    "Mexico_Pacific": {
        "long": {
            "generic": "Mexiko Pazifikzone-Zeit",
            "standard": "Mexiko Pazifikzone-Normalzeit",
            "daylight": "Mexiko Pazifikzone-Sommerzeit"
        }
    },
    "Mongolia": {

```

```
        "long": {
          "generic": "Ulaanbaatar-Zeit",
          "standard": "Ulaanbaatar-Normalzeit",
          "daylight": "Ulaanbaatar-Sommerzeit"
        }
      },
      "Moscow": {
        "long": {
          "generic": "Moskauer Zeit",
          "standard": "Moskauer Normalzeit",
          "daylight": "Moskauer Sommerzeit"
        }
      },
      "Myanmar": {
        "long": {
          "standard": "Myanmar-Zeit"
        }
      },
      "Nauru": {
        "long": {
          "standard": "Nauru-Zeit"
        }
      },
      "Nepal": {
        "long": {
          "standard": "Nepalesische Zeit"
        }
      },
      "New_Caledonia": {
        "long": {
          "generic": "Neukaledonische Zeit",
          "standard": "Neukaledonische Normalzeit",
          "daylight": "Neukaledonische Sommerzeit"
        }
      },
      "New_Zealand": {
        "long": {
          "generic": "Neuseeland-Zeit",
          "standard": "Neuseeland-Normalzeit",
          "daylight": "Neuseeland-Sommerzeit"
        }
      },
      "Newfoundland": {
        "long": {
          "generic": "Neufundland-Zeit",
          "standard": "Neufundland-Normalzeit",
          "daylight": "Neufundland-Sommerzeit"
        }
      },
      "Niue": {
        "long": {
          "standard": "Niue-Zeit"
        }
      },
      "Norfolk": {
        "long": {
          "standard": "Norfolkinsel-Zeit"
```



```

    }
  },
  "Noronha": {
    "long": {
      "generic": "Fernando de Noronha-Zeit",
      "standard": "Fernando de Noronha-Normalzeit",
      "daylight": "Fernando de Noronha-Sommerzeit"
    }
  },
  "North_Mariana": {
    "long": {
      "standard": "Nördliche-Marianen-Zeit"
    }
  },
  "Novosibirsk": {
    "long": {
      "generic": "Nowosibirsk-Zeit",
      "standard": "Nowosibirsk-Normalzeit",
      "daylight": "Nowosibirsk-Sommerzeit"
    }
  },
  "Omsk": {
    "long": {
      "generic": "Omsk-Zeit",
      "standard": "Omsk-Normalzeit",
      "daylight": "Omsk-Sommerzeit"
    }
  },
  "Pakistan": {
    "long": {
      "generic": "Pakistanische Zeit",
      "standard": "Pakistanische Normalzeit",
      "daylight": "Pakistanische Sommerzeit"
    }
  },
  "Palau": {
    "long": {
      "standard": "Palau-Zeit"
    }
  },
  "Papua_New_Guinea": {
    "long": {
      "standard": "Papua-Neuguinea-Zeit"
    }
  },
  "Paraguay": {
    "long": {
      "generic": "Paraguayische Zeit",
      "standard": "Paraguayische Normalzeit",
      "daylight": "Paraguayische Sommerzeit"
    }
  },
  "Peru": {
    "long": {
      "generic": "Peruanische Zeit",
      "standard": "Peruanische Normalzeit",
      "daylight": "Peruanische Sommerzeit"
    }
  }
}

```

```
    },
    "Philippines": {
      "long": {
        "generic": "Philippinische Zeit",
        "standard": "Philippinische Normalzeit",
        "daylight": "Philippinische Sommerzeit"
      }
    },
    "Phoenix_Islands": {
      "long": {
        "standard": "Phoenixinseln-Zeit"
      }
    },
    "Pierre_Miquelon": {
      "long": {
        "generic": "Saint-Pierre-und-Miquelon-Zeit",
        "standard": "Saint-Pierre-und-Miquelon-Normalzeit",
        "daylight": "Saint-Pierre-und-Miquelon-Sommerzeit"
      }
    },
    "Pitcairn": {
      "long": {
        "standard": "Pitcairninseln-Zeit"
      }
    },
    "Ponape": {
      "long": {
        "standard": "Ponape-Zeit"
      }
    },
    "Pyongyang": {
      "long": {
        "standard": "Pjöngjang-Zeit"
      }
    },
    "Qyzylorda": {
      "long": {
        "generic": "Quysylorda-Zeit",
        "standard": "Quysylorda-Normalzeit",
        "daylight": "Qysylorda-Sommerzeit"
      }
    },
    "Reunion": {
      "long": {
        "standard": "Réunion-Zeit"
      }
    },
    "Rothera": {
      "long": {
        "standard": "Rothera-Zeit"
      }
    },
    "Sakhalin": {
      "long": {
        "generic": "Sachalin-Zeit",
        "standard": "Sachalin-Normalzeit",
```

```

        "daylight": "Sachalin-Sommerzeit"
    },
    },
    "Samara": {
        "long": {
            "generic": "Samara-Zeit",
            "standard": "Samara-Normalzeit",
            "daylight": "Samara-Sommerzeit"
        }
    },
    "Samoa": {
        "long": {
            "generic": "Samoa-Zeit",
            "standard": "Samoa-Normalzeit",
            "daylight": "Samoa-Sommerzeit"
        }
    },
    "Seychelles": {
        "long": {
            "standard": "Seychellen-Zeit"
        }
    },
    "Singapore": {
        "long": {
            "standard": "Singapur-Zeit"
        }
    },
    "Solomon": {
        "long": {
            "standard": "Salomoninseln-Zeit"
        }
    },
    "South_Georgia": {
        "long": {
            "standard": "Südgeorgische Zeit"
        }
    },
    "Suriname": {
        "long": {
            "standard": "Suriname-Zeit"
        }
    },
    "Syowa": {
        "long": {
            "standard": "Syowa-Zeit"
        }
    },
    "Tahiti": {
        "long": {
            "standard": "Tahiti-Zeit"
        }
    },
    "Taipei": {
        "long": {
            "generic": "Taipeh-Zeit",
            "standard": "Taipeh-Normalzeit",
            "daylight": "Taipeh-Sommerzeit"
        }
    }
}

```

```
    },
    "Tajikistan": {
      "long": {
        "standard": "Tadschikistan-Zeit"
      }
    },
    "Tokelau": {
      "long": {
        "standard": "Tokelau-Zeit"
      }
    },
    "Tonga": {
      "long": {
        "generic": "Tonganische Zeit",
        "standard": "Tonganische Normalzeit",
        "daylight": "Tonganische Sommerzeit"
      }
    },
    "Truk": {
      "long": {
        "standard": "Chuuk-Zeit"
      }
    },
    "Turkmenistan": {
      "long": {
        "generic": "Turkmenistan-Zeit",
        "standard": "Turkmenistan-Normalzeit",
        "daylight": "Turkmenistan-Sommerzeit"
      }
    },
    "Tuvalu": {
      "long": {
        "standard": "Tuvalu-Zeit"
      }
    },
    "Uruguay": {
      "long": {
        "generic": "Uruguayanische Zeit",
        "standard": "Uruguyanische Normalzeit",
        "daylight": "Uruguayanische Sommerzeit"
      }
    },
    "Uzbekistan": {
      "long": {
        "generic": "Usbekistan-Zeit",
        "standard": "Usbekistan-Normalzeit",
        "daylight": "Usbekistan-Sommerzeit"
      }
    },
    "Vanuatu": {
      "long": {
        "generic": "Vanuatu-Zeit",
        "standard": "Vanuatu-Normalzeit",
        "daylight": "Vanuatu-Sommerzeit"
      }
    }
  },
}
```

```

    "Venezuela": {
      "long": {
        "standard": "Venezuela-Zeit"
      }
    },
    "Vladivostok": {
      "long": {
        "generic": "Wladiwostok-Zeit",
        "standard": "Wladiwostok-Normalzeit",
        "daylight": "Wladiwostok-Sommerzeit"
      }
    },
    "Volgograd": {
      "long": {
        "generic": "Wolgograd-Zeit",
        "standard": "Wolgograd-Normalzeit",
        "daylight": "Wolgograd-Sommerzeit"
      }
    },
    "Vostok": {
      "long": {
        "standard": "Wostok-Zeit"
      }
    },
    "Wake": {
      "long": {
        "standard": "Wake-Insel-Zeit"
      }
    },
    "Wallis": {
      "long": {
        "standard": "Wallis-und-Futuna-Zeit"
      }
    },
    "Yakutsk": {
      "long": {
        "generic": "Jakutsk-Zeit",
        "standard": "Jakutsk-Normalzeit",
        "daylight": "Jakutsk-Sommerzeit"
      }
    },
    "Yekaterinburg": {
      "long": {
        "generic": "Jekaterinburg-Zeit",
        "standard": "Jekaterinburg-Normalzeit",
        "daylight": "Jekaterinburg-Sommerzeit"
      }
    }
  }
}

```

TIMEZONENAMES.JSX

```

{
  "main";
  {
    "de";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 12879 $",
          "_cldrVersion";
          "30.0.3";
        }
        "language";
        "de";
      }
    }
    "dates";
    {
      "timeZoneNames";
      {
        "hourFormat";
        "+HH:mm;-HH:mm",
        "gmtFormat";
        "GMT{0}",
        "gmtZeroFormat";
        "GMT",
        "regionFormat";
        "{0} Zeit",
        "regionFormat-type-daylight";
        "{0} Sommerzeit",
        "regionFormat-type-standard";
        "{0} Normalzeit",
        "fallbackFormat";
        "{1} ({0})",
        "zone";
        {
          "America";
          {
            "Adak";
            {
              "exemplarCity";
              "Adak";
            }
            "Anchorage";
            {
              "exemplarCity";
              "Anchorage";
            }
            "Anguilla";
            {
              "exemplarCity";
              "Anguilla";
            }
            "Antigua";
            {
              "exemplarCity";
            }
          }
        }
      }
    }
  }
}

```

```
        "Antigua";
    }
    "Araguaina";
    {
        "exemplarCity";
        "Araguaina";
    }
    "Argentina";
    {
        "Rio_Gallegos";
        {
            "exemplarCity";
            "Rio Gallegos";
        }
        "San_Juan";
        {
            "exemplarCity";
            "San Juan";
        }
        "Ushuaia";
        {
            "exemplarCity";
            "Ushuaia";
        }
        "La_Rioja";
        {
            "exemplarCity";
            "La Rioja";
        }
        "San_Luis";
        {
            "exemplarCity";
            "San Luis";
        }
        "Salta";
        {
            "exemplarCity";
            "Salta";
        }
        "Tucuman";
        {
            "exemplarCity";
            "Tucuman";
        }
    }
    "Aruba";
    {
        "exemplarCity";
        "Aruba";
    }
    "Asuncion";
    {
        "exemplarCity";
        "Asunción";
    }
    "Bahia";
    {
```

```
        "exemplarCity";
        "Bahia";
    }
    "Bahia_Banderas";
    {
        "exemplarCity";
        "Bahia Banderas";
    }
    "Barbados";
    {
        "exemplarCity";
        "Barbados";
    }
    "Belem";
    {
        "exemplarCity";
        "Belem";
    }
    "Belize";
    {
        "exemplarCity";
        "Belize";
    }
    "Blanc-Sablon";
    {
        "exemplarCity";
        "Blanc-Sablon";
    }
    "Boa_Vista";
    {
        "exemplarCity";
        "Boa Vista";
    }
    "Bogota";
    {
        "exemplarCity";
        "Bogotá";
    }
    "Boise";
    {
        "exemplarCity";
        "Boise";
    }
    "Buenos_Aires";
    {
        "exemplarCity";
        "Buenos Aires";
    }
    "Cambridge_Bay";
    {
        "exemplarCity";
        "Cambridge Bay";
    }
    "Campo_Grande";
    {
        "exemplarCity";
        "Campo Grande";
    }
```



```
}
"Cancun";
{
  "exemplarCity";
  "Cancún";
}
"Caracas";
{
  "exemplarCity";
  "Caracas";
}
"Catamarca";
{
  "exemplarCity";
  "Catamarca";
}
"Cayenne";
{
  "exemplarCity";
  "Cayenne";
}
"Cayman";
{
  "exemplarCity";
  "Kaimaninseln";
}
"Chicago";
{
  "exemplarCity";
  "Chicago";
}
"Chihuahua";
{
  "exemplarCity";
  "Chihuahua";
}
"Coral_Harbour";
{
  "exemplarCity";
  "Atikokan";
}
"Cordoba";
{
  "exemplarCity";
  "Córdoba";
}
"Costa_Rica";
{
  "exemplarCity";
  "Costa Rica";
}
"Creston";
{
  "exemplarCity";
  "Creston";
}
"Cuiaba";
```

```
{
    "exemplarCity";
    "Cuiaba";
}
"Curacao";
{
    "exemplarCity";
    "Curaçao";
}
"Danmarkshavn";
{
    "exemplarCity";
    "Danmarkshavn";
}
"Dawson";
{
    "exemplarCity";
    "Dawson";
}
"Dawson_Creek";
{
    "exemplarCity";
    "Dawson Creek";
}
"Denver";
{
    "exemplarCity";
    "Denver";
}
"Detroit";
{
    "exemplarCity";
    "Detroit";
}
"Dominica";
{
    "exemplarCity";
    "Dominica";
}
"Edmonton";
{
    "exemplarCity";
    "Edmonton";
}
"Eirunepe";
{
    "exemplarCity";
    "Eirunepe";
}
"El_Salvador";
{
    "exemplarCity";
    "El Salvador";
}
"Fort_Nelson";
{
    "exemplarCity";
```

```
        "Fort Nelson";
    }
    "Fortaleza";
    {
        "exemplarCity";
        "Fortaleza";
    }
    "Glace_Bay";
    {
        "exemplarCity";
        "Glace Bay";
    }
    "Godthab";
    {
        "exemplarCity";
        "Nuuk";
    }
    "Goose_Bay";
    {
        "exemplarCity";
        "Goose Bay";
    }
    "Grand_Turk";
    {
        "exemplarCity";
        "Grand Turk";
    }
    "Grenada";
    {
        "exemplarCity";
        "Grenada";
    }
    "Guadeloupe";
    {
        "exemplarCity";
        "Guadeloupe";
    }
    "Guatemala";
    {
        "exemplarCity";
        "Guatemala";
    }
    "Guayaquil";
    {
        "exemplarCity";
        "Guayaquil";
    }
    "Guyana";
    {
        "exemplarCity";
        "Guyana";
    }
    "Halifax";
    {
        "exemplarCity";
        "Halifax";
    }
}
```

```
"Havana";
{
  "exemplarCity";
  "Havanna";
}
"Hermosillo";
{
  "exemplarCity";
  "Hermosillo";
}
"Indiana";
{
  "Vincennes";
  {
    "exemplarCity";
    "Vincennes, Indiana";
  }
  "Petersburg";
  {
    "exemplarCity";
    "Petersburg, Indiana";
  }
  "Tell_City";
  {
    "exemplarCity";
    "Tell City, Indiana";
  }
  "Knox";
  {
    "exemplarCity";
    "Knox, Indiana";
  }
  "Winamac";
  {
    "exemplarCity";
    "Winamac, Indiana";
  }
  "Marengo";
  {
    "exemplarCity";
    "Marengo, Indiana";
  }
  "Vevay";
  {
    "exemplarCity";
    "Vevay, Indiana";
  }
}
"Indianapolis";
{
  "exemplarCity";
  "Indianapolis";
}
"Inuvik";
{
  "exemplarCity";
  "Inuvik";
}
```

```
}
"Iqaluit";
{
  "exemplarCity";
  "Iqaluit";
}
"Jamaica";
{
  "exemplarCity";
  "Jamaika";
}
"Jujuy";
{
  "exemplarCity";
  "Jujuy";
}
"Juneau";
{
  "exemplarCity";
  "Juneau";
}
"Kentucky";
{
  "Monticello";
  {
    "exemplarCity";
    "Monticello, Kentucky";
  }
}
"Kralendijk";
{
  "exemplarCity";
  "Kralendijk";
}
"La_Paz";
{
  "exemplarCity";
  "La Paz";
}
"Lima";
{
  "exemplarCity";
  "Lima";
}
"Los_Angeles";
{
  "exemplarCity";
  "Los Angeles";
}
"Louisville";
{
  "exemplarCity";
  "Louisville";
}
"Lower_Princes";
{
  "exemplarCity";
```

```
        "Lower Prince's Quarter";
    }
    "Maceio";
    {
        "exemplarCity";
        "Maceio";
    }
    "Managua";
    {
        "exemplarCity";
        "Managua";
    }
    "Manaus";
    {
        "exemplarCity";
        "Manaus";
    }
    "Marigot";
    {
        "exemplarCity";
        "Marigot";
    }
    "Martinique";
    {
        "exemplarCity";
        "Martinique";
    }
    "Matamoros";
    {
        "exemplarCity";
        "Matamoros";
    }
    "Mazatlan";
    {
        "exemplarCity";
        "Mazatlan";
    }
    "Mendoza";
    {
        "exemplarCity";
        "Mendoza";
    }
    "Menominee";
    {
        "exemplarCity";
        "Menominee";
    }
    "Merida";
    {
        "exemplarCity";
        "Merida";
    }
    "Metlakatla";
    {
        "exemplarCity";
        "Metlakatla";
    }
}
```

```
"Mexico_City";
{
  "exemplarCity";
  "Mexiko-Stadt";
}
"Miquelon";
{
  "exemplarCity";
  "Miquelon";
}
"Moncton";
{
  "exemplarCity";
  "Moncton";
}
"Monterrey";
{
  "exemplarCity";
  "Monterrey";
}
"Montevideo";
{
  "exemplarCity";
  "Montevideo";
}
"Montserrat";
{
  "exemplarCity";
  "Montserrat";
}
"Nassau";
{
  "exemplarCity";
  "Nassau";
}
"New_York";
{
  "exemplarCity";
  "New York";
}
"Nipigon";
{
  "exemplarCity";
  "Nipigon";
}
"Nome";
{
  "exemplarCity";
  "Nome";
}
"Noronha";
{
  "exemplarCity";
  "Noronha";
}
"North_Dakota";
{
```

```
        "Beulah";
        {
            "exemplarCity";
            "Beulah, North Dakota";
        }
        "New_Salem";
        {
            "exemplarCity";
            "New Salem, North Dakota";
        }
        "Center";
        {
            "exemplarCity";
            "Center, North Dakota";
        }
    }
    "Ojinaga";
    {
        "exemplarCity";
        "Ojinaga";
    }
    "Panama";
    {
        "exemplarCity";
        "Panama";
    }
    "Pangnirtung";
    {
        "exemplarCity";
        "Pangnirtung";
    }
    "Paramaribo";
    {
        "exemplarCity";
        "Paramaribo";
    }
    "Phoenix";
    {
        "exemplarCity";
        "Phoenix";
    }
    "Port-au-Prince";
    {
        "exemplarCity";
        "Port-au-Prince";
    }
    "Port_of_Spain";
    {
        "exemplarCity";
        "Port of Spain";
    }
    "Porto_Velho";
    {
        "exemplarCity";
        "Porto Velho";
    }
    "Puerto_Rico";
```



```
{
    "exemplarCity";
    "Puerto Rico";
}
"Rainy_River";
{
    "exemplarCity";
    "Rainy River";
}
"Rankin_Inlet";
{
    "exemplarCity";
    "Rankin Inlet";
}
"Recife";
{
    "exemplarCity";
    "Recife";
}
"Regina";
{
    "exemplarCity";
    "Regina";
}
"Resolute";
{
    "exemplarCity";
    "Resolute";
}
"Rio_Branco";
{
    "exemplarCity";
    "Rio Branco";
}
"Santa_Isabel";
{
    "exemplarCity";
    "Santa Isabel";
}
"Santarem";
{
    "exemplarCity";
    "Santarem";
}
"Santiago";
{
    "exemplarCity";
    "Santiago";
}
"Santo_Domingo";
{
    "exemplarCity";
    "Santo Domingo";
}
"Sao_Paulo";
{
    "exemplarCity";
```

```
        "São Paulo";
    }
    "Scoresbysund";
    {
        "exemplarCity";
        "Ittoqqortoormiit";
    }
    "Sitka";
    {
        "exemplarCity";
        "Sitka";
    }
    "St_Barthelemy";
    {
        "exemplarCity";
        "Saint-Barthélemy";
    }
    "St_Johns";
    {
        "exemplarCity";
        "St. John's";
    }
    "St_Kitts";
    {
        "exemplarCity";
        "St. Kitts";
    }
    "St_Lucia";
    {
        "exemplarCity";
        "St. Lucia";
    }
    "St_Thomas";
    {
        "exemplarCity";
        "St. Thomas";
    }
    "St_Vincent";
    {
        "exemplarCity";
        "St. Vincent";
    }
    "Swift_Current";
    {
        "exemplarCity";
        "Swift Current";
    }
    "Tegucigalpa";
    {
        "exemplarCity";
        "Tegucigalpa";
    }
    "Thule";
    {
        "exemplarCity";
        "Thule";
    }
}
```

```
"Thunder_Bay";
{
  "exemplarCity";
  "Thunder Bay";
}
"Tijuana";
{
  "exemplarCity";
  "Tijuana";
}
"Toronto";
{
  "exemplarCity";
  "Toronto";
}
"Tortola";
{
  "exemplarCity";
  "Tortola";
}
"Vancouver";
{
  "exemplarCity";
  "Vancouver";
}
"Whitehorse";
{
  "exemplarCity";
  "Whitehorse";
}
"Winnipeg";
{
  "exemplarCity";
  "Winnipeg";
}
"Yakutat";
{
  "exemplarCity";
  "Yakutat";
}
"Yellowknife";
{
  "exemplarCity";
  "Yellowknife";
}
}
"Atlantic";
{
  "Azores";
  {
    "exemplarCity";
    "Azoren";
  }
  "Bermuda";
  {
    "exemplarCity";
    "Bermudas";
  }
}
```

```
}
"Canary";
{
  "exemplarCity";
  "Kanaren";
}
"Cape_Verde";
{
  "exemplarCity";
  "Cabo Verde";
}
"Faeroe";
{
  "exemplarCity";
  "Färöer";
}
"Madeira";
{
  "exemplarCity";
  "Madeira";
}
"Reykjavik";
{
  "exemplarCity";
  "Reykjavík";
}
"South_Georgia";
{
  "exemplarCity";
  "Südgeorgien";
}
"St_Helena";
{
  "exemplarCity";
  "St. Helena";
}
"Stanley";
{
  "exemplarCity";
  "Stanley";
}
}
"Europe";
{
  "Amsterdam";
  {
    "exemplarCity";
    "Amsterdam";
  }
  "Andorra";
  {
    "exemplarCity";
    "Andorra";
  }
  "Astrakhan";
  {
    "exemplarCity";
```

```
        "Astrachan";
    }
    "Athens";
    {
        "exemplarCity";
        "Athen";
    }
    "Belgrade";
    {
        "exemplarCity";
        "Belgrad";
    }
    "Berlin";
    {
        "exemplarCity";
        "Berlin";
    }
    "Bratislava";
    {
        "exemplarCity";
        "Bratislava";
    }
    "Brussels";
    {
        "exemplarCity";
        "Brüssel";
    }
    "Bucharest";
    {
        "exemplarCity";
        "Bukarest";
    }
    "Budapest";
    {
        "exemplarCity";
        "Budapest";
    }
    "Busingen";
    {
        "exemplarCity";
        "Büsingen";
    }
    "Chisinau";
    {
        "exemplarCity";
        "Kischinau";
    }
    "Copenhagen";
    {
        "exemplarCity";
        "Kopenhagen";
    }
    "Dublin";
    {
        "long";
        {
            "daylight";
```

```
        "Irische Sommerzeit";
    }
    "exemplarCity";
    "Dublin";
}
"Gibraltar";
{
    "exemplarCity";
    "Gibraltar";
}
"Guernsey";
{
    "exemplarCity";
    "Guernsey";
}
"Helsinki";
{
    "exemplarCity";
    "Helsinki";
}
"Isle_of_Man";
{
    "exemplarCity";
    "Isle of Man";
}
"Istanbul";
{
    "exemplarCity";
    "Istanbul";
}
"Jersey";
{
    "exemplarCity";
    "Jersey";
}
"Kaliningrad";
{
    "exemplarCity";
    "Kaliningrad";
}
"Kiev";
{
    "exemplarCity";
    "Kiew";
}
"Kirov";
{
    "exemplarCity";
    "Kirow";
}
"Lisbon";
{
    "exemplarCity";
    "Lissabon";
}
"Ljubljana";
{
```

```
        "exemplarCity";
        "Ljubljana";
    }
    "London";
    {
        "long";
        {
            "daylight";
            "Britische Sommerzeit";
        }
        "exemplarCity";
        "London";
    }
    "Luxembourg";
    {
        "exemplarCity";
        "Luxemburg";
    }
    "Madrid";
    {
        "exemplarCity";
        "Madrid";
    }
    "Malta";
    {
        "exemplarCity";
        "Malta";
    }
    "Mariehamn";
    {
        "exemplarCity";
        "Mariehamn";
    }
    "Minsk";
    {
        "exemplarCity";
        "Minsk";
    }
    "Monaco";
    {
        "exemplarCity";
        "Monaco";
    }
    "Moscow";
    {
        "exemplarCity";
        "Moskau";
    }
    "Oslo";
    {
        "exemplarCity";
        "Oslo";
    }
    "Paris";
    {
        "exemplarCity";
        "Paris";
    }
}
```

```
}
"Podgorica";
{
  "exemplarCity";
  "Podgorica";
}
"Prague";
{
  "exemplarCity";
  "Prag";
}
"Riga";
{
  "exemplarCity";
  "Riga";
}
"Rome";
{
  "exemplarCity";
  "Rom";
}
"Samara";
{
  "exemplarCity";
  "Samara";
}
"San_Marino";
{
  "exemplarCity";
  "San Marino";
}
"Sarajevo";
{
  "exemplarCity";
  "Sarajevo";
}
"Simferopol";
{
  "exemplarCity";
  "Simferopol";
}
"Skopje";
{
  "exemplarCity";
  "Skopje";
}
"Sofia";
{
  "exemplarCity";
  "Sofia";
}
"Stockholm";
{
  "exemplarCity";
  "Stockholm";
}
"Tallinn";
```



```
{
    "exemplarCity";
    "Tallinn";
}
"Tirane";
{
    "exemplarCity";
    "Tirana";
}
"Ulyanovsk";
{
    "exemplarCity";
    "Uljanowsk";
}
"Uzhgorod";
{
    "exemplarCity";
    "Uschgorod";
}
"Vaduz";
{
    "exemplarCity";
    "Vaduz";
}
"Vatican";
{
    "exemplarCity";
    "Vatikan";
}
"Vienna";
{
    "exemplarCity";
    "Wien";
}
"Vilnius";
{
    "exemplarCity";
    "Vilnius";
}
"Volgograd";
{
    "exemplarCity";
    "Wolgograd";
}
"Warsaw";
{
    "exemplarCity";
    "Warschau";
}
"Zagreb";
{
    "exemplarCity";
    "Zagreb";
}
"Zaporozhye";
{
    "exemplarCity";
```

```
        "Saporischja";
    }
    "Zurich";
    {
        "exemplarCity";
        "Zürich";
    }
}
"Africa";
{
    "Abidjan";
    {
        "exemplarCity";
        "Abidjan";
    }
    "Accra";
    {
        "exemplarCity";
        "Accra";
    }
    "Addis_Ababa";
    {
        "exemplarCity";
        "Addis Abeba";
    }
    "Algiers";
    {
        "exemplarCity";
        "Algier";
    }
    "Asmera";
    {
        "exemplarCity";
        "Asmara";
    }
    "Bamako";
    {
        "exemplarCity";
        "Bamako";
    }
    "Bangui";
    {
        "exemplarCity";
        "Bangui";
    }
    "Banjul";
    {
        "exemplarCity";
        "Banjul";
    }
    "Bissau";
    {
        "exemplarCity";
        "Bissau";
    }
    "Blantyre";
    {
```

```
        "exemplarCity";
        "Blantyre";
    }
    "Brazzaville";
    {
        "exemplarCity";
        "Brazzaville";
    }
    "Bujumbura";
    {
        "exemplarCity";
        "Bujumbura";
    }
    "Cairo";
    {
        "exemplarCity";
        "Kairo";
    }
    "Casablanca";
    {
        "exemplarCity";
        "Casablanca";
    }
    "Ceuta";
    {
        "exemplarCity";
        "Ceuta";
    }
    "Conakry";
    {
        "exemplarCity";
        "Conakry";
    }
    "Dakar";
    {
        "exemplarCity";
        "Dakar";
    }
    "Dar_es_Salaam";
    {
        "exemplarCity";
        "Daressalam";
    }
    "Djibouti";
    {
        "exemplarCity";
        "Dschibuti";
    }
    "Douala";
    {
        "exemplarCity";
        "Douala";
    }
    "El_Aaiun";
    {
        "exemplarCity";
        "El Aaiún";
    }
```

```
}
"Freetown";
{
  "exemplarCity";
  "Freetown";
}
"Gaborone";
{
  "exemplarCity";
  "Gaborone";
}
"Harare";
{
  "exemplarCity";
  "Harare";
}
"Johannesburg";
{
  "exemplarCity";
  "Johannesburg";
}
"Juba";
{
  "exemplarCity";
  "Juba";
}
"Kampala";
{
  "exemplarCity";
  "Kampala";
}
"Khartoum";
{
  "exemplarCity";
  "Khartum";
}
"Kigali";
{
  "exemplarCity";
  "Kigali";
}
"Kinshasa";
{
  "exemplarCity";
  "Kinshasa";
}
"Lagos";
{
  "exemplarCity";
  "Lagos";
}
"Libreville";
{
  "exemplarCity";
  "Libreville";
}
"Lome";
```

```
{
    "exemplarCity";
    "Lomé";
}
"Luanda";
{
    "exemplarCity";
    "Luanda";
}
"Lubumbashi";
{
    "exemplarCity";
    "Lubumbashi";
}
"Lusaka";
{
    "exemplarCity";
    "Lusaka";
}
"Malabo";
{
    "exemplarCity";
    "Malabo";
}
"Maputo";
{
    "exemplarCity";
    "Maputo";
}
"Maseru";
{
    "exemplarCity";
    "Maseru";
}
"Mbabane";
{
    "exemplarCity";
    "Mbabane";
}
"Mogadishu";
{
    "exemplarCity";
    "Mogadishu";
}
"Monrovia";
{
    "exemplarCity";
    "Monrovia";
}
"Nairobi";
{
    "exemplarCity";
    "Nairobi";
}
"Ndjamena";
{
    "exemplarCity";
```

```
        "N'Djamena";
    }
    "Niamey";
    {
        "exemplarCity";
        "Niamey";
    }
    "Nouakchott";
    {
        "exemplarCity";
        "Nouakchott";
    }
    "Ouagadougou";
    {
        "exemplarCity";
        "Ouagadougou";
    }
    "Porto-Novo";
    {
        "exemplarCity";
        "Porto Novo";
    }
    "Sao_Tome";
    {
        "exemplarCity";
        "São Tomé";
    }
    "Tripoli";
    {
        "exemplarCity";
        "Tripolis";
    }
    "Tunis";
    {
        "exemplarCity";
        "Tunis";
    }
    "Windhoek";
    {
        "exemplarCity";
        "Windhoek";
    }
}
"Asia";
{
    "Aden";
    {
        "exemplarCity";
        "Aden";
    }
    "Almaty";
    {
        "exemplarCity";
        "Almaty";
    }
    "Amman";
    {
```

```
        "exemplarCity";
        "Amman";
    }
    "Anadyr";
    {
        "exemplarCity";
        "Anadyr";
    }
    "Aqtau";
    {
        "exemplarCity";
        "Aqtau";
    }
    "Aqtobe";
    {
        "exemplarCity";
        "Aktobe";
    }
    "Ashgabat";
    {
        "exemplarCity";
        "Aşgabat";
    }
    "Baghdad";
    {
        "exemplarCity";
        "Bagdad";
    }
    "Bahrain";
    {
        "exemplarCity";
        "Bahrain";
    }
    "Baku";
    {
        "exemplarCity";
        "Baku";
    }
    "Bangkok";
    {
        "exemplarCity";
        "Bangkok";
    }
    "Barnaul";
    {
        "exemplarCity";
        "Barnaul";
    }
    "Beirut";
    {
        "exemplarCity";
        "Beirut";
    }
    "Bishkek";
    {
        "exemplarCity";
        "Bischkek";
    }
```

```
}
"Brunei";
{
  "exemplarCity";
  "Brunei";
}
"Calcutta";
{
  "exemplarCity";
  "Kalkutta";
}
"Chita";
{
  "exemplarCity";
  "Tschita";
}
"Choibalsan";
{
  "exemplarCity";
  "Tschoibalsan";
}
"Colombo";
{
  "exemplarCity";
  "Colombo";
}
"Damascus";
{
  "exemplarCity";
  "Damaskus";
}
"Dhaka";
{
  "exemplarCity";
  "Dhaka";
}
"Dili";
{
  "exemplarCity";
  "Dili";
}
"Dubai";
{
  "exemplarCity";
  "Dubai";
}
"Dushanbe";
{
  "exemplarCity";
  "Duschanbe";
}
"Gaza";
{
  "exemplarCity";
  "Gaza";
}
"Hebron";
```



```
{
    "exemplarCity";
    "Hebron";
}
"Hong_Kong";
{
    "exemplarCity";
    "Hongkong";
}
"Hovd";
{
    "exemplarCity";
    "Chowd";
}
"Irkutsk";
{
    "exemplarCity";
    "Irkutsk";
}
"Jakarta";
{
    "exemplarCity";
    "Jakarta";
}
"Jayapura";
{
    "exemplarCity";
    "Jayapura";
}
"Jerusalem";
{
    "exemplarCity";
    "Jerusalem";
}
"Kabul";
{
    "exemplarCity";
    "Kabul";
}
"Kamchatka";
{
    "exemplarCity";
    "Kamtschatka";
}
"Karachi";
{
    "exemplarCity";
    "Karatschi";
}
"Katmandu";
{
    "exemplarCity";
    "Kathmandu";
}
"Khandyga";
{
    "exemplarCity";
```

```
        "Chandyga";
    }
    "Krasnoyarsk";
    {
        "exemplarCity";
        "Krasnojarsk";
    }
    "Kuala_Lumpur";
    {
        "exemplarCity";
        "Kuala Lumpur";
    }
    "Kuching";
    {
        "exemplarCity";
        "Kuching";
    }
    "Kuwait";
    {
        "exemplarCity";
        "Kuwait";
    }
    "Macau";
    {
        "exemplarCity";
        "Macao";
    }
    "Magadan";
    {
        "exemplarCity";
        "Magadan";
    }
    "Makassar";
    {
        "exemplarCity";
        "Makassar";
    }
    "Manila";
    {
        "exemplarCity";
        "Manila";
    }
    "Muscat";
    {
        "exemplarCity";
        "Maskat";
    }
    "Nicosia";
    {
        "exemplarCity";
        "Nikosia";
    }
    "Novokuznetsk";
    {
        "exemplarCity";
        "Nowokuznetsk";
    }
}
```

```
"Novosibirsk";
{
  "exemplarCity";
  "Nowosibirsk";
}
"Omsk";
{
  "exemplarCity";
  "Omsk";
}
"Oral";
{
  "exemplarCity";
  "Oral";
}
"Phnom_Penh";
{
  "exemplarCity";
  "Phnom Penh";
}
"Pontianak";
{
  "exemplarCity";
  "Pontianak";
}
"Pyongyang";
{
  "exemplarCity";
  "Pjöngjang";
}
"Qatar";
{
  "exemplarCity";
  "Katar";
}
"Qyzylorda";
{
  "exemplarCity";
  "Qysylorda";
}
"Rangoon";
{
  "exemplarCity";
  "Rangun";
}
"Riyadh";
{
  "exemplarCity";
  "Riad";
}
"Saigon";
{
  "exemplarCity";
  "Ho-Chi-Minh-Stadt";
}
"Sakhalin";
{
```

```
        "exemplarCity";
        "Sachalin";
    }
    "Samarkand";
    {
        "exemplarCity";
        "Samarkand";
    }
    "Seoul";
    {
        "exemplarCity";
        "Seoul";
    }
    "Shanghai";
    {
        "exemplarCity";
        "Shanghai";
    }
    "Singapore";
    {
        "exemplarCity";
        "Singapur";
    }
    "Srednekolymsk";
    {
        "exemplarCity";
        "Srednekolymsk";
    }
    "Taipei";
    {
        "exemplarCity";
        "Taipeh";
    }
    "Tashkent";
    {
        "exemplarCity";
        "Taschkent";
    }
    "Tbilisi";
    {
        "exemplarCity";
        "Tiflis";
    }
    "Tehran";
    {
        "exemplarCity";
        "Teheran";
    }
    "Thimphu";
    {
        "exemplarCity";
        "Thimphu";
    }
    "Tokyo";
    {
        "exemplarCity";
        "Tokio";
    }
```

```
}
  "Tomsk";
  {
    "exemplarCity";
    "Tomsk";
  }
  "Ulaanbaatar";
  {
    "exemplarCity";
    "Ulaanbaatar";
  }
  "Urumqi";
  {
    "exemplarCity";
    "Ürümqi";
  }
  "Ust-Nera";
  {
    "exemplarCity";
    "Ust-Nera";
  }
  "Vientiane";
  {
    "exemplarCity";
    "Vientiane";
  }
  "Vladivostok";
  {
    "exemplarCity";
    "Wladiwostok";
  }
  "Yakutsk";
  {
    "exemplarCity";
    "Jakutsk";
  }
  "Yekaterinburg";
  {
    "exemplarCity";
    "Jekaterinburg";
  }
  "Yerevan";
  {
    "exemplarCity";
    "Eriwan";
  }
}
"Indian";
{
  "Antananarivo";
  {
    "exemplarCity";
    "Antananarivo";
  }
  "Chagos";
  {
    "exemplarCity";
```

```
        "Chagos";
    }
    "Christmas";
    {
        "exemplarCity";
        "Weihnachtsinsel";
    }
    "Cocos";
    {
        "exemplarCity";
        "Cocos";
    }
    "Comoro";
    {
        "exemplarCity";
        "Komoren";
    }
    "Kerguelen";
    {
        "exemplarCity";
        "Kerguelen";
    }
    "Mahe";
    {
        "exemplarCity";
        "Mahe";
    }
    "Maldives";
    {
        "exemplarCity";
        "Malediven";
    }
    "Mauritius";
    {
        "exemplarCity";
        "Mauritius";
    }
    "Mayotte";
    {
        "exemplarCity";
        "Mayotte";
    }
    "Reunion";
    {
        "exemplarCity";
        "Réunion";
    }
}
"Australia";
{
    "Adelaide";
    {
        "exemplarCity";
        "Adelaide";
    }
    "Brisbane";
    {
```

```
        "exemplarCity";
        "Brisbane";
    }
    "Broken_Hill";
    {
        "exemplarCity";
        "Broken Hill";
    }
    "Currie";
    {
        "exemplarCity";
        "Currie";
    }
    "Darwin";
    {
        "exemplarCity";
        "Darwin";
    }
    "Eucla";
    {
        "exemplarCity";
        "Eucla";
    }
    "Hobart";
    {
        "exemplarCity";
        "Hobart";
    }
    "Lindeman";
    {
        "exemplarCity";
        "Lindeman";
    }
    "Lord_Howe";
    {
        "exemplarCity";
        "Lord Howe";
    }
    "Melbourne";
    {
        "exemplarCity";
        "Melbourne";
    }
    "Perth";
    {
        "exemplarCity";
        "Perth";
    }
    "Sydney";
    {
        "exemplarCity";
        "Sydney";
    }
    }
    "Pacific";
    {
        "Apia";
```

```
{
    "exemplarCity";
    "Apia";
}
"Auckland";
{
    "exemplarCity";
    "Auckland";
}
"Bougainville";
{
    "exemplarCity";
    "Bougainville";
}
"Chatham";
{
    "exemplarCity";
    "Chatham";
}
"Easter";
{
    "exemplarCity";
    "Osterinsel";
}
"Efate";
{
    "exemplarCity";
    "Efate";
}
"Enderbury";
{
    "exemplarCity";
    "Enderbury";
}
"Fakaofu";
{
    "exemplarCity";
    "Fakaofu";
}
"Fiji";
{
    "exemplarCity";
    "Fidschi";
}
"Funafuti";
{
    "exemplarCity";
    "Funafuti";
}
"Galapagos";
{
    "exemplarCity";
    "Galapagos";
}
"Gambier";
{
    "exemplarCity";
```



```
        "Gambier";
    }
    "Guadalcanal";
    {
        "exemplarCity";
        "Guadalcanal";
    }
    "Guam";
    {
        "exemplarCity";
        "Guam";
    }
    "Honolulu";
    {
        "exemplarCity";
        "Honolulu";
    }
    "Johnston";
    {
        "exemplarCity";
        "Johnston";
    }
    "Kiritimati";
    {
        "exemplarCity";
        "Kiritimati";
    }
    "Kosrae";
    {
        "exemplarCity";
        "Kosrae";
    }
    "Kwajalein";
    {
        "exemplarCity";
        "Kwajalein";
    }
    "Majuro";
    {
        "exemplarCity";
        "Majuro";
    }
    "Marquesas";
    {
        "exemplarCity";
        "Marquesas";
    }
    "Midway";
    {
        "exemplarCity";
        "Midway";
    }
    "Nauru";
    {
        "exemplarCity";
        "Nauru";
    }
}
```

```
"Niue";
{
  "exemplarCity";
  "Niue";
}
"Norfolk";
{
  "exemplarCity";
  "Norfolk";
}
"Noumea";
{
  "exemplarCity";
  "Noumea";
}
"Pago_Pago";
{
  "exemplarCity";
  "Pago Pago";
}
"Palau";
{
  "exemplarCity";
  "Palau";
}
"Pitcairn";
{
  "exemplarCity";
  "Pitcairn";
}
"Ponape";
{
  "exemplarCity";
  "Pohnpei";
}
"Port_Moresby";
{
  "exemplarCity";
  "Port Moresby";
}
"Rarotonga";
{
  "exemplarCity";
  "Rarotonga";
}
"Saipan";
{
  "exemplarCity";
  "Saipan";
}
"Tahiti";
{
  "exemplarCity";
  "Tahiti";
}
"Tarawa";
{
```

```
        "exemplarCity";
        "Tarawa";
    }
    "Tongatapu";
    {
        "exemplarCity";
        "Tongatapu";
    }
    "Truk";
    {
        "exemplarCity";
        "Chuuk";
    }
    "Wake";
    {
        "exemplarCity";
        "Wake";
    }
    "Wallis";
    {
        "exemplarCity";
        "Wallis";
    }
}
"Arctic";
{
    "Longyearbyen";
    {
        "exemplarCity";
        "Longyearbyen";
    }
}
"Antarctica";
{
    "Casey";
    {
        "exemplarCity";
        "Casey";
    }
    "Davis";
    {
        "exemplarCity";
        "Davis";
    }
    "DumontDUrville";
    {
        "exemplarCity";
        "Dumont d'Urville";
    }
    "Macquarie";
    {
        "exemplarCity";
        "Macquarie";
    }
    "Mawson";
    {
        "exemplarCity";
```

```

        "Mawson";
    }
    "McMurdo";
    {
        "exemplarCity";
        "McMurdo";
    }
    "Palmer";
    {
        "exemplarCity";
        "Palmer";
    }
    "Rothera";
    {
        "exemplarCity";
        "Rothera";
    }
    "Syowa";
    {
        "exemplarCity";
        "Syowa";
    }
    "Troll";
    {
        "exemplarCity";
        "Troll";
    }
    "Vostok";
    {
        "exemplarCity";
        "Wostok";
    }
}
"Etc";
{
    "GMT";
    {
        "exemplarCity";
        "GMT";
    }
    "GMT1";
    {
        "exemplarCity";
        "GMT+1";
    }
    "GMT10";
    {
        "exemplarCity";
        "GMT+10";
    }
    "GMT11";
    {
        "exemplarCity";
        "GMT+11";
    }
    "GMT12";
    {

```

```
        "exemplarCity";
        "GMT+12";
    }
    "GMT2";
    {
        "exemplarCity";
        "GMT+2";
    }
    "GMT3";
    {
        "exemplarCity";
        "GMT+3";
    }
    "GMT4";
    {
        "exemplarCity";
        "GMT+4";
    }
    "GMT5";
    {
        "exemplarCity";
        "GMT+5";
    }
    "GMT6";
    {
        "exemplarCity";
        "GMT+6";
    }
    "GMT7";
    {
        "exemplarCity";
        "GMT+7";
    }
    "GMT8";
    {
        "exemplarCity";
        "GMT+8";
    }
    "GMT9";
    {
        "exemplarCity";
        "GMT+9";
    }
    "GMT-1";
    {
        "exemplarCity";
        "GMT-1";
    }
    "GMT-10";
    {
        "exemplarCity";
        "GMT-10";
    }
    "GMT-11";
    {
        "exemplarCity";
        "GMT-11";
    }
```

```
}
"GMT-12";
{
  "exemplarCity";
  "GMT-12";
}
"GMT-13";
{
  "exemplarCity";
  "GMT-13";
}
"GMT-14";
{
  "exemplarCity";
  "GMT-14";
}
"GMT-2";
{
  "exemplarCity";
  "GMT-2";
}
"GMT-3";
{
  "exemplarCity";
  "GMT-3";
}
"GMT-4";
{
  "exemplarCity";
  "GMT-4";
}
"GMT-5";
{
  "exemplarCity";
  "GMT-5";
}
"GMT-6";
{
  "exemplarCity";
  "GMT-6";
}
"GMT-7";
{
  "exemplarCity";
  "GMT-7";
}
"GMT-8";
{
  "exemplarCity";
  "GMT-8";
}
"GMT-9";
{
  "exemplarCity";
  "GMT-9";
}
}
"Unknown";
```

```

        {
            "exemplarCity";
            "Unbekannt";
        }
    }
}
"metazone";
{
    "Acre";
    {
        "long";
        {
            "generic";
            "Acre-Zeit",
            "standard";
            "Acre-Normalzeit",
            "daylight";
            "Acre-Sommerzeit";
        }
    }
    "Afghanistan";
    {
        "long";
        {
            "standard";
            "Afghanistan-Zeit";
        }
    }
    "Africa_Central";
    {
        "long";
        {
            "standard";
            "Zentralafrikanische Zeit";
        }
    }
    "Africa_Eastern";
    {
        "long";
        {
            "standard";
            "Ostafrikanische Zeit";
        }
    }
    "Africa_Southern";
    {
        "long";
        {
            "standard";
            "Südafrikanische Zeit";
        }
    }
    "Africa_Western";
    {
        "long";
        {
            "generic";

```

```

        "Westafrikanische Zeit",
        "standard";
        "Westafrikanische Normalzeit",
        "daylight";
        "Westafrikanische Sommerzeit";
    }
}
"Alaska";
{
    "long";
    {
        "generic";
        "Alaska-Zeit",
        "standard";
        "Alaska-Normalzeit",
        "daylight";
        "Alaska-Sommerzeit";
    }
}
"Almaty";
{
    "long";
    {
        "generic";
        "Almaty-Zeit",
        "standard";
        "Almaty-Normalzeit",
        "daylight";
        "Almaty-Sommerzeit";
    }
}
"Amazon";
{
    "long";
    {
        "generic";
        "Amazonas-Zeit",
        "standard";
        "Amazonas-Normalzeit",
        "daylight";
        "Amazonas-Sommerzeit";
    }
}
"America_Central";
{
    "long";
    {
        "generic";
        "Nordamerikanische Inlandzeit",
        "standard";
        "Nordamerikanische Inland-Normalzeit",
        "daylight";
        "Nordamerikanische Inland-Sommerzeit";
    }
}
"America_Eastern";
{

```



```
        "long";
        {
            "generic";
            "Nordamerikanische Ostküstenzeit",
            "standard";
            "Nordamerikanische Ostküsten-Normalzeit",
            "daylight";
            "Nordamerikanische Ostküsten-Sommerzeit";
        }
    }
    "America_Mountain";
    {
        "long";
        {
            "generic";
            "Rocky-Mountain-Zeit",
            "standard";
            "Rocky Mountain-Normalzeit",
            "daylight";
            "Rocky-Mountain-Sommerzeit";
        }
    }
    "America_Pacific";
    {
        "long";
        {
            "generic";
            "Nordamerikanische Westküstenzeit",
            "standard";
            "Nordamerikanische Westküsten-Normalzeit",
            "daylight";
            "Nordamerikanische Westküsten-Sommerzeit";
        }
    }
    "Anadyr";
    {
        "long";
        {
            "generic";
            "Anadyr Zeit",
            "standard";
            "Anadyr Normalzeit",
            "daylight";
            "Anadyr Sommerzeit";
        }
    }
    "Apia";
    {
        "long";
        {
            "generic";
            "Apia-Zeit",
            "standard";
            "Apia-Normalzeit",
            "daylight";
            "Apia-Sommerzeit";
        }
    }
}
```

```

    }
    "Aqtau";
    {
        "long";
        {
            "generic";
            "Aqtau-Zeit",
            "standard";
            "Aqtau-Normalzeit",
            "daylight";
            "Aqtau-Sommerzeit";
        }
    }
    "Aqtobe";
    {
        "long";
        {
            "generic";
            "Aqtöbe-Zeit",
            "standard";
            "Aqtöbe-Normalzeit",
            "daylight";
            "Aqtöbe-Sommerzeit";
        }
    }
    "Arabian";
    {
        "long";
        {
            "generic";
            "Arabische Zeit",
            "standard";
            "Arabische Normalzeit",
            "daylight";
            "Arabische Sommerzeit";
        }
    }
    "Argentina";
    {
        "long";
        {
            "generic";
            "Argentinische Zeit",
            "standard";
            "Argentinische Normalzeit",
            "daylight";
            "Argentinische Sommerzeit";
        }
    }
    "Argentina_Western";
    {
        "long";
        {
            "generic";
            "Westargentinische Zeit",
            "standard";
            "Westargentinische Normalzeit",

```

```

        "daylight";
        "Westargentinische Sommerzeit";
    }
}
"Armenia";
{
    "long";
    {
        "generic";
        "Armenische Zeit",
        "standard";
        "Armenische Normalzeit",
        "daylight";
        "Armenische Sommerzeit";
    }
}
"Atlantic";
{
    "long";
    {
        "generic";
        "Atlantik-Zeit",
        "standard";
        "Atlantik-Normalzeit",
        "daylight";
        "Atlantik-Sommerzeit";
    }
}
"Australia_Central";
{
    "long";
    {
        "generic";
        "Zentralaustralische Zeit",
        "standard";
        "Zentralaustralische Normalzeit",
        "daylight";
        "Zentralaustralische Sommerzeit";
    }
}
"Australia_CentralWestern";
{
    "long";
    {
        "generic";
        "Zentral-/Westaustralische Zeit",
        "standard";
        "Zentral-/Westaustralische Normalzeit",
        "daylight";
        "Zentral-/Westaustralische Sommerzeit";
    }
}
"Australia_Eastern";
{
    "long";
    {
        "generic";

```

```

        "Ostaustralische Zeit",
        "standard";
        "Ostaustralische Normalzeit",
        "daylight";
        "Ostaustralische Sommerzeit";
    }
}
"Australia_Western";
{
    "long";
    {
        "generic";
        "Westaustralische Zeit",
        "standard";
        "Westaustralische Normalzeit",
        "daylight";
        "Westaustralische Sommerzeit";
    }
}
"Azerbaijan";
{
    "long";
    {
        "generic";
        "Aserbaidsschanische Zeit",
        "standard";
        "Aserbeidschanische Normalzeit",
        "daylight";
        "Aserbaidsschanische Sommerzeit";
    }
}
"Azores";
{
    "long";
    {
        "generic";
        "Azoren-Zeit",
        "standard";
        "Azoren-Normalzeit",
        "daylight";
        "Azoren-Sommerzeit";
    }
}
"Bangladesh";
{
    "long";
    {
        "generic";
        "Bangladesch-Zeit",
        "standard";
        "Bangladesch-Normalzeit",
        "daylight";
        "Bangladesch-Sommerzeit";
    }
}
"Bhutan";
{

```

```

        "long";
        {
            "standard";
            "Bhutan-Zeit";
        }
    }
    "Bolivia";
    {
        "long";
        {
            "standard";
            "Bolivianische Zeit";
        }
    }
    "Brasilia";
    {
        "long";
        {
            "generic";
            "Brasília-Zeit",
            "standard";
            "Brasília-Normalzeit",
            "daylight";
            "Brasília-Sommerzeit";
        }
    }
    "Brunei";
    {
        "long";
        {
            "standard";
            "Brunei-Zeit";
        }
    }
    "Cape_Verde";
    {
        "long";
        {
            "generic";
            "Cabo-Verde-Zeit",
            "standard";
            "Cabo-Verde-Normalzeit",
            "daylight";
            "Cabo-Verde-Sommerzeit";
        }
    }
    "Casey";
    {
        "long";
        {
            "standard";
            "Casey-Zeit";
        }
    }
    "Chamorro";
    {
        "long";

```

```
        {
            "standard";
            "Chamorro-Zeit";
        }
    }
    "Chatham";
    {
        "long";
        {
            "generic";
            "Chatham-Zeit",
            "standard";
            "Chatham-Normalzeit",
            "daylight";
            "Chatham-Sommerzeit";
        }
    }
    "Chile";
    {
        "long";
        {
            "generic";
            "Chilenische Zeit",
            "standard";
            "Chilenische Normalzeit",
            "daylight";
            "Chilenische Sommerzeit";
        }
    }
    "China";
    {
        "long";
        {
            "generic";
            "Chinesische Zeit",
            "standard";
            "Chinesische Normalzeit",
            "daylight";
            "Chinesische Sommerzeit";
        }
    }
    "Choibalsan";
    {
        "long";
        {
            "generic";
            "Tschoibalsan-Zeit",
            "standard";
            "Tschoibalsan-Normalzeit",
            "daylight";
            "Tschoibalsan-Sommerzeit";
        }
    }
    "Christmas";
    {
        "long";
        {
```

```
        "standard";
        "Weihnachtsinsel-Zeit";
    }
}
"Cocos";
{
    "long";
    {
        "standard";
        "Kokosinseln-Zeit";
    }
}
"Colombia";
{
    "long";
    {
        "generic";
        "Kolumbianische Zeit",
        "standard";
        "Kolumbianische Normalzeit",
        "daylight";
        "Kolumbianische Sommerzeit";
    }
}
"Cook";
{
    "long";
    {
        "generic";
        "Cookinseln-Zeit",
        "standard";
        "Cookinseln-Normalzeit",
        "daylight";
        "Cookinseln-Sommerzeit";
    }
}
"Cuba";
{
    "long";
    {
        "generic";
        "Kubanische Zeit",
        "standard";
        "Kubanische Normalzeit",
        "daylight";
        "Kubanische Sommerzeit";
    }
}
"Davis";
{
    "long";
    {
        "standard";
        "Davis-Zeit";
    }
}
"DumontDUrville";
```

```

    {
      "long";
      {
        "standard";
        "Dumont-d'Urville-Zeit";
      }
    }
    "East_Timor";
    {
      "long";
      {
        "standard";
        "Osttimor-Zeit";
      }
    }
    "Easter";
    {
      "long";
      {
        "generic";
        "Osterinsel-Zeit",
        "standard";
        "Osterinsel-Normalzeit",
        "daylight";
        "Osterinsel-Sommerzeit";
      }
    }
    "Ecuador";
    {
      "long";
      {
        "standard";
        "Ecuadorianische Zeit";
      }
    }
    "Europe_Central";
    {
      "long";
      {
        "generic";
        "Mitteleuropäische Zeit",
        "standard";
        "Mitteleuropäische Normalzeit",
        "daylight";
        "Mitteleuropäische Sommerzeit";
      }
      "short";
      {
        "generic";
        "MEZ",
        "standard";
        "MEZ",
        "daylight";
        "MESZ";
      }
    }
    "Europe_Eastern";

```



```
{
  "long";
  {
    "generic";
    "Osteuropäische Zeit",
    "standard";
    "Osteuropäische Normalzeit",
    "daylight";
    "Osteuropäische Sommerzeit";
  }
  "short";
  {
    "generic";
    "OEZ",
    "standard";
    "OEZ",
    "daylight";
    "OESZ";
  }
}
"Europe_Further_Eastern";
{
  "long";
  {
    "standard";
    "Kaliningrader Zeit";
  }
}
"Europe_Western";
{
  "long";
  {
    "generic";
    "Westeuropäische Zeit",
    "standard";
    "Westeuropäische Normalzeit",
    "daylight";
    "Westeuropäische Sommerzeit";
  }
  "short";
  {
    "generic";
    "WEZ",
    "standard";
    "WEZ",
    "daylight";
    "WESZ";
  }
}
"Falkland";
{
  "long";
  {
    "generic";
    "Falklandinseln-Zeit",
    "standard";
    "Falklandinseln-Normalzeit",
```

```

        "daylight";
        "Falklandinseln-Sommerzeit";
    }
}
"Fiji";
{
    "long";
    {
        "generic";
        "Fidschi-Zeit",
        "standard";
        "Fidschi-Normalzeit",
        "daylight";
        "Fidschi-Sommerzeit";
    }
}
"French_Guiana";
{
    "long";
    {
        "standard";
        "Französisch-Guayana-Zeit";
    }
}
"French_Southern";
{
    "long";
    {
        "standard";
        "Französische Süd- und Antarktisgebiete-
Zeit";
    }
}
"Galapagos";
{
    "long";
    {
        "standard";
        "Galapagos-Zeit";
    }
}
"Gambier";
{
    "long";
    {
        "standard";
        "Gambier-Zeit";
    }
}
"Georgia";
{
    "long";
    {
        "generic";
        "Georgische Zeit",
        "standard";
        "Georgische Normalzeit",

```

```

        "daylight";
        "Georgische Sommerzeit";
    }
}
"Gilbert_Islands";
{
    "long";
    {
        "standard";
        "Gilbert-Inseln-Zeit";
    }
}
"GMT";
{
    "long";
    {
        "standard";
        "Mittlere Greenwich-Zeit";
    }
}
"Greenland_Eastern";
{
    "long";
    {
        "generic";
        "Ostgrönland-Zeit",
        "standard";
        "Ostgrönland-Normalzeit",
        "daylight";
        "Ostgrönland-Sommerzeit";
    }
}
"Greenland_Western";
{
    "long";
    {
        "generic";
        "Westgrönland-Zeit",
        "standard";
        "Westgrönland-Normalzeit",
        "daylight";
        "Westgrönland-Sommerzeit";
    }
}
"Guam";
{
    "long";
    {
        "standard";
        "Guam-Zeit";
    }
}
"Gulf";
{
    "long";
    {
        "standard";
    }
}

```

```

        "Golf-Zeit";
    }
}
"Guyana";
{
    "long";
    {
        "standard";
        "Guyana-Zeit";
    }
}
"Hawaii_Aleutian";
{
    "long";
    {
        "generic";
        "Hawaii-Aleuten-Zeit",
        "standard";
        "Hawaii-Aleuten-Normalzeit",
        "daylight";
        "Hawaii-Aleuten-Sommerzeit";
    }
}
"Hong_Kong";
{
    "long";
    {
        "generic";
        "Hongkong-Zeit",
        "standard";
        "Hongkong-Normalzeit",
        "daylight";
        "Hongkong-Sommerzeit";
    }
}
"Hovd";
{
    "long";
    {
        "generic";
        "Chowd-Zeit",
        "standard";
        "Chowd-Normalzeit",
        "daylight";
        "Chowd-Sommerzeit";
    }
}
"India";
{
    "long";
    {
        "standard";
        "Indische Zeit";
    }
}
"Indian_Ocean";
{

```

```
        "long";
        {
            "standard";
            "Indischer Ozean-Zeit";
        }
    }
    "Indochina";
    {
        "long";
        {
            "standard";
            "Indochina-Zeit";
        }
    }
    "Indonesia_Central";
    {
        "long";
        {
            "standard";
            "Zentralindonesische Zeit";
        }
    }
    "Indonesia_Eastern";
    {
        "long";
        {
            "standard";
            "Ostindonesische Zeit";
        }
    }
    "Indonesia_Western";
    {
        "long";
        {
            "standard";
            "Westindonesische Zeit";
        }
    }
    "Iran";
    {
        "long";
        {
            "generic";
            "Iranische Zeit",
            "standard";
            "Iranische Normalzeit",
            "daylight";
            "Iranische Sommerzeit";
        }
    }
    "Irkutsk";
    {
        "long";
        {
            "generic";
            "Irkutsk-Zeit",
            "standard";
```

```

        "Irkutsk-Normalzeit",
        "daylight";
        "Irkutsk-Sommerzeit";
    }
}
"Israel";
{
    "long";
    {
        "generic";
        "Israelische Zeit",
        "standard";
        "Israelische Normalzeit",
        "daylight";
        "Israelische Sommerzeit";
    }
}
"Japan";
{
    "long";
    {
        "generic";
        "Japanische Zeit",
        "standard";
        "Japanische Normalzeit",
        "daylight";
        "Japanische Sommerzeit";
    }
}
"Kamchatka";
{
    "long";
    {
        "generic";
        "Kamtschatka-Zeit",
        "standard";
        "Kamtschatka-Normalzeit",
        "daylight";
        "Kamtschatka-Sommerzeit";
    }
}
"Kazakhstan_Eastern";
{
    "long";
    {
        "standard";
        "Ostkasachische Zeit";
    }
}
"Kazakhstan_Western";
{
    "long";
    {
        "standard";
        "Westkasachische Zeit";
    }
}
}

```

```
"Korea";
{
  "long";
  {
    "generic";
    "Koreanische Zeit",
    "standard";
    "Koreanische Normalzeit",
    "daylight";
    "Koreanische Sommerzeit";
  }
}
"Kosrae";
{
  "long";
  {
    "standard";
    "Kosrae-Zeit";
  }
}
"Krasnoyarsk";
{
  "long";
  {
    "generic";
    "Krasnojarsk-Zeit",
    "standard";
    "Krasnojarsk-Normalzeit",
    "daylight";
    "Krasnojarsk-Sommerzeit";
  }
}
"Kyrgystan";
{
  "long";
  {
    "standard";
    "Kirgisistan-Zeit";
  }
}
"Lanka";
{
  "long";
  {
    "standard";
    "Sri-Lanka-Zeit";
  }
}
"Line_Islands";
{
  "long";
  {
    "standard";
    "Linieninseln-Zeit";
  }
}
"Lord_Howe";
```

```

    {
      "long";
      {
        "generic";
        "Lord-Howe-Zeit",
        "standard";
        "Lord-Howe-Normalzeit",
        "daylight";
        "Lord-Howe-Sommerzeit";
      }
    }
    "Macau";
    {
      "long";
      {
        "generic";
        "Macau-Zeit",
        "standard";
        "Macau-Normalzeit",
        "daylight";
        "Macau-Sommerzeit";
      }
    }
    "Macquarie";
    {
      "long";
      {
        "standard";
        "Macquarieinsel-Zeit";
      }
    }
    "Magadan";
    {
      "long";
      {
        "generic";
        "Magadan-Zeit",
        "standard";
        "Magadan-Normalzeit",
        "daylight";
        "Magadan-Sommerzeit";
      }
    }
    "Malaysia";
    {
      "long";
      {
        "standard";
        "Malaysische Zeit";
      }
    }
    "Maldives";
    {
      "long";
      {
        "standard";
        "Malediven-Zeit";
      }
    }
  }

```



```
    }  
  }  
  "Marquesas";  
  {  
    "long";  
    {  
      "standard";  
      "Marquesas-Zeit";  
    }  
  }  
  "Marshall_Islands";  
  {  
    "long";  
    {  
      "standard";  
      "Marshallinseln-Zeit";  
    }  
  }  
  "Mauritius";  
  {  
    "long";  
    {  
      "generic";  
      "Mauritius-Zeit",  
      "standard";  
      "Mauritius-Normalzeit",  
      "daylight";  
      "Mauritius-Sommerzeit";  
    }  
  }  
  "Mawson";  
  {  
    "long";  
    {  
      "standard";  
      "Mawson-Zeit";  
    }  
  }  
  "Mexico_Northwest";  
  {  
    "long";  
    {  
      "generic";  
      "Mexiko Nordwestliche Zone-Zeit",  
      "standard";  
      "Mexiko Nordwestliche Zone-Normalzeit",  
      "daylight";  
      "Mexiko Nordwestliche Zone-Sommerzeit";  
    }  
  }  
  "Mexico_Pacific";  
  {  
    "long";  
    {  
      "generic";  
      "Mexiko Pazifikzone-Zeit",  
      "standard";
```

```

        "Mexiko Pazifikzone-Normalzeit",
        "daylight";
        "Mexiko Pazifikzone-Sommerzeit";
    }
}
"Mongolia";
{
    "long";
    {
        "generic";
        "Ulaanbaatar-Zeit",
        "standard";
        "Ulaanbaatar-Normalzeit",
        "daylight";
        "Ulaanbaatar-Sommerzeit";
    }
}
"Moscow";
{
    "long";
    {
        "generic";
        "Moskauer Zeit",
        "standard";
        "Moskauer Normalzeit",
        "daylight";
        "Moskauer Sommerzeit";
    }
}
"Myanmar";
{
    "long";
    {
        "standard";
        "Myanmar-Zeit";
    }
}
"Nauru";
{
    "long";
    {
        "standard";
        "Nauru-Zeit";
    }
}
"Nepal";
{
    "long";
    {
        "standard";
        "Nepalesische Zeit";
    }
}
"New_Caledonia";
{
    "long";
    {

```

```
        "generic";
        "Neukaledonische Zeit",
        "standard";
        "Neukaledonische Normalzeit",
        "daylight";
        "Neukaledonische Sommerzeit";
    }
}
"New_Zealand";
{
    "long";
    {
        "generic";
        "Neuseeland-Zeit",
        "standard";
        "Neuseeland-Normalzeit",
        "daylight";
        "Neuseeland-Sommerzeit";
    }
}
"Newfoundland";
{
    "long";
    {
        "generic";
        "Neufundland-Zeit",
        "standard";
        "Neufundland-Normalzeit",
        "daylight";
        "Neufundland-Sommerzeit";
    }
}
"Niue";
{
    "long";
    {
        "standard";
        "Niue-Zeit";
    }
}
"Norfolk";
{
    "long";
    {
        "standard";
        "Norfolkinsel-Zeit";
    }
}
"Noronha";
{
    "long";
    {
        "generic";
        "Fernando de Noronha-Zeit",
        "standard";
        "Fernando de Noronha-Normalzeit",
        "daylight";
    }
}
```

```

        "Fernando de Noronha-Sommerzeit";
    }
}
"North_Mariana";
{
    "long";
    {
        "standard";
        "Nördliche-Marianen-Zeit";
    }
}
"Novosibirsk";
{
    "long";
    {
        "generic";
        "Nowosibirsk-Zeit",
        "standard";
        "Nowosibirsk-Normalzeit",
        "daylight";
        "Nowosibirsk-Sommerzeit";
    }
}
"Omsk";
{
    "long";
    {
        "generic";
        "Omsk-Zeit",
        "standard";
        "Omsk-Normalzeit",
        "daylight";
        "Omsk-Sommerzeit";
    }
}
"Pakistan";
{
    "long";
    {
        "generic";
        "Pakistanische Zeit",
        "standard";
        "Pakistanische Normalzeit",
        "daylight";
        "Pakistanische Sommerzeit";
    }
}
"Palau";
{
    "long";
    {
        "standard";
        "Palau-Zeit";
    }
}
"Papua_New_Guinea";
{

```

```
        "long";
        {
            "standard";
            "Papua-Neuguinea-Zeit";
        }
    }
    "Paraguay";
    {
        "long";
        {
            "generic";
            "Paraguayanische Zeit",
            "standard";
            "Paraguayanische Normalzeit",
            "daylight";
            "Paraguayanische Sommerzeit";
        }
    }
    "Peru";
    {
        "long";
        {
            "generic";
            "Peruanische Zeit",
            "standard";
            "Peruanische Normalzeit",
            "daylight";
            "Peruanische Sommerzeit";
        }
    }
    "Philippines";
    {
        "long";
        {
            "generic";
            "Philippinische Zeit",
            "standard";
            "Philippinische Normalzeit",
            "daylight";
            "Philippinische Sommerzeit";
        }
    }
    "Phoenix_Islands";
    {
        "long";
        {
            "standard";
            "Phoenixinseln-Zeit";
        }
    }
    "Pierre_Miquelon";
    {
        "long";
        {
            "generic";
            "Saint-Pierre-und-Miquelon-Zeit",
            "standard";
```

```
        "Saint-Pierre-und-Miquelon-Normalzeit",
        "daylight";
        "Saint-Pierre-und-Miquelon-Sommerzeit";
    }
}
"Pitcairn";
{
    "long";
    {
        "standard";
        "Pitcairninseeln-Zeit";
    }
}
"Ponape";
{
    "long";
    {
        "standard";
        "Ponape-Zeit";
    }
}
"Pyongyang";
{
    "long";
    {
        "standard";
        "Pjöngjang-Zeit";
    }
}
"Qyzylorda";
{
    "long";
    {
        "generic";
        "Quysylorda-Zeit",
        "standard";
        "Quysylorda-Normalzeit",
        "daylight";
        "Qysylorda-Sommerzeit";
    }
}
"Reunion";
{
    "long";
    {
        "standard";
        "Réunion-Zeit";
    }
}
"Rothera";
{
    "long";
    {
        "standard";
        "Rothera-Zeit";
    }
}
}
```

```
"Sakhalin";
{
  "long";
  {
    "generic";
    "Sachalin-Zeit",
    "standard";
    "Sachalin-Normalzeit",
    "daylight";
    "Sachalin-Sommerzeit";
  }
}
"Samara";
{
  "long";
  {
    "generic";
    "Samara-Zeit",
    "standard";
    "Samara-Normalzeit",
    "daylight";
    "Samara-Sommerzeit";
  }
}
"Samoa";
{
  "long";
  {
    "generic";
    "Samoa-Zeit",
    "standard";
    "Samoa-Normalzeit",
    "daylight";
    "Samoa-Sommerzeit";
  }
}
"Seychelles";
{
  "long";
  {
    "standard";
    "Seychellen-Zeit";
  }
}
"Singapore";
{
  "long";
  {
    "standard";
    "Singapur-Zeit";
  }
}
"Solomon";
{
  "long";
  {
    "standard";
```

```
        "Salomoninseln-Zeit";
    }
}
"South_Georgia";
{
    "long";
    {
        "standard";
        "Südgeorgische Zeit";
    }
}
"Suriname";
{
    "long";
    {
        "standard";
        "Suriname-Zeit";
    }
}
"Syowa";
{
    "long";
    {
        "standard";
        "Syowa-Zeit";
    }
}
"Tahiti";
{
    "long";
    {
        "standard";
        "Tahiti-Zeit";
    }
}
"Taipei";
{
    "long";
    {
        "generic";
        "Taipeh-Zeit",
        "standard";
        "Taipeh-Normalzeit",
        "daylight";
        "Taipeh-Sommerzeit";
    }
}
"Tajikistan";
{
    "long";
    {
        "standard";
        "Tadschikistan-Zeit";
    }
}
"Tokelau";
{
```



```
        "long";
        {
            "standard";
            "Tokelau-Zeit";
        }
    }
    "Tonga";
    {
        "long";
        {
            "generic";
            "Tonganische Zeit",
            "standard";
            "Tonganische Normalzeit",
            "daylight";
            "Tonganische Sommerzeit";
        }
    }
    "Truk";
    {
        "long";
        {
            "standard";
            "Chuuk-Zeit";
        }
    }
    "Turkmenistan";
    {
        "long";
        {
            "generic";
            "Turkmenistan-Zeit",
            "standard";
            "Turkmenistan-Normalzeit",
            "daylight";
            "Turkmenistan-Sommerzeit";
        }
    }
    "Tuvalu";
    {
        "long";
        {
            "standard";
            "Tuvalu-Zeit";
        }
    }
    "Uruguay";
    {
        "long";
        {
            "generic";
            "Uruguayische Zeit",
            "standard";
            "Uruguayische Normalzeit",
            "daylight";
            "Uruguayische Sommerzeit";
        }
    }
}
```

```
}
"Uzbekistan";
{
  "long";
  {
    "generic";
    "Uzbekistan-Zeit",
    "standard";
    "Uzbekistan-Normalzeit",
    "daylight";
    "Uzbekistan-Sommerzeit";
  }
}
"Vanuatu";
{
  "long";
  {
    "generic";
    "Vanuatu-Zeit",
    "standard";
    "Vanuatu-Normalzeit",
    "daylight";
    "Vanuatu-Sommerzeit";
  }
}
"Venezuela";
{
  "long";
  {
    "standard";
    "Venezuela-Zeit";
  }
}
"Vladivostok";
{
  "long";
  {
    "generic";
    "Wladiwostok-Zeit",
    "standard";
    "Wladiwostok-Normalzeit",
    "daylight";
    "Wladiwostok-Sommerzeit";
  }
}
"Volgograd";
{
  "long";
  {
    "generic";
    "Wolgograd-Zeit",
    "standard";
    "Wolgograd-Normalzeit",
    "daylight";
    "Wolgograd-Sommerzeit";
  }
}
}
```

```

        "Vostok";
    {
        "long";
        {
            "standard";
            "Wostok-Zeit";
        }
    }
    "Wake";
    {
        "long";
        {
            "standard";
            "Wake-Insel-Zeit";
        }
    }
    "Wallis";
    {
        "long";
        {
            "standard";
            "Wallis-und-Futuna-Zeit";
        }
    }
    "Yakutsk";
    {
        "long";
        {
            "generic";
            "Jakutsk-Zeit",
            "standard";
            "Jakutsk-Normalzeit",
            "daylight";
            "Jakutsk-Sommerzeit";
        }
    }
    "Yekaterinburg";
    {
        "long";
        {
            "generic";
            "Jekaterinburg-Zeit",
            "standard";
            "Jekaterinburg-Normalzeit",
            "daylight";
            "Jekaterinburg-Sommerzeit";
        }
    }
}
}
}
}
}
}
}
}
}
}

```

Right-To-Left

The DateTimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use `enableRtl` property to set the RTL direction.

The following code example initialize the DateTimePicker component in Arabic culture and also explains how to set the localized text to the placeholder using `load` method of [L10n](#) class.

[Class-component]

CA-GREGORIAN.JSON

```
{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "dates": {
        "calendars": {
          "gregorian": {
            "months": {
              "format": {
                "abbreviated": {
                  "1": "يناير",
                  "2": "فبراير",
                  "3": "مارس",
                  "4": "أبريل",
                  "5": "مايو",
                  "6": "يونيو",
                  "7": "يوليو",
                  "8": "أغسطس",
                  "9": "سبتمبر",
                  "10": "أكتوبر",
                  "11": "نوفمبر",
                  "12": "ديسمبر"
                },
                "narrow": {
                  "1": "ي",
                  "2": "ف",
                  "3": "م",
                  "4": "أ",
                  "5": "و",
                  "6": "ن",
                  "7": "ل",
                  "8": "غ",
                  "9": "س",
                  "10": "ك",
                  "11": "ب",
                  "12": "د"
                },
                "wide": {
                  "1": "يناير",
                  "2": "فبراير",
```

```

        "3": "مارس",
        "4": "أبريل",
        "5": "مايو",
        "6": "يونيو",
        "7": "يوليو",
        "8": "أغسطس",
        "9": "سبتمبر",
        "10": "أكتوبر",
        "11": "نوفمبر",
        "12": "ديسمبر"
    },
    },
    "stand-alone": {
        "abbreviated": {
            "1": "يناير",
            "2": "فبراير",
            "3": "مارس",
            "4": "أبريل",
            "5": "مايو",
            "6": "يونيو",
            "7": "يوليو",
            "8": "أغسطس",
            "9": "سبتمبر",
            "10": "أكتوبر",
            "11": "نوفمبر",
            "12": "ديسمبر"
        },
        "narrow": {
            "1": "ي",
            "2": "ف",
            "3": "م",
            "4": "أ",
            "5": "و",
            "6": "ن",
            "7": "ل",
            "8": "غ",
            "9": "س",
            "10": "ك",
            "11": "ب",
            "12": "د"
        },
        "wide": {
            "1": "يناير",
            "2": "فبراير",
            "3": "مارس",
            "4": "أبريل",
            "5": "مايو",
            "6": "يونيو",
            "7": "يوليو",
            "8": "أغسطس",
            "9": "سبتمبر",
            "10": "أكتوبر",
            "11": "نوفمبر",
            "12": "ديسمبر"
        }
    }
}
},

```

```

"days": {
  "format": {
    "abbreviated": {
      "sun": "الأحد",
      "mon": "الاثنين",
      "tue": "الثلاثاء",
      "wed": "الأربعاء",
      "thu": "الخميس",
      "fri": "الجمعة",
      "sat": "السبت"
    },
    "narrow": {
      "sun": "ح",
      "mon": "ن",
      "tue": "ث",
      "wed": "ر",
      "thu": "خ",
      "fri": "ج",
      "sat": "س"
    },
    "short": {
      "sun": "أحد",
      "mon": "إثنين",
      "tue": "ثلاثاء",
      "wed": "أربعاء",
      "thu": "خميس",
      "fri": "جمعة",
      "sat": "سبت"
    },
    "wide": {
      "sun": "الأحد",
      "mon": "الاثنين",
      "tue": "الثلاثاء",
      "wed": "الأربعاء",
      "thu": "الخميس",
      "fri": "الجمعة",
      "sat": "السبت"
    }
  },
  "stand-alone": {
    "abbreviated": {
      "sun": "الأحد",
      "mon": "الاثنين",
      "tue": "الثلاثاء",
      "wed": "الأربعاء",
      "thu": "الخميس",
      "fri": "الجمعة",
      "sat": "السبت"
    },
    "narrow": {
      "sun": "ح",
      "mon": "ن",
      "tue": "ث",
      "wed": "ر",
      "thu": "خ",
      "fri": "ج",
      "sat": "س"
    }
  }
}

```

```

    },
    "short": {
      "sun": "أحد",
      "mon": "إثنين",
      "tue": "ثلاثاء",
      "wed": "أربعاء",
      "thu": "خميس",
      "fri": "جمعة",
      "sat": "سبت"
    },
    "wide": {
      "sun": "الأحد",
      "mon": "الاثنين",
      "tue": "الثلاثاء",
      "wed": "الأربعاء",
      "thu": "الخميس",
      "fri": "الجمعة",
      "sat": "السبت"
    }
  },
  "quarters": {
    "format": {
      "abbreviated": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      },
      "narrow": {
        "1": "١",
        "2": "٢",
        "3": "٣",
        "4": "٤"
      },
      "wide": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      }
    },
    "stand-alone": {
      "abbreviated": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      },
      "narrow": {
        "1": "١",
        "2": "٢",
        "3": "٣",
        "4": "٤"
      },
      "wide": {
        "1": "الربع الأول",

```

```

        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
    }
}
},
"dayPeriods": {
    "format": {
        "abbreviated": {
            "am": "ص",
            "pm": "م",
            "morning1": "فجرًا",
            "morning2": "ص",
            "afternoon1": "ظهرًا",
            "afternoon2": "بعد الظهر",
            "evening1": "مساءً",
            "night1": "منتصف الليل",
            "night2": "ليلاً"
        },
        "narrow": {
            "am": "ص",
            "pm": "م",
            "morning1": "فجرًا",
            "morning2": "صباحًا",
            "afternoon1": "ظهرًا",
            "afternoon2": "بعد الظهر",
            "evening1": "مساءً",
            "night1": "منتصف الليل",
            "night2": "ليلاً"
        },
        "wide": {
            "am": "ص",
            "pm": "م",
            "morning1": "فجرًا",
            "morning2": "صباحًا",
            "afternoon1": "ظهرًا",
            "afternoon2": "بعد الظهر",
            "evening1": "مساءً",
            "night1": "منتصف الليل",
            "night2": "ليلاً"
        }
    },
    "stand-alone": {
        "abbreviated": {
            "am": "ص",
            "pm": "م",
            "morning1": "فجرًا",
            "morning2": "ص",
            "afternoon1": "ظهرًا",
            "afternoon2": "بعد الظهر",
            "evening1": "مساءً",
            "night1": "منتصف الليل",
            "night2": "ليلاً"
        },
        "narrow": {
            "am": "ص",
            "pm": "م",

```



```

        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    },
    "wide": {
        "am": "صباحًا",
        "pm": "مساءً",
        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    }
},
"eras": {
    "eraNames": {
        "0": "قبل الميلاد",
        "0-alt-variant": "قبل الحقبة الحالية",
        "1": "ميلادي",
        "1-alt-variant": "بعد الميلاد"
    },
    "eraAbbr": {
        "0": "ق.م",
        "0-alt-variant": "ق.م.",
        "1": "م",
        "1-alt-variant": "ب.م"
    },
    "eraNarrow": {
        "0": "ق.م",
        "0-alt-variant": "ق.م.",
        "1": "م",
        "1-alt-variant": "ب.م"
    }
},
"dateFormats": {
    "full": "EEEE، d MMMM y",
    "long": "d MMMM y",
    "medium": "dd/MM/y",
    "short": "d/M/y"
},
"timeFormats": {
    "full": "h:mm:ss a zzzz",
    "long": "h:mm:ss a z",
    "medium": "h:mm:ss a",
    "short": "h:mm a"
},
"dateTimeFormats": {
    "full": "{1} {0}",
    "long": "{1} {0}",
    "medium": "{1} {0}",

```

```

"short": "{1} {0}",
"availableFormats": {
  "Bh": "h B",
  "Bhm": "h:mm B",
  "Bhms": "h:mm:ss B",
  "d": "d",
  "E": "ccc",
  "EBhm": "E h:mm B",
  "EBhms": "E h:mm:ss B",
  "Ed": "E, d",
  "Ehm": "E h:mm a",
  "EHm": "E HH:mm",
  "Ehms": "E h:mm:ss a",
  "EHms": "E HH:mm:ss",
  "Gy": "y G",
  "GyMMM": "MMM y G",
  "GyMMMd": "d MMM y G",
  "GyMMMED": "E, d MMM y G",
  "h": "h a",
  "H": "HH",
  "hm": "h:mm a",
  "Hm": "HH:mm",
  "hms": "h:mm:ss a",
  "Hms": "HH:mm:ss",
  "hmsv": "h:mm:ss a v",
  "Hmsv": "HH:mm:ss v",
  "hmv": "h:mm a v",
  "Hmv": "HH:mm v",
  "M": "L",
  "Md": "d/M",
  "MEd": "E, d/M",
  "MMdd": "dd/MM",
  "MMM": "LLL",
  "MMMd": "d MMM",
  "MMMED": "E, d MMM",
  "MMMMd": "d MMMM",
  "MMMMEd": "E, d MMMM",
  "MMMMMW-count-zero": "W من الاسبوع",
  "MMMMMW-count-one": "W من الاسبوع",
  "MMMMMW-count-two": "W من الاسبوع",
  "MMMMMW-count-few": "W من الاسبوع",
  "MMMMMW-count-many": "W من الاسبوع",
  "MMMMMW-count-other": "W من الاسبوع",
  "ms": "mm:ss",
  "y": "y",
  "yM": "M/y",
  "yMd": "d/M/y",
  "yMEd": "E, d/M/y",
  "yMM": "MM/y",
  "yMMM": "MMM y",
  "yMMMd": "d MMM y",
  "yMMMED": "E, d MMM y",
  "yMMMM": "MMMM y",
  "yQQQ": "QQQ y",
  "yQQQQ": "QQQQ y",
  "yw-count-zero": "Y من سنة W الاسبوع",
  "yw-count-one": "Y من سنة W الاسبوع",

```

```

        "yw-count-two": "Y من سنة w الأسبوع",
        "yw-count-few": "Y من سنة w الأسبوع",
        "yw-count-many": "Y من سنة w الأسبوع",
        "yw-count-other": "Y من سنة w الأسبوع"
    },
    "appendItems": {
        "Day": "{0} ({2}: {1})",
        "Day-Of-Week": "{0} {1}",
        "Era": "{1} {0}",
        "Hour": "{0} ({2}: {1})",
        "Minute": "{0} ({2}: {1})",
        "Month": "{0} ({2}: {1})",
        "Quarter": "{0} ({2}: {1})",
        "Second": "{0} ({2}: {1})",
        "Timezone": "{0} {1}",
        "Week": "{0} ({2}: {1})",
        "Year": "{1} {0}"
    },
    "intervalFormats": {
        "intervalFormatFallback": "{0} - {1}",
        "d": {
            "d": "d-d"
        },
        "h": {
            "a": "h a - h a",
            "h": "h-h a"
        },
        "H": {
            "H": "HH-HH"
        },
        "hm": {
            "a": "h:mm a - h:mm a",
            "h": "h:mm-h:mm a",
            "m": "h:mm-h:mm a"
        },
        "Hm": {
            "H": "HH:mm-HH:mm",
            "m": "HH:mm-HH:mm"
        },
        "hmv": {
            "a": "h:mm a - h:mm a v",
            "h": "h:mm-h:mm a v",
            "m": "h:mm-h:mm a v"
        },
        "Hmv": {
            "H": "HH:mm-HH:mm v",
            "m": "HH:mm-HH:mm v"
        },
        "hv": {
            "a": "h a - h a v",
            "h": "h-h a v"
        },
        "Hv": {
            "H": "HH-HH v"
        },
        "M": {
            "M": "M-M"
        }
    }

```

```

    },
    "Md": {
      "d": "M/d - M/d",
      "M": "M/d - M/d"
    },
    "MEd": {
      "d": "E, d/M - E, d/M",
      "M": "E, d/M - E, d/M"
    },
    "MMM": {
      "M": "MMM-MMM"
    },
    "MMMd": {
      "d": "d-d MMM",
      "M": "d MMM - d MMM"
    },
    "MMMEd": {
      "d": "E, d - E, d MMM",
      "M": "E, d MMM - E, d MMM"
    },
    "MMMM": {
      "M": "LLLL-LLLL"
    },
    "Y": {
      "Y": "Y-Y"
    },
    "YM": {
      "M": "M/Y - M/Y",
      "Y": "M/Y - M/Y"
    },
    "YMd": {
      "d": "d/M/Y - d/M/Y",
      "M": "d/M/Y - d/M/Y",
      "Y": "d/M/Y - d/M/Y"
    },
    "YMEd": {
      "d": "E, dd/MM/Y - E, dd/MM/Y",
      "M": "E, d/M/Y - E, d/M/Y",
      "Y": "E, d/M/Y - E, d/M/Y"
    },
    "YMMM": {
      "M": "MMM - MMM, Y",
      "Y": "MMM, Y - MMM, Y"
    },
    "YMMMd": {
      "d": "d-d MMM, Y",
      "M": "d MMM - d MMM, Y",
      "Y": "d MMM, Y - d MMM, Y"
    },
    "YMMMEd": {
      "d": "E, d - E, d MMM, Y",
      "M": "E, d MMM - E, d MMM, Y",
      "Y": "E, d MMM, Y - E, d MMM, Y"
    },
    "YMMMM": {
      "M": "MMMM - MMMM, Y",
      "Y": "MMMM, Y - MMMM, Y"
    }
  }

```

CA-GREGORIAN.JSX

```
{
  "main";
  {
    "ar";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13686 $",
          "_cldrVersion";
          "32";
        }
        "language";
        "ar";
      }
      "dates";
      {
        "calendars";
        {
          "gregorian";
          {
            "months";
            {
              "format";
              {
                "abbreviated";
                {
                  "1";
                  "يناير",
                  "2";
                  "فبراير",
                  "3";
                  "مارس",
                  "4";
                  "أبريل",
                  "5";
                  "مايو",
                  "6";
                  "يونيو",
                  "7";
                  "يوليو",
                  "8";
```

```

        "أغسطس",
        "9";
        "سبتمبر",
        "10";
        "أكتوبر",
        "11";
        "نوفمبر",
        "12";
        "ديسمبر";
    }
    "narrow";
    {
        "1";
        "ي",
        "2";
        "ف",
        "3";
        "م",
        "4";
        "أ",
        "5";
        "و",
        "6";
        "ن",
        "7";
        "ل",
        "8";
        "غ",
        "9";
        "س",
        "10";
        "ك",
        "11";
        "ب",
        "12";
        "د";
    }
    "wide";
    {
        "1";
        "يناير",
        "2";
        "فبراير",
        "3";
        "مارس",
        "4";
        "أبريل",
        "5";
        "مايو",
        "6";
        "يونيو",
        "7";
        "يوليو",
        "8";
        "أغسطس",
        "9";
        "سبتمبر",

```

```

        "10";
        "أكتوبر",
        "11";
        "نوفمبر",
        "12";
        "ديسمبر";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "يناير",
        "2";
        "فبراير",
        "3";
        "مارس",
        "4";
        "أبريل",
        "5";
        "مايو",
        "6";
        "يونيو",
        "7";
        "يوليو",
        "8";
        "أغسطس",
        "9";
        "سبتمبر",
        "10";
        "أكتوبر",
        "11";
        "نوفمبر",
        "12";
        "ديسمبر";
    }
}
"narrow";
{
    "1";
    "ي",
    "2";
    "ف",
    "3";
    "م",
    "4";
    "أ",
    "5";
    "و",
    "6";
    "ن",
    "7";
    "ل",
    "8";
    "غ",
    "9";
    "س";
}

```

```

        "10";
        "ك",
        "11";
        "ب",
        "12";
        "د";
    }
    "wide";
    {
        "1";
        "يناير",
        "2";
        "فبراير",
        "3";
        "مارس",
        "4";
        "أبريل",
        "5";
        "مايو",
        "6";
        "يونيو",
        "7";
        "يوليو",
        "8";
        "أغسطس",
        "9";
        "سبتمبر",
        "10";
        "أكتوبر",
        "11";
        "نوفمبر",
        "12";
        "ديسمبر";
    }
}
"days";
{
    "format";
    {
        "abbreviated";
        {
            "sun";
            "الأحد",
            "mon";
            "الاثنين",
            "tue";
            "الثلاثاء",
            "wed";
            "الأربعاء",
            "thu";
            "الخميس",
            "fri";
            "الجمعة",
            "sat";
            "السبت";
        }
    }
}

```



```
"narrow";
{
  "sun";
  "ح",
  "mon";
  "ن",
  "tue";
  "ث",
  "wed";
  "ر",
  "thu";
  "خ",
  "fri";
  "ج",
  "sat";
  "س";
}
"short";
{
  "sun";
  "أحد",
  "mon";
  "إثنين",
  "tue";
  "ثلاثاء",
  "wed";
  "أربعاء",
  "thu";
  "خميس",
  "fri";
  "جمعة",
  "sat";
  "سبت";
}
"wide";
{
  "sun";
  "الأحد",
  "mon";
  "الاثنين",
  "tue";
  "الثلاثاء",
  "wed";
  "الأربعاء",
  "thu";
  "الخميس",
  "fri";
  "الجمعة",
  "sat";
  "السبت";
}
}
"stand-alone";
{
  "abbreviated";
  {
    "sun";
```

```

        "الأحد",
        "mon";
        "الاثنين",
        "tue";
        "الثلاثاء",
        "wed";
        "الأربعاء",
        "thu";
        "الخميس",
        "fri";
        "الجمعة",
        "sat";
        "السبت";
    }
    "narrow";
    {
        "sun";
        "ح",
        "mon";
        "ن",
        "tue";
        "ث",
        "wed";
        "ر",
        "thu";
        "خ",
        "fri";
        "ج",
        "sat";
        "س";
    }
    "short";
    {
        "sun";
        "أحد",
        "mon";
        "إثنين",
        "tue";
        "ثلاثاء",
        "wed";
        "أربعاء",
        "thu";
        "خميس",
        "fri";
        "جمعة",
        "sat";
        "سبت";
    }
    "wide";
    {
        "sun";
        "الأحد",
        "mon";
        "الاثنين",
        "tue";
        "الثلاثاء",
        "wed";

```

```

        "الأربعاء",
        "thu";
        "الخميس",
        "fri";
        "الجمعة",
        "sat";
        "السبت";
    }
}
}
"quarters";
{
    "format";
    {
        "abbreviated";
        {
            "1";
            "الربع الأول",
            "2";
            "الربع الثاني",
            "3";
            "الربع الثالث",
            "4";
            "الربع الرابع";
        }
        "narrow";
        {
            "1";
            "١",
            "2";
            "٢",
            "3";
            "٣",
            "4";
            "٤";
        }
        "wide";
        {
            "1";
            "الربع الأول",
            "2";
            "الربع الثاني",
            "3";
            "الربع الثالث",
            "4";
            "الربع الرابع";
        }
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "الربع الأول",
        "2";
        "الربع الثاني",
        "3";
    }
}

```

```

        "الربع الثالث",
        "4";
        "الربع الرابع";
    }
    "narrow";
    {
        "1";
        "١",
        "2";
        "٢",
        "3";
        "٣",
        "4";
        "٤";
    }
    "wide";
    {
        "1";
        "الربع الأول",
        "2";
        "الربع الثاني",
        "3";
        "الربع الثالث",
        "4";
        "الربع الرابع";
    }
    }
    "dayPeriods";
    {
        "format";
        {
            "abbreviated";
            {
                "am";
                "ص",
                "pm";
                "م",
                "morning1";
                "فجرًا",
                "morning2";
                "ص",
                "afternoon1";
                "ظهرًا",
                "afternoon2";
                "بعد الظهر",
                "evening1";
                "مساءً",
                "night1";
                "منتصف الليل",
                "night2";
                "ليلاً";
            }
            "narrow";
            {
                "am";
                "ص",

```

```

        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
    "wide";
    {
        "am";
        "ص",
        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "am";
        "ص",
        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "ص",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",

```

```

        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
    "narrow";
    {
        "am";
        "ص",
        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
    "wide";
    {
        "am";
        "صباحًا",
        "pm";
        "مساءً",
        "morning1";
        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
}
"eras";
{
    "eraNames";
    {
        "0";
        "قبل الميلاد",
        "0-alt-variant";
        "قبل الحقبة الحالية",

```

```

        "1";
        "ميلادي",
        "1-alt-variant";
        "بعد الميلاد";
    }
    "eraAbbr";
    {
        "0";
        "ق.م",
        "0-alt-variant";
        "ق.م",
        "1";
        "م",
        "1-alt-variant";
        "ب.م";
    }
    "eraNarrow";
    {
        "0";
        "ق.م",
        "0-alt-variant";
        "ق.م",
        "1";
        "م",
        "1-alt-variant";
        "ب.م";
    }
}
"dateFormats";
{
    "full";
    "EEEE, d MMMM y",
    "long";
    "d MMMM y",
    "medium";
    "dd/MM/y",
    "short";
    "d/M/y";
}
"timeFormats";
{
    "full";
    "h:mm:ss a zzzz",
    "long";
    "h:mm:ss a z",
    "medium";
    "h:mm:ss a",
    "short";
    "h:mm a";
}
"dateTimeFormats";
{
    "full";
    "{1} {0}",
    "long";
    "{1} {0}",
    "medium";

```

```

    "{1} {0}",
    "short";
    "{1} {0}",
    "availableFormats";
    {
        "Bh";
        "h B",
            "Bhm";
        "h:mm B",
            "Bhms";
        "h:mm:ss B",
            "d";
        "d",
            "E";
        "ccc",
            "EBhm";
        "E h:mm B",
            "EBhms";
        "E h:mm:ss B",
            "Ed";
        "E, d",
            "Ehm";
        "E h:mm a",
            "EHm";
        "E HH:mm",
            "Ehms";
        "E h:mm:ss a",
            "EHms";
        "E HH:mm:ss",
            "Gy";
        "y G",
            "GyMMM";
        "MMM y G",
            "GyMMMd";
        "d MMM y G",
            "GyMMMED";
        "E, d MMM y G",
            "h";
        "h a",
            "H";
        "HH",
            "hm";
        "h:mm a",
            "Hm";
        "HH:mm",
            "hms";
        "h:mm:ss a",
            "Hms";
        "HH:mm:ss",
            "hmsv";
        "h:mm:ss a v",
            "Hmsv";
        "HH:mm:ss v",
            "hmv";
        "h:mm a v",
            "Hmv";
        "HH:mm v",

```



```

        "M";
    "L",
        "Md";
    "d/M",
        "MEd";
    "E، d/M",
        "MMdd";
    "dd/MM",
        "MMM";
    "LLL",
        "MMMd";
    "d MMM",
        "MMMEd";
    "E، d MMM",
        "MMMMd";
    "d MMMM",
        "MMMMEd";
    "E، d MMMM",
        "MMMMW-count-zero";
    "من الاسبوع W MMM",
        "MMMMW-count-one";
    "من الاسبوع W MMM",
        "MMMMW-count-two";
    "من الاسبوع W MMM",
        "MMMMW-count-few";
    "من الاسبوع W MMM",
        "MMMMW-count-many";
    "من الاسبوع W MMM",
        "MMMMW-count-other";
    "من الاسبوع W MMM",
        "ms";
    "mm:ss",
        "y";
    "y",
        "yM";
    "M/y",
        "yMd";
    "d/M/y",
        "yMEd";
    "E، d/M/y",
        "yMM";
    "MM/y",
        "yMMM";
    "MMM y",
        "yMMMd";
    "d MMM y",
        "yMMMEd";
    "E، d MMM y",
        "yMMMM";
    "MMMM y",
        "yQQQ";
    "QQQ y",
        "yQQQQ";
    "QQQQ y",
        "yw-count-zero";
    "Y من سنة w الاسبوع",
        "yw-count-one";

```

```

        "Y من سنة w الأسبوع",
        "yw-count-two";
        "Y من سنة w الأسبوع",
        "yw-count-few";
        "Y من سنة w الأسبوع",
        "yw-count-many";
        "Y من سنة w الأسبوع",
        "yw-count-other";
        "Y من سنة w الأسبوع";
    }
    "appendItems";
    {
        "Day";
        "{0} ({2}: {1})",
        "Day-Of-Week";
        "{0} {1}",
        "Era";
        "{1} {0}",
        "Hour";
        "{0} ({2}: {1})",
        "Minute";
        "{0} ({2}: {1})",
        "Month";
        "{0} ({2}: {1})",
        "Quarter";
        "{0} ({2}: {1})",
        "Second";
        "{0} ({2}: {1})",
        "Timezone";
        "{0} {1}",
        "Week";
        "{0} ({2}: {1})",
        "Year";
        "{1} {0}";
    }
    "intervalFormats";
    {
        "intervalFormatFallback";
        "{0} - {1}",
        "d";
        {
            "d";
            "d-d";
        }
        "h";
        {
            "a";
            "h a - h a",
            "h";
            "h-h a";
        }
        "H";
        {
            "H";
            "HH-HH";
        }
        "hm";
    }

```

```
{
    "a";
    "h:mm a - h:mm a",
    "h";
    "h:mm-h:mm a",
    "m";
    "h:mm-h:mm a";
}
"Hm";
{
    "H";
    "HH:mm-HH:mm",
    "m";
    "HH:mm-HH:mm";
}
"Hmv";
{
    "a";
    "h:mm a - h:mm a v",
    "h";
    "h:mm-h:mm a v",
    "m";
    "h:mm-h:mm a v";
}
"Hmv";
{
    "H";
    "HH:mm-HH:mm v",
    "m";
    "HH:mm-HH:mm v";
}
"hv";
{
    "a";
    "h a - h a v",
    "h";
    "h-h a v";
}
"Hv";
{
    "H";
    "HH-HH v";
}
"M";
{
    "M";
    "M-M";
}
"Md";
{
    "d";
    "M/d - M/d",
    "M";
    "M/d - M/d";
}
"MEd";
{
```

```

        "d";
        "E, d/M - E, d/M",
        "M";
        "E, d/M - E, d/M";
    }
    "MMM";
    {
        "M";
        "MMM-MMM";
    }
    "MMMd";
    {
        "d";
        "d-d MMM",
        "M";
        "d MMM - d MMM";
    }
    "MMMEd";
    {
        "d";
        "E, d - E, d MMM",
        "M";
        "E, d MMM - E, d MMM";
    }
    "MMMM";
    {
        "M";
        "LLLL-LLLL";
    }
    "Y";
    {
        "Y";
        "Y-Y";
    }
    "yM";
    {
        "M";
        "M/Y - M/Y",
        "Y";
        "M/Y - M/Y";
    }
    "yMd";
    {
        "d";
        "d/M/Y - d/M/Y",
        "M";
        "d/M/Y - d/M/Y",
        "Y";
        "d/M/Y - d/M/Y";
    }
    "yMEd";
    {
        "d";
        "E, dd/MM/Y - E, dd/MM/Y",
        "M";
        "E, d/M/Y - E, d/M/Y",
        "Y";
    }

```

```

    "E. d/M/y - E. d/M/y";
}
"yMMM";
{
    "M";
    "MMM - MMM. y",
        "y";
    "MMM. y - MMM. y";
}
"yMMMd";
{
    "d";
    "d-d MMM. y",
        "M";
    "d MMM - d MMM. y",
        "y";
    "d MMM. y - d MMM. y";
}
"yMMMED";
{
    "d";
    "E. d - E. d MMM. y",
        "M";
    "E. d MMM - E. d MMM. y",
        "Y";
    "E. d MMM. y - E. d MMM. y";
}
"yMMMM";
{
    "M";
    "MMMM - MMMM. y",
        "Y";
    "MMMM. y - MMMM. y";
}
}
}
}
}
}
}
}
}
}

```

CURRENCIES.JSON

```
{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "numbers": {
```

```

"currencies": {
  "ADP": {
    "displayName": "بيستا أندوري",
    "symbol": "ADP"
  },
  "AED": {
    "displayName": "درهم إماراتي",
    "displayName-count-zero": "درهم إماراتي",
    "displayName-count-one": "درهم إماراتي",
    "displayName-count-two": "درهم إماراتي",
    "displayName-count-few": "درهم إماراتي",
    "displayName-count-many": "درهم إماراتي",
    "displayName-count-other": "درهم إماراتي",
    "symbol": "د.إ."
  },
  "AFA": {
    "displayName": "أفغاني - 2002-1927",
    "symbol": "AFA"
  },
  "AFN": {
    "displayName": "أفغاني",
    "displayName-count-zero": "أفغاني أفغانستان",
    "displayName-count-one": "أفغاني أفغانستان",
    "displayName-count-two": "أفغاني أفغانستان",
    "displayName-count-few": "أفغاني أفغانستان",
    "displayName-count-many": "أفغاني أفغانستان",
    "displayName-count-other": "أفغاني أفغانستان",
    "symbol": "AFN"
  },
  "ALK": {
    "displayName": "ALK",
    "symbol": "ALK"
  },
  "ALL": {
    "displayName": "ليك ألباني",
    "displayName-count-zero": "ليك ألباني",
    "displayName-count-one": "ليك ألباني",
    "displayName-count-two": "ليك ألباني",
    "displayName-count-few": "ليك ألباني",
    "displayName-count-many": "ليك ألباني",
    "displayName-count-other": "ليك ألباني",
    "symbol": "ALL"
  },
  "AMD": {
    "displayName": "درام أرميني",
    "displayName-count-zero": "درام أرميني",
    "displayName-count-one": "درام أرميني",
    "displayName-count-two": "درام أرميني",
    "displayName-count-few": "درام أرميني",
    "displayName-count-many": "درام أرميني",
    "displayName-count-other": "درام أرميني",
    "symbol": "AMD"
  },
  "ANG": {
    "displayName": "غيلدر أنتيلي هولندي",
    "displayName-count-zero": "غيلدر أنتيلي هولندي",
    "displayName-count-one": "غيلدر أنتيلي هولندي",

```

```

    "displayName-count-two": "غیلدر أنتیلی هولندي",
    "displayName-count-few": "غیلدر أنتیلی هولندي",
    "displayName-count-many": "غیلدر أنتیلی هولندي",
    "displayName-count-other": "غیلدر أنتیلی هولندي",
    "symbol": "ANG"
  },
  "AOA": {
    "displayName": "کوانزا أنغولي",
    "displayName-count-zero": "کوانزا أنغولي",
    "displayName-count-one": "کوانزا أنغولي",
    "displayName-count-two": "کوانزا أنغولي",
    "displayName-count-few": "کوانزا أنغولي",
    "displayName-count-many": "کوانزا أنغولي",
    "displayName-count-other": "کوانزا أنغولي",
    "symbol": "AOA",
    "symbol-alt-narrow": "Kz"
  },
  "AOK": {
    "displayName": "کوانزا أنجولي - 1990-1977",
    "symbol": "AOK"
  },
  "AON": {
    "displayName": "کوانزا أنجولي جديدة - 2000-1990",
    "symbol": "AON"
  },
  "AOR": {
    "displayName": "کوانزا أنجولي معدلة - 1999 - 1995",
    "symbol": "AOR"
  },
  "ARA": {
    "displayName": "استرال أرجنتيني",
    "symbol": "ARA"
  },
  "ARL": {
    "displayName": "ARL",
    "symbol": "ARL"
  },
  "ARM": {
    "displayName": "ARM",
    "symbol": "ARM"
  },
  "ARP": {
    "displayName": "بیزو أرجنتيني - 1985-1983",
    "symbol": "ARP"
  },
  "ARS": {
    "displayName": "بیزو أرجنتيني",
    "displayName-count-zero": "بیزو أرجنتيني",
    "displayName-count-one": "بیزو أرجنتيني",
    "displayName-count-two": "بیزو أرجنتيني",
    "displayName-count-few": "بیزو أرجنتيني",
    "displayName-count-many": "بیزو أرجنتيني",
    "displayName-count-other": "بیزو أرجنتيني",
    "symbol": "ARS",
    "symbol-alt-narrow": "AR$"
  },
  "ATS": {

```

```

    "displayName": "شلن نمساوي",
    "symbol": "ATS"
  },
  "AUD": {
    "displayName": "دولار أسترالي",
    "displayName-count-zero": "دولار أسترالي",
    "displayName-count-one": "دولار أسترالي",
    "displayName-count-two": "دولار أسترالي",
    "displayName-count-few": "دولار أسترالي",
    "displayName-count-many": "دولار أسترالي",
    "displayName-count-other": "دولار أسترالي",
    "symbol": "AU$",
    "symbol-alt-narrow": "AU$"
  },
  "AWG": {
    "displayName": "فلورن أروبي",
    "displayName-count-zero": "فلورن أروبي",
    "displayName-count-one": "فلورن أروبي",
    "displayName-count-two": "فلورن أروبي",
    "displayName-count-few": "فلورن أروبي",
    "displayName-count-many": "فلورن أروبي",
    "displayName-count-other": "فلورن أروبي",
    "symbol": "AWG"
  },
  "AZM": {
    "displayName": "مانات أذربيجاني",
    "symbol": "AZM"
  },
  "AZN": {
    "displayName": "مانات أذربيجان",
    "displayName-count-zero": "مانت أذربيجاني",
    "displayName-count-one": "مانت أذربيجاني",
    "displayName-count-two": "مانت أذربيجاني",
    "displayName-count-few": "مانت أذربيجاني",
    "displayName-count-many": "مانت أذربيجاني",
    "displayName-count-other": "مانت أذربيجاني",
    "symbol": "AZN"
  },
  "BAD": {
    "displayName": "دينار البوسنة والهرسك",
    "symbol": "BAD"
  },
  "BAM": {
    "displayName": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-zero": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-one": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-two": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-few": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-many": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-other": "مارك البوسنة والهرسك قابل للتحويل",
    "symbol": "BAM",
    "symbol-alt-narrow": "KM"
  },
  "BAN": {
    "displayName": "BAN",
    "symbol": "BAN"
  },

```



```

"BBD": {
  "displayName": "دولار بربادوسي",
  "displayName-count-zero": "دولار بربادوسي",
  "displayName-count-one": "دولار بربادوسي",
  "displayName-count-two": "دولار بربادوسي",
  "displayName-count-few": "دولار بربادوسي",
  "displayName-count-many": "دولار بربادوسي",
  "displayName-count-other": "دولار بربادوسي",
  "symbol": "BBD",
  "symbol-alt-narrow": "BB$"
},
"BDT": {
  "displayName": "تাকা بنغلاديشي",
  "displayName-count-zero": "تাকা بنغلاديشي",
  "displayName-count-one": "تাকা بنغلاديشي",
  "displayName-count-two": "تাকা بنغلاديشي",
  "displayName-count-few": "تাকা بنغلاديشي",
  "displayName-count-many": "تাকা بنغلاديشي",
  "displayName-count-other": "تাকা بنغلاديشي",
  "symbol": "BDT",
  "symbol-alt-narrow": "₳"
},
"BEC": {
  "displayName": "فرنك بلجيكي قابل للتحويل",
  "symbol": "BEC"
},
"BEF": {
  "displayName": "فرنك بلجيكي",
  "symbol": "BEF"
},
"BEL": {
  "displayName": "فرنك بلجيكي مالي",
  "symbol": "BEL"
},
"BGL": {
  "displayName": "BGL",
  "symbol": "BGL"
},
"BGM": {
  "displayName": "BGM",
  "symbol": "BGM"
},
"BGN": {
  "displayName": "ليف بلغاري",
  "displayName-count-zero": "ليف بلغاري",
  "displayName-count-one": "ليف بلغاري",
  "displayName-count-two": "ليف بلغاري",
  "displayName-count-few": "ليف بلغاري",
  "displayName-count-many": "ليف بلغاري",
  "displayName-count-other": "ليف بلغاري",
  "symbol": "BGN"
},
"BGO": {
  "displayName": "BGO",
  "symbol": "BGO"
},

```

```

"BHD": {
  "displayName": "دينار بحريني",
  "displayName-count-zero": "دينار بحريني",
  "displayName-count-one": "دينار بحريني",
  "displayName-count-two": "دينار بحريني",
  "displayName-count-few": "دينار بحريني",
  "displayName-count-many": "دينار بحريني",
  "displayName-count-other": "دينار بحريني",
  "symbol": "د.ب."
},
"BIF": {
  "displayName": "فرنك بروندي",
  "displayName-count-zero": "فرنك بروندي",
  "displayName-count-one": "فرنك بروندي",
  "displayName-count-two": "فرنك بروندي",
  "displayName-count-few": "فرنك بروندي",
  "displayName-count-many": "فرنك بروندي",
  "displayName-count-other": "فرنك بروندي",
  "symbol": "BIF"
},
"BMD": {
  "displayName": "دولار برمودي",
  "displayName-count-zero": "دولار برمودي",
  "displayName-count-one": "دولار برمودي",
  "displayName-count-two": "دولار برمودي",
  "displayName-count-few": "دولار برمودي",
  "displayName-count-many": "دولار برمودي",
  "displayName-count-other": "دولار برمودي",
  "symbol": "BMD",
  "symbol-alt-narrow": "BM$"
},
"BND": {
  "displayName": "دولار بروناي",
  "displayName-count-zero": "دولار بروناي",
  "displayName-count-one": "دولار بروناي",
  "displayName-count-two": "دولار بروناي",
  "displayName-count-few": "دولار بروناي",
  "displayName-count-many": "دولار بروناي",
  "displayName-count-other": "دولار بروناي",
  "symbol": "BND",
  "symbol-alt-narrow": "BN$"
},
"BOB": {
  "displayName": "بوليفيانو بوليفي",
  "displayName-count-zero": "بوليفيانو بوليفي",
  "displayName-count-one": "بوليفيانو بوليفي",
  "displayName-count-two": "بوليفيانو بوليفي",
  "displayName-count-few": "بوليفيانو بوليفي",
  "displayName-count-many": "بوليفيانو بوليفي",
  "displayName-count-other": "بوليفيانو بوليفي",
  "symbol": "BOB",
  "symbol-alt-narrow": "Bs"
},
"BOL": {
  "displayName": "BOL",
  "symbol": "BOL"
},

```

```

"BOP": {
  "displayName": "بیزو بولیفي",
  "symbol": "BOP"
},
"BOV": {
  "displayName": "مفدول بولیفي",
  "symbol": "BOV"
},
"BRB": {
  "displayName": "نوفو کروزایرو برازیلی - 1986-1967",
  "symbol": "BRB"
},
"BRC": {
  "displayName": "کروزادو برازیلی",
  "symbol": "BRC"
},
"BRE": {
  "displayName": "کروزایرو برازیلی - 1993-1990",
  "symbol": "BRE"
},
"BRL": {
  "displayName": "ریال برازیلی",
  "displayName-count-zero": "ریال برازیلی",
  "displayName-count-one": "ریال برازیلی",
  "displayName-count-two": "ریال برازیلی",
  "displayName-count-few": "ریال برازیلی",
  "displayName-count-many": "ریال برازیلی",
  "displayName-count-other": "ریال برازیلی",
  "symbol": "R$",
  "symbol-alt-narrow": "R$"
},
"BRN": {
  "displayName": "BRN",
  "symbol": "BRN"
},
"BRR": {
  "displayName": "BRR",
  "symbol": "BRR"
},
"BRZ": {
  "displayName": "BRZ",
  "symbol": "BRZ"
},
"BSD": {
  "displayName": "دولار باهامی",
  "displayName-count-zero": "دولار باهامی",
  "displayName-count-one": "دولار باهامی",
  "displayName-count-two": "دولار باهامی",
  "displayName-count-few": "دولار باهامی",
  "displayName-count-many": "دولار باهامی",
  "displayName-count-other": "دولار باهامی",
  "symbol": "BSD",
  "symbol-alt-narrow": "BS$"
},
"BTN": {
  "displayName": "نولتوم بوتانی",
  "displayName-count-zero": "نولتوم بوتانی",

```

```

    "displayName-count-one": "نولتوم بوتاني",
    "displayName-count-two": "نولتوم بوتاني",
    "displayName-count-few": "نولتوم بوتاني",
    "displayName-count-many": "نولتوم بوتاني",
    "displayName-count-other": "نولتوم بوتاني",
    "symbol": "BTN"
  },
  "BUK": {
    "displayName": "کیات بورمی",
    "symbol": "BUK"
  },
  "BWP": {
    "displayName": "بولا بتسوani",
    "displayName-count-zero": "بولا بتسوani",
    "displayName-count-one": "بولا بتسوani",
    "displayName-count-two": "بولا بتسوani",
    "displayName-count-few": "بولا بتسوani",
    "displayName-count-many": "بولا بتسوani",
    "displayName-count-other": "بولا بتسوani",
    "symbol": "BWP",
    "symbol-alt-narrow": "P"
  },
  "BYB": {
    "displayName": "روبل بیلاروسی جدید - 1994-1999",
    "symbol": "BYB"
  },
  "BYN": {
    "displayName": "روبل بیلاروسی",
    "displayName-count-zero": "روبل بیلاروسی",
    "displayName-count-one": "روبل بیلاروسی",
    "displayName-count-two": "روبل بیلاروسی",
    "displayName-count-few": "روبل بیلاروسی",
    "displayName-count-many": "روبل بیلاروسی",
    "displayName-count-other": "روبل بیلاروسی",
    "symbol": "BYN",
    "symbol-alt-narrow": "p."
  },
  "BYR": {
    "displayName": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-zero": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-one": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-two": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-few": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-many": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "displayName-count-other": "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",
    "symbol": "BYR"
  },
  "BZD": {
    "displayName": "دولار بلیزی",
    "displayName-count-zero": "دولار بلیزی",
    "displayName-count-one": "دولار بلیزی",
    "displayName-count-two": "دولاران بلیزیان",
    "displayName-count-few": "دولار بلیزی",
    "displayName-count-many": "دولار بلیزی",
    "displayName-count-other": "دولار بلیزی",
    "symbol": "BZD",
    "symbol-alt-narrow": "BZ$"
  }

```

```

    },
    "CAD": {
      "displayName": "دولار كندي",
      "displayName-count-zero": "دولار كندي",
      "displayName-count-one": "دولار كندي",
      "displayName-count-two": "دولار كندي",
      "displayName-count-few": "دولار كندي",
      "displayName-count-many": "دولار كندي",
      "displayName-count-other": "دولار كندي",
      "symbol": "CA$",
      "symbol-alt-narrow": "CA$"
    },
    "CDF": {
      "displayName": "فرنك كونغولي",
      "displayName-count-zero": "فرنك كونغولي",
      "displayName-count-one": "فرنك كونغولي",
      "displayName-count-two": "فرنك كونغولي",
      "displayName-count-few": "فرنك كونغولي",
      "displayName-count-many": "فرنك كونغولي",
      "displayName-count-other": "فرنك كونغولي",
      "symbol": "CDF"
    },
    "CHE": {
      "displayName": "CHE",
      "symbol": "CHE"
    },
    "CHF": {
      "displayName": "فرنك سويسري",
      "displayName-count-zero": "فرنك سويسري",
      "displayName-count-one": "فرنك سويسري",
      "displayName-count-two": "فرنك سويسري",
      "displayName-count-few": "فرنك سويسري",
      "displayName-count-many": "فرنك سويسري",
      "displayName-count-other": "فرنك سويسري",
      "symbol": "CHF"
    },
    "CHW": {
      "displayName": "CHW",
      "symbol": "CHW"
    },
    "CLE": {
      "displayName": "CLE",
      "symbol": "CLE"
    },
    "CLF": {
      "displayName": "CLF",
      "symbol": "CLF"
    },
    "CLP": {
      "displayName": "بيزو تشيلي",
      "displayName-count-zero": "بيزو تشيلي",
      "displayName-count-one": "بيزو تشيلي",
      "displayName-count-two": "بيزو تشيلي",
      "displayName-count-few": "بيزو تشيلي",
      "displayName-count-many": "بيزو تشيلي",
      "displayName-count-other": "بيزو تشيلي",
      "symbol": "CLP",

```

```

    "symbol-alt-narrow": "CL$"
  },
  "CNH": {
    "displayName": "يوان صيني (في الخارج)",
    "displayName-count-zero": "يوان صيني (في الخارج)",
    "displayName-count-one": "يوان صيني (في الخارج)",
    "displayName-count-two": "يوان صيني (في الخارج)",
    "displayName-count-few": "يوان صيني (في الخارج)",
    "displayName-count-many": "يوان صيني (في الخارج)",
    "displayName-count-other": "يوان صيني (في الخارج)",
    "symbol": "CNH"
  },
  "CNX": {
    "displayName": "CNX",
    "symbol": "CNX"
  },
  "CNY": {
    "displayName": "يوان صيني",
    "displayName-count-zero": "يوان صيني",
    "displayName-count-one": "يوان صيني",
    "displayName-count-two": "يوان صيني",
    "displayName-count-few": "يوان صيني",
    "displayName-count-many": "يوان صيني",
    "displayName-count-other": "يوان صيني",
    "symbol": "CN¥",
    "symbol-alt-narrow": "CN¥"
  },
  "COP": {
    "displayName": "بيزو كولومبي",
    "displayName-count-zero": "بيزو كولومبي",
    "displayName-count-one": "بيزو كولومبي",
    "displayName-count-two": "بيزو كولومبي",
    "displayName-count-few": "بيزو كولومبي",
    "displayName-count-many": "بيزو كولومبي",
    "displayName-count-other": "بيزو كولومبي",
    "symbol": "COP",
    "symbol-alt-narrow": "CO$"
  },
  "COU": {
    "displayName": "COU",
    "symbol": "COU"
  },
  "CRC": {
    "displayName": "كولن كوستاريكي",
    "displayName-count-zero": "كولن كوستاريكي",
    "displayName-count-one": "كولن كوستاريكي",
    "displayName-count-two": "كولن كوستاريكي",
    "displayName-count-few": "كولن كوستاريكي",
    "displayName-count-many": "كولن كوستاريكي",
    "displayName-count-other": "كولن كوستاريكي",
    "symbol": "CRC",
    "symbol-alt-narrow": "₡"
  },
  "CSD": {
    "displayName": "دينار صربي قديم",
    "symbol": "CSD"
  },

```

```

"CSK": {
  "displayName": "كرونة تشيكوسلوفاكيا",
  "symbol": "CSK"
},
"CUC": {
  "displayName": "بيزو كوبي قابل للتحويل",
  "displayName-count-zero": "بيزو كوبي قابل للتحويل",
  "displayName-count-one": "بيزو كوبي قابل للتحويل",
  "displayName-count-two": "بيزو كوبي قابل للتحويل",
  "displayName-count-few": "بيزو كوبي قابل للتحويل",
  "displayName-count-many": "بيزو كوبي قابل للتحويل",
  "displayName-count-other": "بيزو كوبي قابل للتحويل",
  "symbol": "CUC",
  "symbol-alt-narrow": "$"
},
"CUP": {
  "displayName": "بيزو كوبي",
  "displayName-count-zero": "بيزو كوبي",
  "displayName-count-one": "بيزو كوبي",
  "displayName-count-two": "بيزو كوبي",
  "displayName-count-few": "بيزو كوبي",
  "displayName-count-many": "بيزو كوبي",
  "displayName-count-other": "بيزو كوبي",
  "symbol": "CUP",
  "symbol-alt-narrow": "CU$"
},
"CVE": {
  "displayName": "اسكودو الرأس الخضراء",
  "displayName-count-zero": "اسكودو الرأس الخضراء",
  "displayName-count-one": "اسكودو الرأس الخضراء",
  "displayName-count-two": "اسكودو الرأس الخضراء",
  "displayName-count-few": "اسكودو الرأس الخضراء",
  "displayName-count-many": "اسكودو الرأس الخضراء",
  "displayName-count-other": "اسكودو الرأس الخضراء",
  "symbol": "CVE"
},
"CYP": {
  "displayName": "جنيه قبرصي",
  "symbol": "CYP"
},
"CZK": {
  "displayName": "كرونة تشيكية",
  "displayName-count-zero": "كرونة تشيكية",
  "displayName-count-one": "كرونة تشيكية",
  "displayName-count-two": "كرونة تشيكية",
  "displayName-count-few": "كرونة تشيكية",
  "displayName-count-many": "كرونة تشيكية",
  "displayName-count-other": "كرونة تشيكية",
  "symbol": "CZK",
  "symbol-alt-narrow": "Kč"
},
"DDM": {
  "displayName": "أوستمارك ألماني شرقي",
  "symbol": "DDM"
},
"DEM": {
  "displayName": "مارك ألماني",

```

```

    "symbol": "DEM"
  },
  "DJF": {
    "displayName": "فرنك جيبوتي",
    "displayName-count-zero": "فرنك جيبوتي",
    "displayName-count-one": "فرنك جيبوتي",
    "displayName-count-two": "فرنك جيبوتي",
    "displayName-count-few": "فرنك جيبوتي",
    "displayName-count-many": "فرنك جيبوتي",
    "displayName-count-other": "فرنك جيبوتي",
    "symbol": "DJF"
  },
  "DKK": {
    "displayName": "كرونة دنماركية",
    "displayName-count-zero": "كرونة دنماركية",
    "displayName-count-one": "كرونة دنماركية",
    "displayName-count-two": "كرونة دنماركية",
    "displayName-count-few": "كرونة دنماركية",
    "displayName-count-many": "كرونة دنماركية",
    "displayName-count-other": "كرونة دنماركية",
    "symbol": "DKK",
    "symbol-alt-narrow": "kr"
  },
  "DOP": {
    "displayName": "بيزو الدومنيكان",
    "displayName-count-zero": "بيزو الدومنيكان",
    "displayName-count-one": "بيزو الدومنيكان",
    "displayName-count-two": "بيزو الدومنيكان",
    "displayName-count-few": "بيزو الدومنيكان",
    "displayName-count-many": "بيزو الدومنيكان",
    "displayName-count-other": "بيزو الدومنيكان",
    "symbol": "DOP",
    "symbol-alt-narrow": "DO$"
  },
  "DZD": {
    "displayName": "دينار جزائري",
    "displayName-count-zero": "دينار جزائري",
    "displayName-count-one": "دينار جزائري",
    "displayName-count-two": "ديناران جزائريان",
    "displayName-count-few": "دينارات جزائرية",
    "displayName-count-many": "دينارًا جزائريًا",
    "displayName-count-other": "دينار جزائري",
    "symbol": "د.ج."
  },
  "ECS": {
    "displayName": "ECS",
    "symbol": "ECS"
  },
  "ECV": {
    "displayName": "ECV",
    "symbol": "ECV"
  },
  "EEK": {
    "displayName": "كرونة استونية",
    "symbol": "EEK"
  },
  "EGP": {

```



```

    "displayName": "جنيه مصري",
    "displayName-count-zero": "جنيه مصري",
    "displayName-count-one": "جنيه مصري",
    "displayName-count-two": "جنيهان مصريان",
    "displayName-count-few": "جنيهات مصرية",
    "displayName-count-many": "جنيهًا مصريًا",
    "displayName-count-other": "جنيه مصري",
    "symbol": "ج.م.",
    "symbol-alt-narrow": "££"
  },
  "ERN": {
    "displayName": "ناكفا أريتري",
    "displayName-count-zero": "ناكفا أريتري",
    "displayName-count-one": "ناكفا أريتري",
    "displayName-count-two": "ناكفا أريتري",
    "displayName-count-few": "ناكفا أريتري",
    "displayName-count-many": "ناكفا أريتري",
    "displayName-count-other": "ناكفا أريتري",
    "symbol": "ERN"
  },
  "ESA": {
    "displayName": "ESA",
    "symbol": "ESA"
  },
  "ESB": {
    "displayName": "ESB",
    "symbol": "ESB"
  },
  "ESP": {
    "displayName": "بيزيتا إسباني",
    "symbol": "ESP",
    "symbol-alt-narrow": "₧"
  },
  "ETB": {
    "displayName": "بئر أثيوبي",
    "displayName-count-zero": "بئر أثيوبي",
    "displayName-count-one": "بئر أثيوبي",
    "displayName-count-two": "بئر أثيوبي",
    "displayName-count-few": "بئر أثيوبي",
    "displayName-count-many": "بئر أثيوبي",
    "displayName-count-other": "بئر أثيوبي",
    "symbol": "ETB"
  },
  "EUR": {
    "displayName": "يورو",
    "displayName-count-zero": "يورو",
    "displayName-count-one": "يورو",
    "displayName-count-two": "يورو",
    "displayName-count-few": "يورو",
    "displayName-count-many": "يورو",
    "displayName-count-other": "يورو",
    "symbol": "€",
    "symbol-alt-narrow": "€"
  },
  "FIM": {
    "displayName": "ماركا فنلندي",
    "symbol": "FIM"
  }

```

```

    },
    "FJD": {
      "displayName": "دولار فيجي",
      "displayName-count-zero": "دولار فيجي",
      "displayName-count-one": "دولار فيجي",
      "displayName-count-two": "دولار فيجي",
      "displayName-count-few": "دولار فيجي",
      "displayName-count-many": "دولار فيجي",
      "displayName-count-other": "دولار فيجي",
      "symbol": "FJD",
      "symbol-alt-narrow": "FJ$"
    },
    "FKP": {
      "displayName": "جنيه جزر فوكلاند",
      "displayName-count-zero": "جنيه جزر فوكلاند",
      "displayName-count-one": "جنيه جزر فوكلاند",
      "displayName-count-two": "جنيه جزر فوكلاند",
      "displayName-count-few": "جنيه جزر فوكلاند",
      "displayName-count-many": "جنيه جزر فوكلاند",
      "displayName-count-other": "جنيه جزر فوكلاند",
      "symbol": "FKP",
      "symbol-alt-narrow": "£"
    },
    "FRF": {
      "displayName": "فرنك فرنسي",
      "symbol": "FRF"
    },
    "GBP": {
      "displayName": "جنيه إسترليني",
      "displayName-count-zero": "جنيه إسترليني",
      "displayName-count-one": "جنيه إسترليني",
      "displayName-count-two": "جنيه إسترليني",
      "displayName-count-few": "جنيه إسترليني",
      "displayName-count-many": "جنيه إسترليني",
      "displayName-count-other": "جنيه إسترليني",
      "symbol": "£",
      "symbol-alt-narrow": "UK£"
    },
    "GEK": {
      "displayName": "GEK",
      "symbol": "GEK"
    },
    "GEL": {
      "displayName": "لاري جورجى",
      "displayName-count-zero": "لاري جورجى",
      "displayName-count-one": "لاري جورجى",
      "displayName-count-two": "لاري جورجى",
      "displayName-count-few": "لاري جورجى",
      "displayName-count-many": "لاري جورجى",
      "displayName-count-other": "لاري جورجى",
      "symbol": "GEL",
      "symbol-alt-narrow": "ლ",
      "symbol-alt-variant": "₾"
    },
    "GHC": {
      "displayName": "سيدي غاني",
      "symbol": "GHC"
    }
  }

```

```

    },
    "GHS": {
      "displayName": "سيدي غانا",
      "displayName-count-zero": "سيدي غانا",
      "displayName-count-one": "سيدي غانا",
      "displayName-count-two": "سيدي غانا",
      "displayName-count-few": "سيدي غانا",
      "displayName-count-many": "سيدي غانا",
      "displayName-count-other": "سيدي غانا",
      "symbol": "GHS"
    },
    "GIP": {
      "displayName": "جنيه جبل طارق",
      "displayName-count-zero": "جنيه جبل طارق",
      "displayName-count-one": "جنيه جبل طارق",
      "displayName-count-two": "جنيه جبل طارق",
      "displayName-count-few": "جنيه جبل طارق",
      "displayName-count-many": "جنيه جبل طارق",
      "displayName-count-other": "جنيه جبل طارق",
      "symbol": "GIP",
      "symbol-alt-narrow": "£"
    },
    "GMD": {
      "displayName": "دلاسي غامبي",
      "displayName-count-zero": "دلاسي غامبي",
      "displayName-count-one": "دلاسي غامبي",
      "displayName-count-two": "دلاسي غامبي",
      "displayName-count-few": "دلاسي غامبي",
      "displayName-count-many": "دلاسي غامبي",
      "displayName-count-other": "دلاسي غامبي",
      "symbol": "GMD"
    },
    "GNF": {
      "displayName": "فرنك غينيا",
      "displayName-count-zero": "فرنك غينيا",
      "displayName-count-one": "فرنك غينيا",
      "displayName-count-two": "فرنك غينيا",
      "displayName-count-few": "فرنك غينيا",
      "displayName-count-many": "فرنك غينيا",
      "displayName-count-other": "فرنك غينيا",
      "symbol": "GNF",
      "symbol-alt-narrow": "FG"
    },
    "GNS": {
      "displayName": "سيلي غينيا",
      "symbol": "GNS"
    },
    "GQE": {
      "displayName": "اكويل جونيونا غينيا الاستوائية",
      "symbol": "GQE"
    },
    "GRD": {
      "displayName": "دراخما يوناني",
      "symbol": "GRD"
    },
    "GTQ": {
      "displayName": "كوتزال غواتيمالا",

```

```

    "displayName-count-zero": "کوتزال غواتيمالا",
    "displayName-count-one": "کوتزال غواتيمالا",
    "displayName-count-two": "کوتزال غواتيمالا",
    "displayName-count-few": "کوتزال غواتيمالا",
    "displayName-count-many": "کوتزال غواتيمالا",
    "displayName-count-other": "کوتزال غواتيمالا",
    "symbol": "GTQ",
    "symbol-alt-narrow": "Q"
  },
  "GWE": {
    "displayName": "اسکود برتغالي غينيا",
    "symbol": "GWE"
  },
  "GWP": {
    "displayName": "بیزو غينيا بيساو",
    "symbol": "GWP"
  },
  "GYD": {
    "displayName": "دولار غيانا",
    "displayName-count-zero": "دولار غيانا",
    "displayName-count-one": "دولار غيانا",
    "displayName-count-two": "دولار غيانا",
    "displayName-count-few": "دولار غيانا",
    "displayName-count-many": "دولار غيانا",
    "displayName-count-other": "دولار غيانا",
    "symbol": "GYD",
    "symbol-alt-narrow": "GY$"
  },
  "HKD": {
    "displayName": "دولار هونغ كونغ",
    "displayName-count-zero": "دولار هونغ كونغ",
    "displayName-count-one": "دولار هونغ كونغ",
    "displayName-count-two": "دولار هونغ كونغ",
    "displayName-count-few": "دولار هونغ كونغ",
    "displayName-count-many": "دولار هونغ كونغ",
    "displayName-count-other": "دولار هونغ كونغ",
    "symbol": "HK$",
    "symbol-alt-narrow": "HK$"
  },
  "HNL": {
    "displayName": "ليمبيرا هنداروس",
    "displayName-count-zero": "ليمبيرا هندوراس",
    "displayName-count-one": "ليمبيرا هندوراس",
    "displayName-count-two": "ليمبيرا هندوراس",
    "displayName-count-few": "ليمبيرا هندوراس",
    "displayName-count-many": "ليمبيرا هندوراس",
    "displayName-count-other": "ليمبيرا هندوراس",
    "symbol": "HNL",
    "symbol-alt-narrow": "L"
  },
  "HRD": {
    "displayName": "دينار كرواتي",
    "symbol": "HRD"
  },
  "HRK": {
    "displayName": "کونا کرواتى",
    "displayName-count-zero": "کونا کرواتى",

```

```

        "displayName-count-one": "كونا كرواتي",
        "displayName-count-two": "كونا كرواتي",
        "displayName-count-few": "كونا كرواتي",
        "displayName-count-many": "كونا كرواتي",
        "displayName-count-other": "كونا كرواتي",
        "symbol": "HRK",
        "symbol-alt-narrow": "kn"
    },
    "HTG": {
        "displayName": "جوردی هایتي",
        "displayName-count-zero": "جوردی هایتي",
        "displayName-count-one": "جوردی هایتي",
        "displayName-count-two": "جوردی هایتي",
        "displayName-count-few": "جوردی هایتي",
        "displayName-count-many": "جوردی هایتي",
        "displayName-count-other": "جوردی هایتي",
        "symbol": "HTG"
    },
    "HUF": {
        "displayName": "فورينت هنگاري",
        "displayName-count-zero": "فورينت هنگاري",
        "displayName-count-one": "فورينت هنگاري",
        "displayName-count-two": "فورينت هنگاري",
        "displayName-count-few": "فورينت هنگاري",
        "displayName-count-many": "فورينت هنگاري",
        "displayName-count-other": "فورينت هنگاري",
        "symbol": "HUF",
        "symbol-alt-narrow": "Ft"
    },
    "IDR": {
        "displayName": "روبية إندونيسية",
        "displayName-count-zero": "روبية إندونيسية",
        "displayName-count-one": "روبية إندونيسية",
        "displayName-count-two": "روبية إندونيسية",
        "displayName-count-few": "روبية إندونيسية",
        "displayName-count-many": "روبية إندونيسية",
        "displayName-count-other": "روبية إندونيسية",
        "symbol": "IDR",
        "symbol-alt-narrow": "Rp"
    },
    "IEP": {
        "displayName": "جنيه إيرلندي",
        "symbol": "IEP"
    },
    "ILP": {
        "displayName": "جنيه إسرائيلي",
        "symbol": "ILP"
    },
    "ILR": {
        "displayName": "ILR",
        "symbol": "ILR"
    },
    "ILS": {
        "displayName": "شيكل إسرائيلي جديد",
        "displayName-count-zero": "شيكل إسرائيلي جديد",
        "displayName-count-one": "شيكل إسرائيلي جديد",
        "displayName-count-two": "شيكل إسرائيلي جديد",

```

```

    "displayName-count-few": "شيكل إسرائيلي جديد",
    "displayName-count-many": "شيكل إسرائيلي جديد",
    "displayName-count-other": "شيكل إسرائيلي جديد",
    "symbol": "₪",
    "symbol-alt-narrow": "₪"
  },
  "INR": {
    "displayName": "روبية هندي",
    "displayName-count-zero": "روبية هندي",
    "displayName-count-one": "روبية هندي",
    "displayName-count-two": "روبية هندي",
    "displayName-count-few": "روبية هندي",
    "displayName-count-many": "روبية هندي",
    "displayName-count-other": "روبية هندي",
    "symbol": "₹",
    "symbol-alt-narrow": "₹"
  },
  "IQD": {
    "displayName": "دينار عراقي",
    "displayName-count-zero": "دينار عراقي",
    "displayName-count-one": "دينار عراقي",
    "displayName-count-two": "دينار عراقي",
    "displayName-count-few": "دينار عراقي",
    "displayName-count-many": "دينار عراقي",
    "displayName-count-other": "دينار عراقي",
    "symbol": "د.ع."
  },
  "IRR": {
    "displayName": "₪ إيراني",
    "displayName-count-zero": "₪ إيراني",
    "displayName-count-one": "₪ إيراني",
    "displayName-count-two": "₪ إيراني",
    "displayName-count-few": "₪ إيراني",
    "displayName-count-many": "₪ إيراني",
    "displayName-count-other": "₪ إيراني",
    "symbol": "₪.ر."
  },
  "ISJ": {
    "displayName": "ISJ",
    "symbol": "ISJ"
  },
  "ISK": {
    "displayName": "كرونة أيسلندية",
    "displayName-count-zero": "كرونة أيسلندية",
    "displayName-count-one": "كرونة أيسلندية",
    "displayName-count-two": "كرونة أيسلندية",
    "displayName-count-few": "كرونة أيسلندية",
    "displayName-count-many": "كرونة أيسلندية",
    "displayName-count-other": "كرونة أيسلندية",
    "symbol": "ISK",
    "symbol-alt-narrow": "kr"
  },
  "ITL": {
    "displayName": "ليرة إيطالية",
    "symbol": "ITL"
  },
  "JMD": {

```

```

    "displayName": "دولار جامايڪي",
    "displayName-count-zero": "دولار جامايڪي",
    "displayName-count-one": "دولار جامايڪي",
    "displayName-count-two": "دولار جامايڪي",
    "displayName-count-few": "دولار جامايڪي",
    "displayName-count-many": "دولار جامايڪي",
    "displayName-count-other": "دولار جامايڪي",
    "symbol": "JMD",
    "symbol-alt-narrow": "JM$"
  },
  "JOD": {
    "displayName": "دينار اردني",
    "displayName-count-zero": "دينار اردني",
    "displayName-count-one": "دينار اردني",
    "displayName-count-two": "دينار اردني",
    "displayName-count-few": "دينار اردني",
    "displayName-count-many": "دينار اردني",
    "displayName-count-other": "دينار اردني",
    "symbol": "د.أ."
  },
  "JPY": {
    "displayName": "ين ياباني",
    "displayName-count-zero": "ين ياباني",
    "displayName-count-one": "ين ياباني",
    "displayName-count-two": "ين ياباني",
    "displayName-count-few": "ين ياباني",
    "displayName-count-many": "ين ياباني",
    "displayName-count-other": "ين ياباني",
    "symbol": "JP¥",
    "symbol-alt-narrow": "JP¥"
  },
  "KES": {
    "displayName": "شلن كينيي",
    "displayName-count-zero": "شلن كينيي",
    "displayName-count-one": "شلن كينيي",
    "displayName-count-two": "شلن كينيي",
    "displayName-count-few": "شلن كينيي",
    "displayName-count-many": "شلن كينيي",
    "displayName-count-other": "شلن كينيي",
    "symbol": "KES"
  },
  "KGS": {
    "displayName": "سوم قيرغستاني",
    "displayName-count-zero": "سوم قيرغستاني",
    "displayName-count-one": "سوم قيرغستاني",
    "displayName-count-two": "سوم قيرغستاني",
    "displayName-count-few": "سوم قيرغستاني",
    "displayName-count-many": "سوم قيرغستاني",
    "displayName-count-other": "سوم قيرغستاني",
    "symbol": "KGS"
  },
  "KHR": {
    "displayName": "ريال كمبودي",
    "displayName-count-zero": "ريال كمبودي",
    "displayName-count-one": "ريال كمبودي",
    "displayName-count-two": "ريال كمبودي",
    "displayName-count-few": "ريال كمبودي",

```

```

    "displayName-count-many": "ريال كمبودي",
    "displayName-count-other": "ريال كمبودي",
    "symbol": "KHR",
    "symbol-alt-narrow": "₭"
  },
  "KMF": {
    "displayName": "فرنك جزر القمر",
    "displayName-count-zero": "فرنك جزر القمر",
    "displayName-count-one": "فرنك جزر القمر",
    "displayName-count-two": "فرنك جزر القمر",
    "displayName-count-few": "فرنك جزر القمر",
    "displayName-count-many": "فرنك جزر القمر",
    "displayName-count-other": "فرنك جزر القمر",
    "symbol": "KMF",
    "symbol-alt-narrow": "CF"
  },
  "KPW": {
    "displayName": "وون كوريا الشمالية",
    "displayName-count-zero": "وون كوريا الشمالية",
    "displayName-count-one": "وون كوريا الشمالية",
    "displayName-count-two": "وون كوريا الشمالية",
    "displayName-count-few": "وون كوريا الشمالية",
    "displayName-count-many": "وون كوريا الشمالية",
    "displayName-count-other": "وون كوريا الشمالية",
    "symbol": "KPW",
    "symbol-alt-narrow": "₩"
  },
  "KRH": {
    "displayName": "KRH",
    "symbol": "KRH"
  },
  "KRO": {
    "displayName": "KRO",
    "symbol": "KRO"
  },
  "KRW": {
    "displayName": "وون كوريا الجنوبية",
    "displayName-count-zero": "وون كوريا الجنوبية",
    "displayName-count-one": "وون كوريا الجنوبية",
    "displayName-count-two": "وون كوريا الجنوبية",
    "displayName-count-few": "وون كوريا الجنوبية",
    "displayName-count-many": "وون كوريا الجنوبية",
    "displayName-count-other": "وون كوريا الجنوبية",
    "symbol": "₩",
    "symbol-alt-narrow": "₩"
  },
  "KWD": {
    "displayName": "دينار كويتي",
    "displayName-count-zero": "دينار كويتي",
    "displayName-count-one": "دينار كويتي",
    "displayName-count-two": "دينار كويتي",
    "displayName-count-few": "دينار كويتي",
    "displayName-count-many": "دينار كويتي",
    "displayName-count-other": "دينار كويتي",
    "symbol": "د.ك."
  },
},

```



```

"KYD": {
  "displayName": "دولار جزر كيمن",
  "displayName-count-zero": "دولار جزر كيمن",
  "displayName-count-one": "دولار جزر كيمن",
  "displayName-count-two": "دولار جزر كيمن",
  "displayName-count-few": "دولار جزر كيمن",
  "displayName-count-many": "دولار جزر كيمن",
  "displayName-count-other": "دولار جزر كيمن",
  "symbol": "KYD",
  "symbol-alt-narrow": "KY$"
},
"KZT": {
  "displayName": "تينغ كازاخستاني",
  "displayName-count-zero": "تينغ كازاخستاني",
  "displayName-count-one": "تينغ كازاخستاني",
  "displayName-count-two": "تينغ كازاخستاني",
  "displayName-count-few": "تينغ كازاخستاني",
  "displayName-count-many": "تينغ كازاخستاني",
  "displayName-count-other": "تينغ كازاخستاني",
  "symbol": "KZT",
  "symbol-alt-narrow": "₸"
},
"LAK": {
  "displayName": "كيب لاوسي",
  "displayName-count-zero": "كيب لاوسي",
  "displayName-count-one": "كيب لاوسي",
  "displayName-count-two": "كيب لاوسي",
  "displayName-count-few": "كيب لاوسي",
  "displayName-count-many": "كيب لاوسي",
  "displayName-count-other": "كيب لاوسي",
  "symbol": "LAK",
  "symbol-alt-narrow": "₭"
},
"LBP": {
  "displayName": "جنيه لبناني",
  "displayName-count-zero": "جنيه لبناني",
  "displayName-count-one": "جنيه لبناني",
  "displayName-count-two": "جنيه لبناني",
  "displayName-count-few": "جنيه لبناني",
  "displayName-count-many": "جنيه لبناني",
  "displayName-count-other": "جنيه لبناني",
  "symbol": "ل.ل.",
  "symbol-alt-narrow": "L£"
},
"LKR": {
  "displayName": "روبية سريلانكية",
  "displayName-count-zero": "روبية سريلانكية",
  "displayName-count-one": "روبية سريلانكية",
  "displayName-count-two": "روبية سريلانكية",
  "displayName-count-few": "روبية سريلانكية",
  "displayName-count-many": "روبية سريلانكية",
  "displayName-count-other": "روبية سريلانكية",
  "symbol": "LKR",
  "symbol-alt-narrow": "Rs"
},
"LRD": {
  "displayName": "دولار ليبيري",

```

```

    "displayName-count-zero": "دولار ليبيري",
    "displayName-count-one": "دولار ليبيري",
    "displayName-count-two": "دولاران ليبيريان",
    "displayName-count-few": "دولارات ليبيرية",
    "displayName-count-many": "دولارًا ليبيريًا",
    "displayName-count-other": "دولار ليبيري",
    "symbol": "LRD",
    "symbol-alt-narrow": "$",
  },
  "LSL": {
    "displayName": "لوتي ليسوتو",
    "symbol": "LSL"
  },
  "LTL": {
    "displayName": "ليتلا ليتوانية",
    "displayName-count-zero": "ليتلا ليتوانية",
    "displayName-count-one": "ليتلا ليتوانية",
    "displayName-count-two": "ليتلا ليتوانية",
    "displayName-count-few": "ليتلا ليتوانية",
    "displayName-count-many": "ليتلا ليتوانية",
    "displayName-count-other": "ليتلا ليتوانية",
    "symbol": "LTL",
    "symbol-alt-narrow": "Lt"
  },
  "LTT": {
    "displayName": "تالوناس ليتواني",
    "symbol": "LTT"
  },
  "LUC": {
    "displayName": "فرنك لوكسمبرج قابل للتحويل",
    "symbol": "LUC"
  },
  "LUF": {
    "displayName": "فرنك لوكسمبرج",
    "symbol": "LUF"
  },
  "LUL": {
    "displayName": "فرنك لوكسمبرج المالي",
    "symbol": "LUL"
  },
  "LVL": {
    "displayName": "لاتس لاتفيا",
    "displayName-count-zero": "لاتس لاتفي",
    "displayName-count-one": "لاتس لاتفي",
    "displayName-count-two": "لاتس لاتفي",
    "displayName-count-few": "لاتس لاتفي",
    "displayName-count-many": "لاتس لاتفي",
    "displayName-count-other": "لاتس لاتفي",
    "symbol": "LVL",
    "symbol-alt-narrow": "Ls"
  },
  "LVR": {
    "displayName": "روبل لاتفيا",
    "symbol": "LVR"
  },
  "LYD": {
    "displayName": "دينار ليبي",

```

```

    "displayName-count-zero": "دينار ليبي",
    "displayName-count-one": "دينار ليبي",
    "displayName-count-two": "ديناران ليبيان",
    "displayName-count-few": "دينارات ليبية",
    "displayName-count-many": "دينارًا ليبيا",
    "displayName-count-other": "دينار ليبي",
    "symbol": "د.ل.",
  },
  "MAD": {
    "displayName": "درهم مغربي",
    "displayName-count-zero": "درهم مغربي",
    "displayName-count-one": "درهم مغربي",
    "displayName-count-two": "درهمان مغربيان",
    "displayName-count-few": "دراهم مغربية",
    "displayName-count-many": "درهمًا مغربيًا",
    "displayName-count-other": "درهم مغربي",
    "symbol": "د.م.",
  },
  "MAF": {
    "displayName": "فرنك مغربي",
    "symbol": "MAF",
  },
  "MCF": {
    "displayName": "MCF",
    "symbol": "MCF",
  },
  "MDC": {
    "displayName": "MDC",
    "symbol": "MDC",
  },
  "MDL": {
    "displayName": "ليو مولدوفي",
    "displayName-count-zero": "ليو مولدوفي",
    "displayName-count-one": "ليو مولدوفي",
    "displayName-count-two": "ليو مولدوفي",
    "displayName-count-few": "ليو مولدوفي",
    "displayName-count-many": "ليو مولدوفي",
    "displayName-count-other": "ليو مولدوفي",
    "symbol": "MDL",
  },
  "MGA": {
    "displayName": "أرياري مدغشقر",
    "displayName-count-zero": "أرياري مدغشقر",
    "displayName-count-one": "أرياري مدغشقر",
    "displayName-count-two": "أرياري مدغشقر",
    "displayName-count-few": "أرياري مدغشقر",
    "displayName-count-many": "أرياري مدغشقر",
    "displayName-count-other": "أرياري مدغشقر",
    "symbol": "MGA",
    "symbol-alt-narrow": "Ar",
  },
  "MGF": {
    "displayName": "فرنك مدغشقر",
    "symbol": "MGF",
  },
  "MKD": {
    "displayName": "دينار مقدوني",

```

```

    "displayName-count-zero": "دينار مقدوني",
    "displayName-count-one": "دينار مقدوني",
    "displayName-count-two": "ديناران مقدونيان",
    "displayName-count-few": "دينارات مقدونية",
    "displayName-count-many": "دينارًا مقدونيًا",
    "displayName-count-other": "دينار مقدوني",
    "symbol": "MKD"
  },
  "MKN": {
    "displayName": "MKN",
    "symbol": "MKN"
  },
  "MLF": {
    "displayName": "فرنك مالي",
    "symbol": "MLF"
  },
  "MMK": {
    "displayName": "كيات ميانمار",
    "displayName-count-zero": "كيات ميانمار",
    "displayName-count-one": "كيات ميانمار",
    "displayName-count-two": "كيات ميانمار",
    "displayName-count-few": "كيات ميانمار",
    "displayName-count-many": "كيات ميانمار",
    "displayName-count-other": "كيات ميانمار",
    "symbol": "MMK",
    "symbol-alt-narrow": "K"
  },
  "MNT": {
    "displayName": "توغروغ منغولي",
    "displayName-count-zero": "توغروغ منغولي",
    "displayName-count-one": "توغروغ منغولي",
    "displayName-count-two": "توغروغ منغولي",
    "displayName-count-few": "توغروغ منغولي",
    "displayName-count-many": "توغروغ منغولي",
    "displayName-count-other": "توغروغ منغولي",
    "symbol": "MNT",
    "symbol-alt-narrow": "₮"
  },
  "MOP": {
    "displayName": "باتاكا ماكاوي",
    "displayName-count-zero": "باتاكا ماكاوي",
    "displayName-count-one": "باتاكا ماكاوي",
    "displayName-count-two": "باتاكا ماكاوي",
    "displayName-count-few": "باتاكا ماكاوي",
    "displayName-count-many": "باتاكا ماكاوي",
    "displayName-count-other": "باتاكا ماكاوي",
    "symbol": "MOP"
  },
  "MRO": {
    "displayName": "أوقية موريتانية",
    "displayName-count-zero": "أوقية موريتانية",
    "displayName-count-one": "أوقية موريتانية",
    "displayName-count-two": "أوقية موريتانية",
    "displayName-count-few": "أوقية موريتانية",
    "displayName-count-many": "أوقية موريتانية",
    "displayName-count-other": "أوقية موريتانية",
    "symbol": "أ.م."
  }

```

```

    },
    "MTL": {
      "displayName": "ليرة مالطية",
      "symbol": "MTL"
    },
    "MTP": {
      "displayName": "جنيه مالطي",
      "symbol": "MTP"
    },
    "MUR": {
      "displayName": "روبية موريشيوسية",
      "displayName-count-zero": "روبية موريشيوسية",
      "displayName-count-one": "روبية موريشيوسية",
      "displayName-count-two": "روبية موريشيوسية",
      "displayName-count-few": "روبية موريشيوسية",
      "displayName-count-many": "روبية موريشيوسية",
      "displayName-count-other": "روبية موريشيوسية",
      "symbol": "MUR",
      "symbol-alt-narrow": "Rs"
    },
    "MVP": {
      "displayName": "MVP",
      "symbol": "MVP"
    },
    "MVR": {
      "displayName": "روفيه جزر المالديف",
      "displayName-count-zero": "روفيه جزر المالديف",
      "displayName-count-one": "روفيه جزر المالديف",
      "displayName-count-two": "روفيه جزر المالديف",
      "displayName-count-few": "روفيه جزر المالديف",
      "displayName-count-many": "روفيه جزر المالديف",
      "displayName-count-other": "روفيه جزر المالديف",
      "symbol": "MVR"
    },
    "MWK": {
      "displayName": "كواشا مالاوي",
      "displayName-count-zero": "كواشا مالاوي",
      "displayName-count-one": "كواشا مالاوي",
      "displayName-count-two": "كواشا مالاوي",
      "displayName-count-few": "كواشا مالاوي",
      "displayName-count-many": "كواشا مالاوي",
      "displayName-count-other": "كواشا مالاوي",
      "symbol": "MWK"
    },
    "MXN": {
      "displayName": "بيزو مكسيكي",
      "displayName-count-zero": "بيزو مكسيكي",
      "displayName-count-one": "بيزو مكسيكي",
      "displayName-count-two": "بيزو مكسيكي",
      "displayName-count-few": "بيزو مكسيكي",
      "displayName-count-many": "بيزو مكسيكي",
      "displayName-count-other": "بيزو مكسيكي",
      "symbol": "MX$",
      "symbol-alt-narrow": "MX$"
    },
    "MXP": {
      "displayName": "بيزو فضي مكسيكي - 1861-1992",

```

```

    "symbol": "MXP"
  },
  "MXV": {
    "displayName": "MXV",
    "symbol": "MXV"
  },
  "MYR": {
    "displayName": "رينغيت ماليزي",
    "displayName-count-zero": "رينغيت ماليزي",
    "displayName-count-one": "رينغيت ماليزي",
    "displayName-count-two": "رينغيت ماليزي",
    "displayName-count-few": "رينغيت ماليزي",
    "displayName-count-many": "رينغيت ماليزي",
    "displayName-count-other": "رينغيت ماليزي",
    "symbol": "MYR",
    "symbol-alt-narrow": "RM"
  },
  "MZE": {
    "displayName": "اسكود موزمبيقي",
    "symbol": "MZE"
  },
  "MZM": {
    "displayName": "MZM",
    "symbol": "MZM"
  },
  "MZN": {
    "displayName": "متكال موزمبيقي",
    "displayName-count-zero": "متكال موزمبيقي",
    "displayName-count-one": "متكال موزمبيقي",
    "displayName-count-two": "متكال موزمبيقي",
    "displayName-count-few": "متكال موزمبيقي",
    "displayName-count-many": "متكال موزمبيقي",
    "displayName-count-other": "متكال موزمبيقي",
    "symbol": "MZN"
  },
  "NAD": {
    "displayName": "دولار ناميبي",
    "displayName-count-zero": "دولار ناميبي",
    "displayName-count-one": "دولار ناميبي",
    "displayName-count-two": "دولار ناميبي",
    "displayName-count-few": "دولار ناميبي",
    "displayName-count-many": "دولار ناميبي",
    "displayName-count-other": "دولار ناميبي",
    "symbol": "NAD",
    "symbol-alt-narrow": "$"
  },
  "NGN": {
    "displayName": "نايرا نيجيري",
    "displayName-count-zero": "نايرا نيجيري",
    "displayName-count-one": "نايرا نيجيري",
    "displayName-count-two": "نايرا نيجيري",
    "displayName-count-few": "نايرا نيجيري",
    "displayName-count-many": "نايرا نيجيري",
    "displayName-count-other": "نايرا نيجيري",
    "symbol": "NGN",
    "symbol-alt-narrow": "₦"
  },
}

```

```

    "NIC": {
      "displayName": "كوردوبه نيكارا جوا",
      "symbol": "NIC"
    },
    "NIO": {
      "displayName": "قرطبة نيكارا جوا",
      "displayName-count-zero": "قرطبة نيكارا جوا",
      "displayName-count-one": "قرطبة نيكارا جوا",
      "displayName-count-two": "قرطبة نيكارا جوا",
      "displayName-count-few": "قرطبة نيكارا جوا",
      "displayName-count-many": "قرطبة نيكارا جوا",
      "displayName-count-other": "قرطبة نيكارا جوا",
      "symbol": "NIO",
      "symbol-alt-narrow": "C$"
    },
    "NLG": {
      "displayName": "جلدر هولندي",
      "symbol": "NLG"
    },
    "NOK": {
      "displayName": "كرونه نرويجية",
      "displayName-count-zero": "كرونه نرويجية",
      "displayName-count-one": "كرونه نرويجية",
      "displayName-count-two": "كرونه نرويجية",
      "displayName-count-few": "كرونه نرويجية",
      "displayName-count-many": "كرونه نرويجية",
      "displayName-count-other": "كرونه نرويجية",
      "symbol": "NOK",
      "symbol-alt-narrow": "kr"
    },
    "NPR": {
      "displayName": "روبية نيبالي",
      "displayName-count-zero": "روبية نيبالي",
      "displayName-count-one": "روبية نيبالي",
      "displayName-count-two": "روبية نيبالي",
      "displayName-count-few": "روبية نيبالي",
      "displayName-count-many": "روبية نيبالي",
      "displayName-count-other": "روبية نيبالي",
      "symbol": "NPR",
      "symbol-alt-narrow": "Rs"
    },
    "NZD": {
      "displayName": "دولار نيوزيلندي",
      "displayName-count-zero": "دولار نيوزيلندي",
      "displayName-count-one": "دولار نيوزيلندي",
      "displayName-count-two": "دولار نيوزيلندي",
      "displayName-count-few": "دولار نيوزيلندي",
      "displayName-count-many": "دولار نيوزيلندي",
      "displayName-count-other": "دولار نيوزيلندي",
      "symbol": "NZ$",
      "symbol-alt-narrow": "NZ$"
    },
    "OMR": {
      "displayName": "ول عماني",
      "displayName-count-zero": "ول عماني",
      "displayName-count-one": "ول عماني",
      "displayName-count-two": "ول عماني",

```

```

    "displayName-count-few": "٧٥ عماني",
    "displayName-count-many": "٧٥ عماني",
    "displayName-count-other": "٧٥ عماني",
    "symbol": "ر.ع."
  },
  "PAB": {
    "displayName": "بالبوا بنمي",
    "displayName-count-zero": "بالبوا بنمي",
    "displayName-count-one": "بالبوا بنمي",
    "displayName-count-two": "بالبوا بنمي",
    "displayName-count-few": "بالبوا بنمي",
    "displayName-count-many": "بالبوا بنمي",
    "displayName-count-other": "بالبوا بنمي",
    "symbol": "PAB"
  },
  "PEI": {
    "displayName": "PEI",
    "symbol": "PEI"
  },
  "PEN": {
    "displayName": "سول بيروفي",
    "displayName-count-zero": "سول بيروفي",
    "displayName-count-one": "سول بيروفي",
    "displayName-count-two": "سول بيروفي",
    "displayName-count-few": "سول بيروفي",
    "displayName-count-many": "سول بيروفي",
    "displayName-count-other": "سول بيروفي",
    "symbol": "PEN"
  },
  "PES": {
    "displayName": "PES",
    "symbol": "PES"
  },
  "PGK": {
    "displayName": "كينيا بابوا غينيا الجديدة",
    "displayName-count-zero": "كينيا بابوا غينيا الجديدة",
    "displayName-count-one": "كينيا بابوا غينيا الجديدة",
    "displayName-count-two": "كينيا بابوا غينيا الجديدة",
    "displayName-count-few": "كينيا بابوا غينيا الجديدة",
    "displayName-count-many": "كينيا بابوا غينيا الجديدة",
    "displayName-count-other": "كينيا بابوا غينيا الجديدة",
    "symbol": "PGK"
  },
  "PHP": {
    "displayName": "بيزو فلبيني",
    "displayName-count-zero": "بيزو فلبيني",
    "displayName-count-one": "بيزو فلبيني",
    "displayName-count-two": "بيزو فلبيني",
    "displayName-count-few": "بيزو فلبيني",
    "displayName-count-many": "بيزو فلبيني",
    "displayName-count-other": "بيزو فلبيني",
    "symbol": "PHP",
    "symbol-alt-narrow": "₱"
  },
  "PKR": {
    "displayName": "روبية باكستاني",
    "displayName-count-zero": "روبية باكستاني",

```



```

    "displayName-count-one": "روبيۃ باڪستاني",
    "displayName-count-two": "روبيۃ باڪستاني",
    "displayName-count-few": "روبيۃ باڪستاني",
    "displayName-count-many": "روبيۃ باڪستاني",
    "displayName-count-other": "روبيۃ باڪستاني",
    "symbol": "PKR",
    "symbol-alt-narrow": "Rs"
  },
  "PLN": {
    "displayName": "زلوتي پولندي",
    "displayName-count-zero": "زلوتي پولندي",
    "displayName-count-one": "زلوتي پولندي",
    "displayName-count-two": "زلوتي پولندي",
    "displayName-count-few": "زلوتي پولندي",
    "displayName-count-many": "زلوتي پولندي",
    "displayName-count-other": "زلوتي پولندي",
    "symbol": "PLN",
    "symbol-alt-narrow": "zł"
  },
  "PLZ": {
    "displayName": "زلوتي پولندي - 1950-1995",
    "symbol": "PLZ"
  },
  "PTE": {
    "displayName": "اسڪود پرتگالي",
    "symbol": "PTE"
  },
  "PYG": {
    "displayName": "غواراني پاراگواي",
    "displayName-count-zero": "غواراني پاراگواي",
    "displayName-count-one": "غواراني پاراگواي",
    "displayName-count-two": "غواراني پاراگواي",
    "displayName-count-few": "غواراني پاراگواي",
    "displayName-count-many": "غواراني پاراگواي",
    "displayName-count-other": "غواراني پاراگواي",
    "symbol": "PYG",
    "symbol-alt-narrow": "₧"
  },
  "QAR": {
    "displayName": "ڀڙ قطري",
    "displayName-count-zero": "ڀڙ قطري",
    "displayName-count-one": "ڀڙ قطري",
    "displayName-count-two": "ڀڙ قطري",
    "displayName-count-few": "ڀڙ قطري",
    "displayName-count-many": "ڀڙ قطري",
    "displayName-count-other": "ڀڙ قطري",
    "symbol": "ر.ق."
  },
  "RHD": {
    "displayName": "ڊولار روڊيسي",
    "symbol": "RHD"
  },
  "ROL": {
    "displayName": "ليو روماني قديم",
    "symbol": "ROL"
  },
  "RON": {

```

```

        "displayName": "ليو روماني",
        "displayName-count-zero": "ليو روماني",
        "displayName-count-one": "ليو روماني",
        "displayName-count-two": "ليو روماني",
        "displayName-count-few": "ليو روماني",
        "displayName-count-many": "ليو روماني",
        "displayName-count-other": "ليو روماني",
        "symbol": "RON",
        "symbol-alt-narrow": "lei"
    },
    "RSD": {
        "displayName": "دينار صربي",
        "displayName-count-zero": "دينار صربي",
        "displayName-count-one": "دينار صربي",
        "displayName-count-two": "ديناران صربيان",
        "displayName-count-few": "دينارات صربية",
        "displayName-count-many": "دينارًا صربيًا",
        "displayName-count-other": "دينار صربي",
        "symbol": "RSD"
    },
    "RUB": {
        "displayName": "روبل روسي",
        "displayName-count-zero": "روبل روسي",
        "displayName-count-one": "روبل روسي",
        "displayName-count-two": "روبل روسي",
        "displayName-count-few": "روبل روسي",
        "displayName-count-many": "روبل روسي",
        "displayName-count-other": "روبل روسي",
        "symbol": "RUB",
        "symbol-alt-narrow": "₽"
    },
    "RUR": {
        "displayName": "روبل روسي - 1998-1991",
        "symbol": "RUR",
        "symbol-alt-narrow": "p."
    },
    "RWF": {
        "displayName": "فرنك رواندي",
        "displayName-count-zero": "فرنك رواندي",
        "displayName-count-one": "فرنك رواندي",
        "displayName-count-two": "فرنك رواندي",
        "displayName-count-few": "فرنك رواندي",
        "displayName-count-many": "فرنك رواندي",
        "displayName-count-other": "فرنك رواندي",
        "symbol": "RWF",
        "symbol-alt-narrow": "RF"
    },
    "SAR": {
        "displayName": "ريال سعودي",
        "displayName-count-zero": "ريال سعودي",
        "displayName-count-one": "ريال سعودي",
        "displayName-count-two": "ريال سعودي",
        "displayName-count-few": "ريال سعودي",
        "displayName-count-many": "ريال سعودي",
        "displayName-count-other": "ريال سعودي",
        "symbol": "ر.س."
    },
    },

```

```

"SBD": {
  "displayName": "دولار جزر سليمان",
  "displayName-count-zero": "دولار جزر سليمان",
  "displayName-count-one": "دولار جزر سليمان",
  "displayName-count-two": "دولار جزر سليمان",
  "displayName-count-few": "دولار جزر سليمان",
  "displayName-count-many": "دولار جزر سليمان",
  "displayName-count-other": "دولار جزر سليمان",
  "symbol": "SBD",
  "symbol-alt-narrow": "SB$"
},
"SCR": {
  "displayName": "روبية سيشيلية",
  "displayName-count-zero": "روبية سيشيلية",
  "displayName-count-one": "روبية سيشيلية",
  "displayName-count-two": "روبية سيشيلية",
  "displayName-count-few": "روبية سيشيلية",
  "displayName-count-many": "روبية سيشيلية",
  "displayName-count-other": "روبية سيشيلية",
  "symbol": "SCR"
},
"SDD": {
  "displayName": "دينار سوداني",
  "symbol": "د.س."
},
"SDG": {
  "displayName": "جنيه سوداني",
  "displayName-count-zero": "جنيه سوداني",
  "displayName-count-one": "جنيه سوداني",
  "displayName-count-two": "جنيه سوداني",
  "displayName-count-few": "جنيهات سودانية",
  "displayName-count-many": "جنيها سودانيا",
  "displayName-count-other": "جنيه سوداني",
  "symbol": "ج.س."
},
"SDP": {
  "displayName": "جنيه سوداني قديم",
  "symbol": "SDP"
},
"SEK": {
  "displayName": "كرونة سويدية",
  "displayName-count-zero": "كرونة سويدية",
  "displayName-count-one": "كرونة سويدية",
  "displayName-count-two": "كرونة سويدية",
  "displayName-count-few": "كرونة سويدية",
  "displayName-count-many": "كرونة سويدية",
  "displayName-count-other": "كرونة سويدية",
  "symbol": "SEK",
  "symbol-alt-narrow": "kr"
},
"SGD": {
  "displayName": "دولار سنغافوري",
  "displayName-count-zero": "دولار سنغافوري",
  "displayName-count-one": "دولار سنغافوري",
  "displayName-count-two": "دولار سنغافوري",
  "displayName-count-few": "دولار سنغافوري",
  "displayName-count-many": "دولار سنغافوري",

```

```

        "displayName-count-other": "دولار سنغافوري",
        "symbol": "SGD",
        "symbol-alt-narrow": "$"
    },
    "SHP": {
        "displayName": "جنيه سانت هيلين",
        "displayName-count-zero": "جنيه سانت هيلين",
        "displayName-count-one": "جنيه سانت هيلين",
        "displayName-count-two": "جنيه سانت هيلين",
        "displayName-count-few": "جنيه سانت هيلين",
        "displayName-count-many": "جنيه سانت هيلين",
        "displayName-count-other": "جنيه سانت هيلين",
        "symbol": "SHP",
        "symbol-alt-narrow": "£"
    },
    "SIT": {
        "displayName": "تولار سلوفيني",
        "symbol": "SIT"
    },
    "SKK": {
        "displayName": "كرونة سلوفاكية",
        "symbol": "SKK"
    },
    "SLL": {
        "displayName": "ليون سيراليوني",
        "displayName-count-zero": "ليون سيراليوني",
        "displayName-count-one": "ليون سيراليوني",
        "displayName-count-two": "ليون سيراليوني",
        "displayName-count-few": "ليون سيراليوني",
        "displayName-count-many": "ليون سيراليوني",
        "displayName-count-other": "ليون سيراليوني",
        "symbol": "SLL"
    },
    "SOS": {
        "displayName": "شلن صومالي",
        "displayName-count-zero": "شلن صومالي",
        "displayName-count-one": "شلن صومالي",
        "displayName-count-two": "شلن صومالي",
        "displayName-count-few": "شلن صومالي",
        "displayName-count-many": "شلن صومالي",
        "displayName-count-other": "شلن صومالي",
        "symbol": "SOS"
    },
    "SRD": {
        "displayName": "دولار سورينامي",
        "displayName-count-zero": "دولار سورينامي",
        "displayName-count-one": "دولار سورينامي",
        "displayName-count-two": "دولار سورينامي",
        "displayName-count-few": "دولار سورينامي",
        "displayName-count-many": "دولار سورينامي",
        "displayName-count-other": "دولار سورينامي",
        "symbol": "SRD",
        "symbol-alt-narrow": "SR$"
    },
    "SRG": {
        "displayName": "جلدر سورينامي",
        "symbol": "SRG"
    }

```

```

    },
    "SSP": {
      "displayName": "جنيه جنوب السودان",
      "displayName-count-zero": "جنيه جنوب السودان",
      "displayName-count-one": "جنيه جنوب السودان",
      "displayName-count-two": "جنيهان جنوب السودان",
      "displayName-count-few": "جنيهات جنوب السودان",
      "displayName-count-many": "جنيهًا جنوب السودان",
      "displayName-count-other": "جنيه جنوب السودان",
      "symbol": "SSP",
      "symbol-alt-narrow": "£"
    },
    "STD": {
      "displayName": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-zero": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-one": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-two": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-few": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-many": "دوبرا ساو تومي وبرينسيبي",
      "displayName-count-other": "دوبرا ساو تومي وبرينسيبي",
      "symbol": "STD",
      "symbol-alt-narrow": "Db"
    },
    "STN": {
      "displayName": "STN",
      "symbol": "STN"
    },
    "SUR": {
      "displayName": "روبل سوفيتي",
      "symbol": "SUR"
    },
    "SVC": {
      "displayName": "كولون سلفادوري",
      "symbol": "SVC"
    },
    "SYP": {
      "displayName": "ليرة سورية",
      "displayName-count-zero": "ليرة سورية",
      "displayName-count-one": "ليرة سورية",
      "displayName-count-two": "ليرة سورية",
      "displayName-count-few": "ليرة سورية",
      "displayName-count-many": "ليرة سورية",
      "displayName-count-other": "ليرة سورية",
      "symbol": "ل.س.",
      "symbol-alt-narrow": "£"
    },
    "SZL": {
      "displayName": "ليلانجيني سوازيلندي",
      "displayName-count-zero": "ليلانجيني سوازيلندي",
      "displayName-count-one": "ليلانجيني سوازيلندي",
      "displayName-count-two": "ليلانجيني سوازيلندي",
      "displayName-count-few": "ليلانجيني سوازيلندي",
      "displayName-count-many": "ليلانجيني سوازيلندي",
      "displayName-count-other": "ليلانجيني سوازيلندي",
      "symbol": "SZL"
    },
    "THB": {

```

```

    "displayName": "باخت تایلاندي",
    "displayName-count-zero": "باخت تایلاندي",
    "displayName-count-one": "باخت تایلاندي",
    "displayName-count-two": "باخت تایلاندي",
    "displayName-count-few": "باخت تایلاندي",
    "displayName-count-many": "باخت تایلاندي",
    "displayName-count-other": "باخت تایلاندي",
    "symbol": "฿",
    "symbol-alt-narrow": "฿"
  },
  "TJR": {
    "displayName": "روبل طاجیکستانی",
    "symbol": "TJR"
  },
  "TJS": {
    "displayName": "سومونی طاجیکستانی",
    "displayName-count-zero": "سومونی طاجیکستانی",
    "displayName-count-one": "سومونی طاجیکستانی",
    "displayName-count-two": "سومونی طاجیکستانی",
    "displayName-count-few": "سومونی طاجیکستانی",
    "displayName-count-many": "سومونی طاجیکستانی",
    "displayName-count-other": "سومونی طاجیکستانی",
    "symbol": "TJS"
  },
  "TMM": {
    "displayName": "مانات ترکمنستانی",
    "symbol": "TMM"
  },
  "TMT": {
    "displayName": "مانات ترکمانستان",
    "displayName-count-zero": "مانات ترکمانستان",
    "displayName-count-one": "مانات ترکمانستان",
    "displayName-count-two": "مانات ترکمانستان",
    "displayName-count-few": "مانات ترکمانستان",
    "displayName-count-many": "مانات ترکمانستان",
    "displayName-count-other": "مانات ترکمانستان",
    "symbol": "TMT"
  },
  "TND": {
    "displayName": "دینار تونس",
    "displayName-count-zero": "دینار تونس",
    "displayName-count-one": "دینار تونس",
    "displayName-count-two": "دیناران تونس",
    "displayName-count-few": "دینارات تونسیة",
    "displayName-count-many": "دینارًا تونسیًا",
    "displayName-count-other": "دینار تونس",
    "symbol": "د.ت."
  },
  "TOP": {
    "displayName": "بانغا تونگا",
    "displayName-count-zero": "بانغا تونگا",
    "displayName-count-one": "بانغا تونگا",
    "displayName-count-two": "بانغا تونگا",
    "displayName-count-few": "بانغا تونگا",
    "displayName-count-many": "بانغا تونگا",
    "displayName-count-other": "بانغا تونگا",
    "symbol": "TOP",

```

```

    "symbol-alt-narrow": "T$"
  },
  "TPE": {
    "displayName": "اسكود تيموري",
    "symbol": "TPE"
  },
  "TRL": {
    "displayName": "ليرة تركي",
    "symbol": "TRL"
  },
  "TRY": {
    "displayName": "ليرة تركية",
    "displayName-count-zero": "ليرة تركية",
    "displayName-count-one": "ليرة تركية",
    "displayName-count-two": "ليرة تركية",
    "displayName-count-few": "ليرة تركية",
    "displayName-count-many": "ليرة تركية",
    "displayName-count-other": "ليرة تركية",
    "symbol": "TRY",
    "symbol-alt-narrow": "₺",
    "symbol-alt-variant": "TL"
  },
  "TTD": {
    "displayName": "دولار ترينداد وتوباغو",
    "displayName-count-zero": "دولار ترينداد وتوباغو",
    "displayName-count-one": "دولار ترينداد وتوباغو",
    "displayName-count-two": "دولار ترينداد وتوباغو",
    "displayName-count-few": "دولار ترينداد وتوباغو",
    "displayName-count-many": "دولار ترينداد وتوباغو",
    "displayName-count-other": "دولار ترينداد وتوباغو",
    "symbol": "TTD",
    "symbol-alt-narrow": "TT$"
  },
  "TWD": {
    "displayName": "دولار تايواني",
    "displayName-count-zero": "دولار تايواني",
    "displayName-count-one": "دولار تايواني",
    "displayName-count-two": "دولار تايواني",
    "displayName-count-few": "دولار تايواني",
    "displayName-count-many": "دولار تايواني",
    "displayName-count-other": "دولار تايواني",
    "symbol": "NT$",
    "symbol-alt-narrow": "NT$"
  },
  "TZS": {
    "displayName": "شلن تنزاني",
    "displayName-count-zero": "شلن تنزاني",
    "displayName-count-one": "شلن تنزاني",
    "displayName-count-two": "شلن تنزاني",
    "displayName-count-few": "شلن تنزاني",
    "displayName-count-many": "شلن تنزاني",
    "displayName-count-other": "شلن تنزاني",
    "symbol": "TZS"
  },
  "UAH": {
    "displayName": "هريفنيا أوكراني",
    "displayName-count-zero": "هريفنيا أوكراني",

```

```

    "displayName-count-one": "هريفنيا أوكراني",
    "displayName-count-two": "هريفنيا أوكراني",
    "displayName-count-few": "هريفنيا أوكراني",
    "displayName-count-many": "هريفنيا أوكراني",
    "displayName-count-other": "هريفنيا أوكراني",
    "symbol": "UAH",
    "symbol-alt-narrow": "₴"
  },
  "UAK": {
    "displayName": "UAK",
    "symbol": "UAK"
  },
  "UGS": {
    "displayName": "1987-1966 - شلن أوغندي",
    "symbol": "UGS"
  },
  "UGX": {
    "displayName": "شلن أوغندي",
    "displayName-count-zero": "شلن أوغندي",
    "displayName-count-one": "شلن أوغندي",
    "displayName-count-two": "شلن أوغندي",
    "displayName-count-few": "شلن أوغندي",
    "displayName-count-many": "شلن أوغندي",
    "displayName-count-other": "شلن أوغندي",
    "symbol": "UGX"
  },
  "USD": {
    "displayName": "دولار أمريكي",
    "displayName-count-zero": "دولار أمريكي",
    "displayName-count-one": "دولار أمريكي",
    "displayName-count-two": "دولار أمريكي",
    "displayName-count-few": "دولار أمريكي",
    "displayName-count-many": "دولار أمريكي",
    "displayName-count-other": "دولار أمريكي",
    "symbol": "US$",
    "symbol-alt-narrow": "US$"
  },
  "USN": {
    "displayName": "دولار أمريكي (اليوم التالي)",
    "symbol": "USN"
  },
  "USS": {
    "displayName": "دولار أمريكي (نفس اليوم)",
    "symbol": "USS"
  },
  "UYI": {
    "displayName": "UYI",
    "symbol": "UYI"
  },
  "UYP": {
    "displayName": "بيزو أوروغواي - 1993-1975",
    "symbol": "UYP"
  },
  "UYU": {
    "displayName": "بيزو أوروغواي",
    "displayName-count-zero": "بيزو أوروغواي",
    "displayName-count-one": "بيزو أوروغواي",

```



```

    "displayName-count-two": "بیزو اوروغوای",
    "displayName-count-few": "بیزو اوروغوای",
    "displayName-count-many": "بیزو اوروغوای",
    "displayName-count-other": "بیزو اوروغوای",
    "symbol": "UYU",
    "symbol-alt-narrow": "UY$"
  },
  "UZS": {
    "displayName": "سوم اۆزبکستانی",
    "displayName-count-zero": "سوم اۆزبکستانی",
    "displayName-count-one": "سوم اۆزبکستانی",
    "displayName-count-two": "سوم اۆزبکستانی",
    "displayName-count-few": "سوم اۆزبکستانی",
    "displayName-count-many": "سوم اۆزبکستانی",
    "displayName-count-other": "سوم اۆزبکستانی",
    "symbol": "UZS"
  },
  "VEB": {
    "displayName": "بولیفار فنزویلی - 2008-1871",
    "symbol": "VEB"
  },
  "VEF": {
    "displayName": "بولیفار فنزویلی",
    "displayName-count-zero": "بولیفار فنزویلی",
    "displayName-count-one": "بولیفار فنزویلی",
    "displayName-count-two": "بولیفار فنزویلی",
    "displayName-count-few": "بولیفار فنزویلی",
    "displayName-count-many": "بولیفار فنزویلی",
    "displayName-count-other": "بولیفار فنزویلی",
    "symbol": "VEF",
    "symbol-alt-narrow": "Bs"
  },
  "VND": {
    "displayName": "دونج فیتنامی",
    "displayName-count-zero": "دونج فیتنامی",
    "displayName-count-one": "دونج فیتنامی",
    "displayName-count-two": "دونج فیتنامی",
    "displayName-count-few": "دونج فیتنامی",
    "displayName-count-many": "دونج فیتنامی",
    "displayName-count-other": "دونج فیتنامی",
    "symbol": "₫",
    "symbol-alt-narrow": "₫"
  },
  "VNN": {
    "displayName": "VNN",
    "symbol": "VNN"
  },
  "VUV": {
    "displayName": "فاتو فانواتو",
    "displayName-count-zero": "فاتو فانواتو",
    "displayName-count-one": "فاتو فانواتو",
    "displayName-count-two": "فاتو فانواتو",
    "displayName-count-few": "فاتو فانواتو",
    "displayName-count-many": "فاتو فانواتو",
    "displayName-count-other": "فاتو فانواتو",
    "symbol": "VUV"
  },
}

```

```

"WST": {
  "displayName": "تالا ساموا",
  "displayName-count-zero": "تالا ساموا",
  "displayName-count-one": "تالا ساموا",
  "displayName-count-two": "تالا ساموا",
  "displayName-count-few": "تالا ساموا",
  "displayName-count-many": "تالا ساموا",
  "displayName-count-other": "تالا ساموا",
  "symbol": "WST"
},
"XAF": {
  "displayName": "فرنك وسط أفريقي",
  "displayName-count-zero": "فرنك وسط أفريقي",
  "displayName-count-one": "فرنك وسط أفريقي",
  "displayName-count-two": "فرنك وسط أفريقي",
  "displayName-count-few": "فرنك وسط أفريقي",
  "displayName-count-many": "فرنك وسط أفريقي",
  "displayName-count-other": "فرنك وسط أفريقي",
  "symbol": "FCFA"
},
"XAG": {
  "displayName": "فضة",
  "symbol": "XAG"
},
"XAU": {
  "displayName": "ذهب",
  "symbol": "XAU"
},
"XBA": {
  "displayName": "الوحدة الأوروبية المركبة",
  "symbol": "XBA"
},
"XBB": {
  "displayName": "الوحدة المالية الأوروبية",
  "symbol": "XBB"
},
"XBC": {
  "displayName": "الوحدة الحسابية الأوروبية",
  "symbol": "XBC"
},
"XBD": {
  "displayName": "وحدة الحساب الأوروبية (XBD)",
  "symbol": "XBD"
},
"XCD": {
  "displayName": "دولار شرق الكاريبي",
  "displayName-count-zero": "دولار شرق الكاريبي",
  "displayName-count-one": "دولار شرق الكاريبي",
  "displayName-count-two": "دولار شرق الكاريبي",
  "displayName-count-few": "دولار شرق الكاريبي",
  "displayName-count-many": "دولار شرق الكاريبي",
  "displayName-count-other": "دولار شرق الكاريبي",
  "symbol": "EC$",
  "symbol-alt-narrow": "$"
},
"XDR": {
  "displayName": "حقوق السحب الخاصة",

```

```

    "symbol": "XDR"
  },
  "XEU": {
    "displayName": "وحدة النقد الأوروبية",
    "symbol": "XEU"
  },
  "XFO": {
    "displayName": "فرنك فرنسي ذهبي",
    "symbol": "XFO"
  },
  "XFU": {
    "displayName": "فرنك فرنسي (UIC)",
    "symbol": "XFU"
  },
  "XOF": {
    "displayName": "فرنك غرب أفريقي",
    "displayName-count-zero": "فرنك غرب أفريقي",
    "displayName-count-one": "فرنك غرب أفريقي",
    "displayName-count-two": "فرنك غرب أفريقي",
    "displayName-count-few": "فرنك غرب أفريقي",
    "displayName-count-many": "فرنك غرب أفريقي",
    "displayName-count-other": "فرنك غرب أفريقي",
    "symbol": "CFA"
  },
  "XPD": {
    "displayName": "بالاديوم",
    "symbol": "XPD"
  },
  "XPF": {
    "displayName": "فرنك سي إف بي",
    "displayName-count-zero": "فرنك سي إف بي",
    "displayName-count-one": "فرنك سي إف بي",
    "displayName-count-two": "فرنك سي إف بي",
    "displayName-count-few": "فرنك سي إف بي",
    "displayName-count-many": "فرنك سي إف بي",
    "displayName-count-other": "فرنك سي إف بي",
    "symbol": "CFPF"
  },
  "XPT": {
    "displayName": "البلاتين",
    "symbol": "XPT"
  },
  "XRE": {
    "displayName": "XRE",
    "symbol": "XRE"
  },
  "XSU": {
    "displayName": "XSU",
    "symbol": "XSU"
  },
  "XTS": {
    "displayName": "كود اختبار العملة",
    "symbol": "XTS"
  },
  "XUA": {
    "displayName": "XUA",
    "symbol": "XUA"
  }

```

```

    },
    "XXX": {
      "displayName": "عملة غير معروفة",
      "displayName-count-zero": "(عملة غير معروفة)",
      "displayName-count-one": "(عملة غير معروفة)",
      "displayName-count-two": "(عملة غير معروفة)",
      "displayName-count-few": "(عملة غير معروفة)",
      "displayName-count-many": "(عملة غير معروفة)",
      "displayName-count-other": "(عملة غير معروفة)",
      "symbol": "***"
    },
    "YDD": {
      "displayName": "دينار يمني",
      "symbol": "YDD"
    },
    "YER": {
      "displayName": "ول يمني",
      "displayName-count-zero": "ول يمني",
      "displayName-count-one": "ول يمني",
      "displayName-count-two": "ول يمني",
      "displayName-count-few": "ول يمني",
      "displayName-count-many": "ول يمني",
      "displayName-count-other": "ول يمني",
      "symbol": "ر.ي."
    },
    "YUD": {
      "displayName": "دينار يوغسلافي",
      "symbol": "YUD"
    },
    "YUM": {
      "displayName": "YUM",
      "symbol": "YUM"
    },
    "YUN": {
      "displayName": "دينار يوغسلافي قابل للتحويل",
      "symbol": "YUN"
    },
    "YUR": {
      "displayName": "YUR",
      "symbol": "YUR"
    },
    "ZAL": {
      "displayName": "راند جنوب أفريقيا -مالي",
      "symbol": "ZAL"
    },
    "ZAR": {
      "displayName": "راند جنوب أفريقيا",
      "displayName-count-zero": "راند جنوب أفريقيا",
      "displayName-count-one": "راند جنوب أفريقيا",
      "displayName-count-two": "راند جنوب أفريقيا",
      "displayName-count-few": "راند جنوب أفريقيا",
      "displayName-count-many": "راند جنوب أفريقيا",
      "displayName-count-other": "راند جنوب أفريقيا",
      "symbol": "ZAR",
      "symbol-alt-narrow": "R"
    },
    "ZMK": {

```

```
"displayName": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-zero": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-one": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-two": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-few": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-many": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"displayName-count-other": "2012-1968 - \"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
"symbol": "ZMK"  
},  
"ZMW": {  
    "displayName": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-zero": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-one": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-two": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-few": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-many": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "displayName-count-other": "\"\u06a9\u0648\u0627\u0634\u0627 \u0630\u0627\u0645\u0628\u06cc\"",  
    "symbol": "ZMW",  
    "symbol-alt-narrow": "ZK"  
},  
"ZRN": {  
    "displayName": "\"\u0630\u0627\u06cc\u0631 \u0630\u0627\u06cc\u0631\u06cc \u062c\u062f\u06cc\u062f\"",  
    "symbol": "ZRN"  
},  
"ZRZ": {  
    "displayName": "\"\u0630\u0627\u06cc\u0631 \u0630\u0627\u06cc\u0631\u06cc\"",  
    "symbol": "ZRZ"  
},  
"ZWD": {  
    "displayName": "\"\u062f\u0648\u0644\u0627\u0631 \u0630\u0645\u0628\u0627\u0628\u0648\u06cc\"",  
    "symbol": "ZWD"  
},  
"ZWL": {  
    "displayName": "\"\u062f\u0648\u0644\u0627\u0631 \u0630\u0645\u0628\u0627\u0628\u0648\u06cc 2009\"",  
    "symbol": "ZWL"  
},  
"ZWR": {  
    "displayName": "ZWR",  
    "symbol": "ZWR"  
}  
}  
}  
}
```

CURRENCIES.JSX

```
{
    "main";
    {
        "ar";
        {
            "identity";
            {
                "version";
            }
        }
    }
}
```

```

        {
            "_number";
            "$Revision: 13686 $",
            "_cldrVersion";
            "32";
        }
        "language";
        "ar";
    }
    "numbers";
    {
        "currencies";
        {
            "ADP";
            {
                "displayName";
                "بيستا أندوري",
                "symbol";
                "ADP";
            }
            "AED";
            {
                "displayName";
                "درهم إماراتي",
                "displayName-count-zero";
                "درهم إماراتي",
                "displayName-count-one";
                "درهم إماراتي",
                "displayName-count-two";
                "درهم إماراتي",
                "displayName-count-few";
                "درهم إماراتي",
                "displayName-count-many";
                "درهم إماراتي",
                "displayName-count-other";
                "درهم إماراتي",
                "symbol";
                "د.إ.";
            }
            "AFA";
            {
                "displayName";
                "أفغاني - 2002-1927",
                "symbol";
                "AFA";
            }
            "AFN";
            {
                "displayName";
                "أفغاني",
                "displayName-count-zero";
                "أفغاني أفغانستان",
                "displayName-count-one";
                "أفغاني أفغانستان",
                "displayName-count-two";
                "أفغاني أفغانستان",
                "displayName-count-few";
            }
        }
    }

```

```

        "أفغاني أفغانستان",
        "displayName-count-many";
        "أفغاني أفغانستان",
        "displayName-count-other";
        "أفغاني أفغانستان",
        "symbol";
        "AFN";
    }
    "ALK";
    {
        "displayName";
        "ALK",
        "symbol";
        "ALK";
    }
    "ALL";
    {
        "displayName";
        "ليك ألباني",
        "displayName-count-zero";
        "ليك ألباني",
        "displayName-count-one";
        "ليك ألباني",
        "displayName-count-two";
        "ليك ألباني",
        "displayName-count-few";
        "ليك ألباني",
        "displayName-count-many";
        "ليك ألباني",
        "displayName-count-other";
        "ليك ألباني",
        "symbol";
        "ALL";
    }
    "AMD";
    {
        "displayName";
        "درام أرميني",
        "displayName-count-zero";
        "درام أرميني",
        "displayName-count-one";
        "درام أرميني",
        "displayName-count-two";
        "درام أرميني",
        "displayName-count-few";
        "درام أرميني",
        "displayName-count-many";
        "درام أرميني",
        "displayName-count-other";
        "درام أرميني",
        "symbol";
        "AMD";
    }
    "ANG";
    {
        "displayName";
        "غيلدر أنتيلي هولندي",

```

```

        "displayName-count-zero";
        "غيلدر أنتيلي هولندي",
        "displayName-count-one";
        "غيلدر أنتيلي هولندي",
        "displayName-count-two";
        "غيلدر أنتيلي هولندي",
        "displayName-count-few";
        "غيلدر أنتيلي هولندي",
        "displayName-count-many";
        "غيلدر أنتيلي هولندي",
        "displayName-count-other";
        "غيلدر أنتيلي هولندي",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "كوانزا أنغولي",
        "displayName-count-zero";
        "كوانزا أنغولي",
        "displayName-count-one";
        "كوانزا أنغولي",
        "displayName-count-two";
        "كوانزا أنغولي",
        "displayName-count-few";
        "كوانزا أنغولي",
        "displayName-count-many";
        "كوانزا أنغولي",
        "displayName-count-other";
        "كوانزا أنغولي",
        "symbol";
        "AOA",
        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "كوانزا أنجولي - 1990-1977",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "كوانزا أنجولي جديدة - 2000-1990",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "كوانزا أنجولي معدلة - 1999 - 1995",
        "symbol";
        "AOR";
    }
}

```



```

"ARA";
{
  "displayName";
  "استرال أرجنتيني",
  "symbol";
  "ARA";
}
"ARL";
{
  "displayName";
  "ARL",
  "symbol";
  "ARL";
}
"ARM";
{
  "displayName";
  "ARM",
  "symbol";
  "ARM";
}
"ARP";
{
  "displayName";
  "1985-1983 - أرجنتيني",
  "symbol";
  "ARP";
}
"ARS";
{
  "displayName";
  "بيزو أرجنتيني",
  "displayName-count-zero";
  "بيزو أرجنتيني",
  "displayName-count-one";
  "بيزو أرجنتيني",
  "displayName-count-two";
  "بيزو أرجنتيني",
  "displayName-count-few";
  "بيزو أرجنتيني",
  "displayName-count-many";
  "بيزو أرجنتيني",
  "displayName-count-other";
  "بيزو أرجنتيني",
  "symbol";
  "ARS",
  "symbol-alt-narrow";
  "AR$";
}
"ATS";
{
  "displayName";
  "شلن نمساوي",
  "symbol";
  "ATS";
}
"AUD";

```

```

    {
      "displayName";
      "دولار أسترالي",
      "displayName-count-zero";
      "دولار أسترالي",
      "displayName-count-one";
      "دولار أسترالي",
      "displayName-count-two";
      "دولار أسترالي",
      "displayName-count-few";
      "دولار أسترالي",
      "displayName-count-many";
      "دولار أسترالي",
      "displayName-count-other";
      "دولار أسترالي",
      "symbol";
      "AU$";
      "symbol-alt-narrow";
      "AU$";
    }
    "AWG";
    {
      "displayName";
      "فلورن أروبي",
      "displayName-count-zero";
      "فلورن أروبي",
      "displayName-count-one";
      "فلورن أروبي",
      "displayName-count-two";
      "فلورن أروبي",
      "displayName-count-few";
      "فلورن أروبي",
      "displayName-count-many";
      "فلورن أروبي",
      "displayName-count-other";
      "فلورن أروبي",
      "symbol";
      "AWG";
    }
    "AZM";
    {
      "displayName";
      "مانات أذربيجاني",
      "symbol";
      "AZM";
    }
    "AZN";
    {
      "displayName";
      "مانات أذربيجان",
      "displayName-count-zero";
      "مانت أذربيجاني",
      "displayName-count-one";
      "مانت أذربيجاني",
      "displayName-count-two";
      "مانت أذربيجاني",
      "displayName-count-few";
    }

```

```

        "مانت أذربيجاني",
        "displayName-count-many";
        "مانت أذربيجاني",
        "displayName-count-other";
        "مانت أذربيجاني",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "دينار البوسنة والهرسك",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-zero";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-one";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-two";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-few";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-many";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-other";
        "مارك البوسنة والهرسك قابل للتحويل",
        "symbol";
        "BAM",
        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "BAN",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "دولار بربادوسي",
        "displayName-count-zero";
        "دولار بربادوسي",
        "displayName-count-one";
        "دولار بربادوسي",
        "displayName-count-two";
        "دولار بربادوسي",
        "displayName-count-few";
        "دولار بربادوسي",
        "displayName-count-many";
        "دولار بربادوسي",
    }

```

```

        "displayName-count-other";
        "دولار بربادوسي",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "BB$";
    }
    "BDT";
    {
        "displayName";
        "তাকা বাংলাদেশি",
        "displayName-count-zero";
        "তাকা বাংলাদেশি",
        "displayName-count-one";
        "তাকা বাংলাদেশি",
        "displayName-count-two";
        "তাকা বাংলাদেশি",
        "displayName-count-few";
        "তাকা বাংলাদেশি",
        "displayName-count-many";
        "তাকা বাংলাদেশি",
        "displayName-count-other";
        "তাকা বাংলাদেশি",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }
    "BEC";
    {
        "displayName";
        "فرنك بلجيكي قابل للتحويل",
        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "فرنك بلجيكي",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "فرنك بلجيكي مالي",
        "symbol";
        "BEL";
    }
    "BGL";
    {
        "displayName";
        "BGL",
        "symbol";
        "BGL";
    }
}

```

```
"BGM";
{
  "displayName";
  "BGM",
  "symbol";
  "BGM";
}
"BGN";
{
  "displayName";
  "ليف بلغاري",
  "displayName-count-zero";
  "ليف بلغاري",
  "displayName-count-one";
  "ليف بلغاري",
  "displayName-count-two";
  "ليف بلغاري",
  "displayName-count-few";
  "ليف بلغاري",
  "displayName-count-many";
  "ليف بلغاري",
  "displayName-count-other";
  "ليف بلغاري",
  "symbol";
  "BGN";
}
"BGO";
{
  "displayName";
  "BGO",
  "symbol";
  "BGO";
}
"BHD";
{
  "displayName";
  "دينار بحريني",
  "displayName-count-zero";
  "دينار بحريني",
  "displayName-count-one";
  "دينار بحريني",
  "displayName-count-two";
  "دينار بحريني",
  "displayName-count-few";
  "دينار بحريني",
  "displayName-count-many";
  "دينار بحريني",
  "displayName-count-other";
  "دينار بحريني",
  "symbol";
  "د.ب.";
}
"BIF";
{
  "displayName";
  "فرنك بروندي",
  "displayName-count-zero";
```

```

        "فرنك بروندي",
        "displayName-count-one";
        "فرنك بروندي",
        "displayName-count-two";
        "فرنك بروندي",
        "displayName-count-few";
        "فرنك بروندي",
        "displayName-count-many";
        "فرنك بروندي",
        "displayName-count-other";
        "فرنك بروندي",
        "symbol";
        "BIF";
    }
    "BMD";
    {
        "displayName";
        "دولار برمودي",
        "displayName-count-zero";
        "دولار برمودي",
        "displayName-count-one";
        "دولار برمودي",
        "displayName-count-two";
        "دولار برمودي",
        "displayName-count-few";
        "دولار برمودي",
        "displayName-count-many";
        "دولار برمودي",
        "displayName-count-other";
        "دولار برمودي",
        "symbol";
        "BMD",
        "symbol-alt-narrow";
        "BM$";
    }
    "BND";
    {
        "displayName";
        "دولار بروناي",
        "displayName-count-zero";
        "دولار بروناي",
        "displayName-count-one";
        "دولار بروناي",
        "displayName-count-two";
        "دولار بروناي",
        "displayName-count-few";
        "دولار بروناي",
        "displayName-count-many";
        "دولار بروناي",
        "displayName-count-other";
        "دولار بروناي",
        "symbol";
        "BND",
        "symbol-alt-narrow";
        "BN$";
    }
    "BOB";

```

```

    {
      "displayName";
      "بولىفيانو بولىفي",
      "displayName-count-zero";
      "بولىفيانو بولىفي",
      "displayName-count-one";
      "بولىفيانو بولىفي",
      "displayName-count-two";
      "بولىفيانو بولىفي",
      "displayName-count-few";
      "بولىفيانو بولىفي",
      "displayName-count-many";
      "بولىفيانو بولىفي",
      "displayName-count-other";
      "بولىفيانو بولىفي",
      "symbol";
      "BOB",
      "symbol-alt-narrow";
      "Bs";
    }
    "BOL";
    {
      "displayName";
      "BOL",
      "symbol";
      "BOL";
    }
    "BOP";
    {
      "displayName";
      "بىزو بولىفي",
      "symbol";
      "BOP";
    }
    "BOV";
    {
      "displayName";
      "مفدول بولىفي",
      "symbol";
      "BOV";
    }
    "BRB";
    {
      "displayName";
      "نوفو كروزايرو برازيلى - 1986-1967",
      "symbol";
      "BRB";
    }
    "BRC";
    {
      "displayName";
      "كروزالدو برازيلى",
      "symbol";
      "BRC";
    }
    "BRE";
    {

```

```

        "displayName";
        "1993-1990 - كروزايرو برازيلى",
        "symbol";
        "BRE";
    }
    "BRL";
    {
        "displayName";
        "ريال برازيلى",
        "displayName-count-zero";
        "ريال برازيلى",
        "displayName-count-one";
        "ريال برازيلى",
        "displayName-count-two";
        "ريال برازيلى",
        "displayName-count-few";
        "ريال برازيلى",
        "displayName-count-many";
        "ريال برازيلى",
        "displayName-count-other";
        "ريال برازيلى",
        "symbol";
        "R$",
        "symbol-alt-narrow";
        "R$";
    }
    "BRN";
    {
        "displayName";
        "BRN",
        "symbol";
        "BRN";
    }
    "BRR";
    {
        "displayName";
        "BRR",
        "symbol";
        "BRR";
    }
    "BRZ";
    {
        "displayName";
        "BRZ",
        "symbol";
        "BRZ";
    }
    "BSD";
    {
        "displayName";
        "دولار باهامى",
        "displayName-count-zero";
        "دولار باهامى",
        "displayName-count-one";
        "دولار باهامى",
        "displayName-count-two";
        "دولار باهامى";
    }

```



```

        "displayName-count-few";
        "دولار باهامي",
        "displayName-count-many";
        "دولار باهامي",
        "displayName-count-other";
        "دولار باهامي",
        "symbol";
        "BSD",
        "symbol-alt-narrow";
        "BS$";
    }
    "BTN";
    {
        "displayName";
        "نولتوم بوتاني",
        "displayName-count-zero";
        "نولتوم بوتاني",
        "displayName-count-one";
        "نولتوم بوتاني",
        "displayName-count-two";
        "نولتوم بوتاني",
        "displayName-count-few";
        "نولتوم بوتاني",
        "displayName-count-many";
        "نولتوم بوتاني",
        "displayName-count-other";
        "نولتوم بوتاني",
        "symbol";
        "BTN";
    }
    "BUK";
    {
        "displayName";
        "کيات بورمي",
        "symbol";
        "BUK";
    }
    "BWP";
    {
        "displayName";
        "بولا بتسواني",
        "displayName-count-zero";
        "بولا بتسواني",
        "displayName-count-one";
        "بولا بتسواني",
        "displayName-count-two";
        "بولا بتسواني",
        "displayName-count-few";
        "بولا بتسواني",
        "displayName-count-many";
        "بولا بتسواني",
        "displayName-count-other";
        "بولا بتسواني",
        "symbol";
        "BWP",
        "symbol-alt-narrow";
        "P";
    }

```

```

    }
    "BYB";
    {
      "displayName";
      "1999-1994 - روبل بيلاروسي جديد",
      "symbol";
      "BYB";
    }
    "BYN";
    {
      "displayName";
      "روبل بيلاروسي",
      "displayName-count-zero";
      "روبل بيلاروسي",
      "displayName-count-one";
      "روبل بيلاروسي",
      "displayName-count-two";
      "روبل بيلاروسي",
      "displayName-count-few";
      "روبل بيلاروسي",
      "displayName-count-many";
      "روبل بيلاروسي",
      "displayName-count-other";
      "روبل بيلاروسي",
      "symbol";
      "BYN",
      "symbol-alt-narrow";
      "p.";
    }
    "BYR";
    {
      "displayName";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-zero";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-one";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-two";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-few";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-many";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "displayName-count-other";
      "۲۰۱۶-۲۰۰۰ (روبل بيلاروسي)",
      "symbol";
      "BYR";
    }
    "BZD";
    {
      "displayName";
      "دولار بليزي",
      "displayName-count-zero";
      "دولار بليزي",
      "displayName-count-one";
      "دولار بليزي",
      "displayName-count-two";
    }

```

```

        "دولار ان بليزيان",
        "displayName-count-few";
        "دولار بليزي",
        "displayName-count-many";
        "دولار بليزي",
        "displayName-count-other";
        "دولار بليزي",
        "symbol";
        "BZD",
        "symbol-alt-narrow";
        "BZ$";
    }
    "CAD";
    {
        "displayName";
        "دولار كندي",
        "displayName-count-zero";
        "دولار كندي",
        "displayName-count-one";
        "دولار كندي",
        "displayName-count-two";
        "دولار كندي",
        "displayName-count-few";
        "دولار كندي",
        "displayName-count-many";
        "دولار كندي",
        "displayName-count-other";
        "دولار كندي",
        "symbol";
        "CA$",
        "symbol-alt-narrow";
        "CA$";
    }
    "CDF";
    {
        "displayName";
        "فرنك كونغولي",
        "displayName-count-zero";
        "فرنك كونغولي",
        "displayName-count-one";
        "فرنك كونغولي",
        "displayName-count-two";
        "فرنك كونغولي",
        "displayName-count-few";
        "فرنك كونغولي",
        "displayName-count-many";
        "فرنك كونغولي",
        "displayName-count-other";
        "فرنك كونغولي",
        "symbol";
        "CDF";
    }
    "CHE";
    {
        "displayName";
        "CHE",
        "symbol";
    }

```

```

        "CHE";
    }
    "CHF";
    {
        "displayName";
        "فرنك سويسري",
        "displayName-count-zero";
        "فرنك سويسري",
        "displayName-count-one";
        "فرنك سويسري",
        "displayName-count-two";
        "فرنك سويسري",
        "displayName-count-few";
        "فرنك سويسري",
        "displayName-count-many";
        "فرنك سويسري",
        "displayName-count-other";
        "فرنك سويسري",
        "symbol";
        "CHF";
    }
    "CHW";
    {
        "displayName";
        "CHW",
        "symbol";
        "CHW";
    }
    "CLE";
    {
        "displayName";
        "CLE",
        "symbol";
        "CLE";
    }
    "CLF";
    {
        "displayName";
        "CLF",
        "symbol";
        "CLF";
    }
    "CLP";
    {
        "displayName";
        "بيزو تشيلي",
        "displayName-count-zero";
        "بيزو تشيلي",
        "displayName-count-one";
        "بيزو تشيلي",
        "displayName-count-two";
        "بيزو تشيلي",
        "displayName-count-few";
        "بيزو تشيلي",
        "displayName-count-many";
        "بيزو تشيلي",
        "displayName-count-other";
    }

```

```

        "بيزو تشيلي",
        "symbol";
    "CLP",
        "symbol-alt-narrow";
    "CL$";
}
"CNH";
{
    "displayName";
    "يوان صيني (في الخارج)",
        "displayName-count-zero";
    "يوان صيني (في الخارج)",
        "displayName-count-one";
    "يوان صيني (في الخارج)",
        "displayName-count-two";
    "يوان صيني (في الخارج)",
        "displayName-count-few";
    "يوان صيني (في الخارج)",
        "displayName-count-many";
    "يوان صيني (في الخارج)",
        "displayName-count-other";
    "يوان صيني (في الخارج)",
        "symbol";
    "CNH";
}
"CNX";
{
    "displayName";
    "CNX",
        "symbol";
    "CNX";
}
"CNY";
{
    "displayName";
    "يوان صيني",
        "displayName-count-zero";
    "يوان صيني",
        "displayName-count-one";
    "يوان صيني",
        "displayName-count-two";
    "يوان صيني",
        "displayName-count-few";
    "يوان صيني",
        "displayName-count-many";
    "يوان صيني",
        "displayName-count-other";
    "يوان صيني",
        "symbol";
    "CN¥",
        "symbol-alt-narrow";
    "CN¥";
}
"COP";
{
    "displayName";
    "بيزو كولومبي",

```

```

        "displayName-count-zero";
        "بیزو کولومبی",
        "displayName-count-one";
        "بیزو کولومبی",
        "displayName-count-two";
        "بیزو کولومبی",
        "displayName-count-few";
        "بیزو کولومبی",
        "displayName-count-many";
        "بیزو کولومبی",
        "displayName-count-other";
        "بیزو کولومبی",
        "symbol";
        "COP",
        "symbol-alt-narrow";
        "CO$";
    }
    "COU";
    {
        "displayName";
        "COU",
        "symbol";
        "COU";
    }
    "CRC";
    {
        "displayName";
        "کولن کوستاریکی",
        "displayName-count-zero";
        "کولن کوستاریکی",
        "displayName-count-one";
        "کولن کوستاریکی",
        "displayName-count-two";
        "کولن کوستاریکی",
        "displayName-count-few";
        "کولن کوستاریکی",
        "displayName-count-many";
        "کولن کوستاریکی",
        "displayName-count-other";
        "کولن کوستاریکی",
        "symbol";
        "CRC",
        "symbol-alt-narrow";
        "₡";
    }
    "CSD";
    {
        "displayName";
        "دینار صربی قدیم",
        "symbol";
        "CSD";
    }
    "CSK";
    {
        "displayName";
        "کرونة تشیکوسلواکیا",
        "symbol";
    }

```

```

        "CSK";
    }
    "CUC";
    {
        "displayName";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-zero";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-one";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-two";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-few";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-many";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-other";
        "بيزو كوبي قابل للتحويل",
        "symbol";
        "CUC",
        "symbol-alt-narrow";
        "$";
    }
    "CUP";
    {
        "displayName";
        "بيزو كوبي",
        "displayName-count-zero";
        "بيزو كوبي",
        "displayName-count-one";
        "بيزو كوبي",
        "displayName-count-two";
        "بيزو كوبي",
        "displayName-count-few";
        "بيزو كوبي",
        "displayName-count-many";
        "بيزو كوبي",
        "displayName-count-other";
        "بيزو كوبي",
        "symbol";
        "CUP",
        "symbol-alt-narrow";
        "CU$";
    }
    "CVE";
    {
        "displayName";
        "اسكودو الرأس الخضراء",
        "displayName-count-zero";
        "اسكودو الرأس الخضراء",
        "displayName-count-one";
        "اسكودو الرأس الخضراء",
        "displayName-count-two";
        "اسكودو الرأس الخضراء",
        "displayName-count-few";
        "اسكودو الرأس الخضراء",
        "displayName-count-many";
    }

```

```

        "اسكودو الرأس الخضراء",
        "displayName-count-other";
        "اسكودو الرأس الخضراء",
        "symbol";
        "CVE";
    }
    "CYP";
    {
        "displayName";
        "جنیه قبرصي",
        "symbol";
        "CYP";
    }
    "CZK";
    {
        "displayName";
        "كرونة تشيكية",
        "displayName-count-zero";
        "كرونة تشيكية",
        "displayName-count-one";
        "كرونة تشيكية",
        "displayName-count-two";
        "كرونة تشيكية",
        "displayName-count-few";
        "كرونة تشيكية",
        "displayName-count-many";
        "كرونة تشيكية",
        "displayName-count-other";
        "كرونة تشيكية",
        "symbol";
        "CZK",
        "symbol-alt-narrow";
        "Kč";
    }
    "DDM";
    {
        "displayName";
        "أوستمارك ألماني شرقي",
        "symbol";
        "DDM";
    }
    "DEM";
    {
        "displayName";
        "مارك ألماني",
        "symbol";
        "DEM";
    }
    "DJF";
    {
        "displayName";
        "فرنك جيبوتي",
        "displayName-count-zero";
        "فرنك جيبوتي",
        "displayName-count-one";
        "فرنك جيبوتي",
        "displayName-count-two";
    }

```



```

        "فرنك جيبوتي",
        "displayName-count-few";
        "فرنك جيبوتي",
        "displayName-count-many";
        "فرنك جيبوتي",
        "displayName-count-other";
        "فرنك جيبوتي",
        "symbol";
        "DJF";
    }
    "DKK";
    {
        "displayName";
        "كرونة دنماركية",
        "displayName-count-zero";
        "كرونة دنماركية",
        "displayName-count-one";
        "كرونة دنماركية",
        "displayName-count-two";
        "كرونة دنماركية",
        "displayName-count-few";
        "كرونة دنماركية",
        "displayName-count-many";
        "كرونة دنماركية",
        "displayName-count-other";
        "كرونة دنماركية",
        "symbol";
        "DKK",
        "symbol-alt-narrow";
        "kr";
    }
    "DOP";
    {
        "displayName";
        "بيزو الدومنيكان",
        "displayName-count-zero";
        "بيزو الدومنيكان",
        "displayName-count-one";
        "بيزو الدومنيكان",
        "displayName-count-two";
        "بيزو الدومنيكان",
        "displayName-count-few";
        "بيزو الدومنيكان",
        "displayName-count-many";
        "بيزو الدومنيكان",
        "displayName-count-other";
        "بيزو الدومنيكان",
        "symbol";
        "DOP",
        "symbol-alt-narrow";
        "DO$";
    }
    "DZD";
    {
        "displayName";
        "دينار جزائري",
        "displayName-count-zero";

```

```

        "دينار جزائري",
        "displayName-count-one";
        "دينار جزائري",
        "displayName-count-two";
        "ديناران جزائريان",
        "displayName-count-few";
        "دينارات جزائرية",
        "displayName-count-many";
        "دينارًا جزائريًا",
        "displayName-count-other";
        "دينار جزائري",
        "symbol";
        "د.ج.";
    }
    "ECS";
    {
        "displayName";
        "ECS",
        "symbol";
        "ECS";
    }
    "ECV";
    {
        "displayName";
        "ECV",
        "symbol";
        "ECV";
    }
    "EEK";
    {
        "displayName";
        "كرونة استونية",
        "symbol";
        "EEK";
    }
    "EGP";
    {
        "displayName";
        "جنيه مصري",
        "displayName-count-zero";
        "جنيه مصري",
        "displayName-count-one";
        "جنيه مصري",
        "displayName-count-two";
        "جنيهان مصريان",
        "displayName-count-few";
        "جنيهات مصرية",
        "displayName-count-many";
        "جنيهاً مصريًا",
        "displayName-count-other";
        "جنيه مصري",
        "symbol";
        "ج.م.",
        "symbol-alt-narrow";
        "£";
    }
    "ERN";

```

```

    {
      "displayName";
      "ناكفا أريتري",
      "displayName-count-zero";
      "ناكفا أريتري",
      "displayName-count-one";
      "ناكفا أريتري",
      "displayName-count-two";
      "ناكفا أريتري",
      "displayName-count-few";
      "ناكفا أريتري",
      "displayName-count-many";
      "ناكفا أريتري",
      "displayName-count-other";
      "ناكفا أريتري",
      "symbol";
      "ERN";
    }
    "ESA";
    {
      "displayName";
      "ESA",
      "symbol";
      "ESA";
    }
    "ESB";
    {
      "displayName";
      "ESB",
      "symbol";
      "ESB";
    }
    "ESP";
    {
      "displayName";
      "بیزیتا إسبانی",
      "symbol";
      "ESP",
      "symbol-alt-narrow";
      "₧";
    }
    "ETB";
    {
      "displayName";
      "بیر أثیوبی",
      "displayName-count-zero";
      "بیر أثیوبی",
      "displayName-count-one";
      "بیر أثیوبی",
      "displayName-count-two";
      "بیر أثیوبی",
      "displayName-count-few";
      "بیر أثیوبی",
      "displayName-count-many";
      "بیر أثیوبی",
      "displayName-count-other";
      "بیر أثیوبی",
    }

```

```

        "symbol";
        "ETB";
    }
    "EUR";
    {
        "displayName";
        "يورو",
        "displayName-count-zero";
        "يورو",
        "displayName-count-one";
        "يورو",
        "displayName-count-two";
        "يورو",
        "displayName-count-few";
        "يورو",
        "displayName-count-many";
        "يورو",
        "displayName-count-other";
        "يورو",
        "symbol";
        "€",
        "symbol-alt-narrow";
        "€";
    }
    "FIM";
    {
        "displayName";
        "ماركا فنلندي",
        "symbol";
        "FIM";
    }
    "FJD";
    {
        "displayName";
        "دولار فيجي",
        "displayName-count-zero";
        "دولار فيجي",
        "displayName-count-one";
        "دولار فيجي",
        "displayName-count-two";
        "دولار فيجي",
        "displayName-count-few";
        "دولار فيجي",
        "displayName-count-many";
        "دولار فيجي",
        "displayName-count-other";
        "دولار فيجي",
        "symbol";
        "FJD",
        "symbol-alt-narrow";
        "FJ$";
    }
    "FKP";
    {
        "displayName";
        "جنيه جزر فوكلاند",
        "displayName-count-zero";

```

```

        "جنيه جزر فوكلاند",
        "displayName-count-one";
        "جنيه جزر فوكلاند",
        "displayName-count-two";
        "جنيه جزر فوكلاند",
        "displayName-count-few";
        "جنيه جزر فوكلاند",
        "displayName-count-many";
        "جنيه جزر فوكلاند",
        "displayName-count-other";
        "جنيه جزر فوكلاند",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "£";
    }
    "FRF";
    {
        "displayName";
        "فرنك فرنسي",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "جنيه إسترليني",
        "displayName-count-zero";
        "جنيه إسترليني",
        "displayName-count-one";
        "جنيه إسترليني",
        "displayName-count-two";
        "جنيه إسترليني",
        "displayName-count-few";
        "جنيه إسترليني",
        "displayName-count-many";
        "جنيه إسترليني",
        "displayName-count-other";
        "جنيه إسترليني",
        "symbol";
        "£",
        "symbol-alt-narrow";
        "UK£";
    }
    "GEK";
    {
        "displayName";
        "GEK",
        "symbol";
        "GEK";
    }
    "GEL";
    {
        "displayName";
        "لاري جورجي",
        "displayName-count-zero";
        "لاري جورجي",

```

```

        "displayName-count-one";
        "ლარი გორგი",
        "displayName-count-two";
        "ლარი გორგი",
        "displayName-count-few";
        "ლარი გორგი",
        "displayName-count-many";
        "ლარი გორგი",
        "displayName-count-other";
        "ლარი გორგი",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }
    "GHC";
    {
        "displayName";
        "სიდი განი",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "სიდი განა",
        "displayName-count-zero";
        "სიდი განა",
        "displayName-count-one";
        "სიდი განა",
        "displayName-count-two";
        "სიდი განა",
        "displayName-count-few";
        "სიდი განა",
        "displayName-count-many";
        "სიდი განა",
        "displayName-count-other";
        "სიდი განა",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "განიე ბილ ტარი",
        "displayName-count-zero";
        "განიე ბილ ტარი",
        "displayName-count-one";
        "განიე ბილ ტარი",
        "displayName-count-two";
        "განიე ბილ ტარი",
        "displayName-count-few";
        "განიე ბილ ტარი",
        "displayName-count-many";
        "განიე ბილ ტარი",
    }

```

```

        "displayName-count-other";
        "جنيه جبل طارق",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
    "GMD";
    {
        "displayName";
        "دلّاسي غامبي",
        "displayName-count-zero";
        "دلّاسي غامبي",
        "displayName-count-one";
        "دلّاسي غامبي",
        "displayName-count-two";
        "دلّاسي غامبي",
        "displayName-count-few";
        "دلّاسي غامبي",
        "displayName-count-many";
        "دلّاسي غامبي",
        "displayName-count-other";
        "دلّاسي غامبي",
        "symbol";
        "GMD";
    }
    "GNF";
    {
        "displayName";
        "فرنك غينيا",
        "displayName-count-zero";
        "فرنك غينيا",
        "displayName-count-one";
        "فرنك غينيا",
        "displayName-count-two";
        "فرنك غينيا",
        "displayName-count-few";
        "فرنك غينيا",
        "displayName-count-many";
        "فرنك غينيا",
        "displayName-count-other";
        "فرنك غينيا",
        "symbol";
        "GNF",
        "symbol-alt-narrow";
        "FG";
    }
    "GNS";
    {
        "displayName";
        "سيلبي غينيا",
        "symbol";
        "GNS";
    }
    "GQE";
    {
        "displayName";

```

```

        "أكويل جونيئا غينيا الاستوائية",
        "symbol";
        "GQE";
    }
    "GRD";
    {
        "displayName";
        "دراخما يوناني",
        "symbol";
        "GRD";
    }
    "GTQ";
    {
        "displayName";
        "كوتزال غواتيمالا",
        "displayName-count-zero";
        "كوتزال غواتيمالا",
        "displayName-count-one";
        "كوتزال غواتيمالا",
        "displayName-count-two";
        "كوتزال غواتيمالا",
        "displayName-count-few";
        "كوتزال غواتيمالا",
        "displayName-count-many";
        "كوتزال غواتيمالا",
        "displayName-count-other";
        "كوتزال غواتيمالا",
        "symbol";
        "GTQ",
        "symbol-alt-narrow";
        "Q";
    }
    "GWE";
    {
        "displayName";
        "اسكود برتغالي غينيا",
        "symbol";
        "GWE";
    }
    "GWP";
    {
        "displayName";
        "بيزو غينيا بيساو",
        "symbol";
        "GWP";
    }
    "GYD";
    {
        "displayName";
        "دولار غيانا",
        "displayName-count-zero";
        "دولار غيانا",
        "displayName-count-one";
        "دولار غيانا",
        "displayName-count-two";
        "دولار غيانا",
        "displayName-count-few";
    }

```



```

        "دولار غيانا",
        "displayName-count-many";
        "دولار غيانا",
        "displayName-count-other";
        "دولار غيانا",
        "symbol";
        "GYD",
        "symbol-alt-narrow";
        "GY$";
    }
    "HKD";
    {
        "displayName";
        "دولار هونغ كونغ",
        "displayName-count-zero";
        "دولار هونغ كونغ",
        "displayName-count-one";
        "دولار هونغ كونغ",
        "displayName-count-two";
        "دولار هونغ كونغ",
        "displayName-count-few";
        "دولار هونغ كونغ",
        "displayName-count-many";
        "دولار هونغ كونغ",
        "displayName-count-other";
        "دولار هونغ كونغ",
        "symbol";
        "HK$",
        "symbol-alt-narrow";
        "HK$";
    }
    "HNL";
    {
        "displayName";
        "ليمبيرا هندوراس",
        "displayName-count-zero";
        "ليمبيرا هندوراس",
        "displayName-count-one";
        "ليمبيرا هندوراس",
        "displayName-count-two";
        "ليمبيرا هندوراس",
        "displayName-count-few";
        "ليمبيرا هندوراس",
        "displayName-count-many";
        "ليمبيرا هندوراس",
        "displayName-count-other";
        "ليمبيرا هندوراس",
        "symbol";
        "HNL",
        "symbol-alt-narrow";
        "L";
    }
    "HRD";
    {
        "displayName";
        "دينار كرواتي",
        "symbol";
    }

```

```

        "HRD";
    }
    "HRK";
    {
        "displayName";
        "کونا کرواتى",
        "displayName-count-zero";
        "کونا کرواتى",
        "displayName-count-one";
        "کونا کرواتى",
        "displayName-count-two";
        "کونا کرواتى",
        "displayName-count-few";
        "کونا کرواتى",
        "displayName-count-many";
        "کونا کرواتى",
        "displayName-count-other";
        "کونا کرواتى",
        "symbol";
        "HRK",
        "symbol-alt-narrow";
        "kn";
    }
    "HTG";
    {
        "displayName";
        "جوردی هایتي",
        "displayName-count-zero";
        "جوردی هایتي",
        "displayName-count-one";
        "جوردی هایتي",
        "displayName-count-two";
        "جوردی هایتي",
        "displayName-count-few";
        "جوردی هایتي",
        "displayName-count-many";
        "جوردی هایتي",
        "displayName-count-other";
        "جوردی هایتي",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "فورينت هنگاري",
        "displayName-count-zero";
        "فورينت هنگاري",
        "displayName-count-one";
        "فورينت هنگاري",
        "displayName-count-two";
        "فورينت هنگاري",
        "displayName-count-few";
        "فورينت هنگاري",
        "displayName-count-many";
        "فورينت هنگاري",
        "displayName-count-other";
    }

```

```

        "فورينت هنگاري",
        "symbol";
    "HUF",
        "symbol-alt-narrow";
    "Ft";
}
"IDR";
{
    "displayName";
    "روبية إندونيسية",
        "displayName-count-zero";
    "روبية إندونيسية",
        "displayName-count-one";
    "روبية إندونيسية",
        "displayName-count-two";
    "روبية إندونيسية",
        "displayName-count-few";
    "روبية إندونيسية",
        "displayName-count-many";
    "روبية إندونيسية",
        "displayName-count-other";
    "روبية إندونيسية",
        "symbol";
    "IDR",
        "symbol-alt-narrow";
    "Rp";
}
"IEP";
{
    "displayName";
    "جنيه إيرلندي",
        "symbol";
    "IEP";
}
"ILP";
{
    "displayName";
    "جنيه إسرائيلي",
        "symbol";
    "ILP";
}
"ILR";
{
    "displayName";
    "ILR",
        "symbol";
    "ILR";
}
"ILS";
{
    "displayName";
    "شيكل إسرائيلي جديد",
        "displayName-count-zero";
    "شيكل إسرائيلي جديد",
        "displayName-count-one";
    "شيكل إسرائيلي جديد",
        "displayName-count-two";
}

```

```

        "شيكل إسرائيلى جديد",
        "displayName-count-few";
        "شيكل إسرائيلى جديد",
        "displayName-count-many";
        "شيكل إسرائيلى جديد",
        "displayName-count-other";
        "شيكل إسرائيلى جديد",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "روبية هندي",
        "displayName-count-zero";
        "روبية هندي",
        "displayName-count-one";
        "روبية هندي",
        "displayName-count-two";
        "روبية هندي",
        "displayName-count-few";
        "روبية هندي",
        "displayName-count-many";
        "روبية هندي",
        "displayName-count-other";
        "روبية هندي",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
    "IQD";
    {
        "displayName";
        "دينار عراقي",
        "displayName-count-zero";
        "دينار عراقي",
        "displayName-count-one";
        "دينار عراقي",
        "displayName-count-two";
        "دينار عراقي",
        "displayName-count-few";
        "دينار عراقي",
        "displayName-count-many";
        "دينار عراقي",
        "displayName-count-other";
        "دينار عراقي",
        "symbol";
        "د.ع.";
    }
    "IRR";
    {
        "displayName";
        "هل ایرانى",
        "displayName-count-zero";
    }

```

```

        "دولار إيراني",
        "displayName-count-one";
        "دولار إيراني",
        "displayName-count-two";
        "دولار إيراني",
        "displayName-count-few";
        "دولار إيراني",
        "displayName-count-many";
        "دولار إيراني",
        "displayName-count-other";
        "دولار إيراني",
        "symbol";
        "ر.إ.";
    }
    "ISJ";
    {
        "displayName";
        "ISJ",
        "symbol";
        "ISJ";
    }
    "ISK";
    {
        "displayName";
        "كرونة أيسلندية",
        "displayName-count-zero";
        "كرونة أيسلندية",
        "displayName-count-one";
        "كرونة أيسلندية",
        "displayName-count-two";
        "كرونة أيسلندية",
        "displayName-count-few";
        "كرونة أيسلندية",
        "displayName-count-many";
        "كرونة أيسلندية",
        "displayName-count-other";
        "كرونة أيسلندية",
        "symbol";
        "ISK",
        "symbol-alt-narrow";
        "kr";
    }
    "ITL";
    {
        "displayName";
        "ليرة إيطالية",
        "symbol";
        "ITL";
    }
    "JMD";
    {
        "displayName";
        "دولار جامايكي",
        "displayName-count-zero";
        "دولار جامايكي",
        "displayName-count-one";
        "دولار جامايكي",

```

```

        "displayName-count-two";
        "دولار جامايكي",
        "displayName-count-few";
        "دولار جامايكي",
        "displayName-count-many";
        "دولار جامايكي",
        "displayName-count-other";
        "دولار جامايكي",
        "symbol";
        "JMD",
        "symbol-alt-narrow";
        "JM$";
    }
    "JOD";
    {
        "displayName";
        "دينار أردني",
        "displayName-count-zero";
        "دينار أردني",
        "displayName-count-one";
        "دينار أردني",
        "displayName-count-two";
        "دينار أردني",
        "displayName-count-few";
        "دينار أردني",
        "displayName-count-many";
        "دينار أردني",
        "displayName-count-other";
        "دينار أردني",
        "symbol";
        "د.أ.";
    }
    "JPY";
    {
        "displayName";
        "ين ياباني",
        "displayName-count-zero";
        "ين ياباني",
        "displayName-count-one";
        "ين ياباني",
        "displayName-count-two";
        "ين ياباني",
        "displayName-count-few";
        "ين ياباني",
        "displayName-count-many";
        "ين ياباني",
        "displayName-count-other";
        "ين ياباني",
        "symbol";
        "JP¥",
        "symbol-alt-narrow";
        "JP¥";
    }
    "KES";
    {
        "displayName";
        "شلن كينيي",

```

```

        "displayName-count-zero";
        "شلن كينيي",
        "displayName-count-one";
        "شلن كينيي",
        "displayName-count-two";
        "شلن كينيي",
        "displayName-count-few";
        "شلن كينيي",
        "displayName-count-many";
        "شلن كينيي",
        "displayName-count-other";
        "شلن كينيي",
        "symbol";
        "KES";
    }
    "KGS";
    {
        "displayName";
        "سوم قيرغستاني",
        "displayName-count-zero";
        "سوم قيرغستاني",
        "displayName-count-one";
        "سوم قيرغستاني",
        "displayName-count-two";
        "سوم قيرغستاني",
        "displayName-count-few";
        "سوم قيرغستاني",
        "displayName-count-many";
        "سوم قيرغستاني",
        "displayName-count-other";
        "سوم قيرغستاني",
        "symbol";
        "KGS";
    }
    "KHR";
    {
        "displayName";
        "ريال كمبودي",
        "displayName-count-zero";
        "ريال كمبودي",
        "displayName-count-one";
        "ريال كمبودي",
        "displayName-count-two";
        "ريال كمبودي",
        "displayName-count-few";
        "ريال كمبودي",
        "displayName-count-many";
        "ريال كمبودي",
        "displayName-count-other";
        "ريال كمبودي",
        "symbol";
        "KHR",
        "symbol-alt-narrow";
        "฿";
    }
    "KMF";

```

```

    {
      "displayName";
      "فرنك جزر القمر",
      "displayName-count-zero";
      "فرنك جزر القمر",
      "displayName-count-one";
      "فرنك جزر القمر",
      "displayName-count-two";
      "فرنك جزر القمر",
      "displayName-count-few";
      "فرنك جزر القمر",
      "displayName-count-many";
      "فرنك جزر القمر",
      "displayName-count-other";
      "فرنك جزر القمر",
      "symbol";
      "KMF",
      "symbol-alt-narrow";
      "CF";
    }
    "KPW";
    {
      "displayName";
      "وون كوريا الشمالية",
      "displayName-count-zero";
      "وون كوريا الشمالية",
      "displayName-count-one";
      "وون كوريا الشمالية",
      "displayName-count-two";
      "وون كوريا الشمالية",
      "displayName-count-few";
      "وون كوريا الشمالية",
      "displayName-count-many";
      "وون كوريا الشمالية",
      "displayName-count-other";
      "وون كوريا الشمالية",
      "symbol";
      "KPW",
      "symbol-alt-narrow";
      "₩";
    }
    "KRH";
    {
      "displayName";
      "KRH",
      "symbol";
      "KRH";
    }
    "KRO";
    {
      "displayName";
      "KRO",
      "symbol";
      "KRO";
    }
    "KRW";
    {

```



```

        "displayName";
        "وون كوريا الجنوبية",
        "displayName-count-zero";
        "وون كوريا الجنوبية",
        "displayName-count-one";
        "وون كوريا الجنوبية",
        "displayName-count-two";
        "وون كوريا الجنوبية",
        "displayName-count-few";
        "وون كوريا الجنوبية",
        "displayName-count-many";
        "وون كوريا الجنوبية",
        "displayName-count-other";
        "وون كوريا الجنوبية",
        "symbol";
        "₩",
        "symbol-alt-narrow";
        "₩";
    }
    "KWD";
    {
        "displayName";
        "دينار كويتي",
        "displayName-count-zero";
        "دينار كويتي",
        "displayName-count-one";
        "دينار كويتي",
        "displayName-count-two";
        "دينار كويتي",
        "displayName-count-few";
        "دينار كويتي",
        "displayName-count-many";
        "دينار كويتي",
        "displayName-count-other";
        "دينار كويتي",
        "symbol";
        "د.ك.";
    }
    "KYD";
    {
        "displayName";
        "دولار جزر كيمن",
        "displayName-count-zero";
        "دولار جزر كيمن",
        "displayName-count-one";
        "دولار جزر كيمن",
        "displayName-count-two";
        "دولار جزر كيمن",
        "displayName-count-few";
        "دولار جزر كيمن",
        "displayName-count-many";
        "دولار جزر كيمن",
        "displayName-count-other";
        "دولار جزر كيمن",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
    }

```

```

        "KY$";
    }
    "KZT";
    {
        "displayName";
        "تينغ كازاخستاني",
        "displayName-count-zero";
        "تينغ كازاخستاني",
        "displayName-count-one";
        "تينغ كازاخستاني",
        "displayName-count-two";
        "تينغ كازاخستاني",
        "displayName-count-few";
        "تينغ كازاخستاني",
        "displayName-count-many";
        "تينغ كازاخستاني",
        "displayName-count-other";
        "تينغ كازاخستاني",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "Т";
    }
    "LAK";
    {
        "displayName";
        "كيب لاوسي",
        "displayName-count-zero";
        "كيب لاوسي",
        "displayName-count-one";
        "كيب لاوسي",
        "displayName-count-two";
        "كيب لاوسي",
        "displayName-count-few";
        "كيب لاوسي",
        "displayName-count-many";
        "كيب لاوسي",
        "displayName-count-other";
        "كيب لاوسي",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "جنيه لبناني",
        "displayName-count-zero";
        "جنيه لبناني",
        "displayName-count-one";
        "جنيه لبناني",
        "displayName-count-two";
        "جنيه لبناني",
        "displayName-count-few";
        "جنيه لبناني",
        "displayName-count-many";
    }

```

```

        "جنيه لبناني",
        "displayName-count-other";
        "جنيه لبناني",
        "symbol";
        ".ل.ل",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";
        "روبية سريلانكية",
        "displayName-count-zero";
        "روبية سريلانكية",
        "displayName-count-one";
        "روبية سريلانكية",
        "displayName-count-two";
        "روبية سريلانكية",
        "displayName-count-few";
        "روبية سريلانكية",
        "displayName-count-many";
        "روبية سريلانكية",
        "displayName-count-other";
        "روبية سريلانكية",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "LRD";
    {
        "displayName";
        "دولار ليبيري",
        "displayName-count-zero";
        "دولار ليبيري",
        "displayName-count-one";
        "دولار ليبيري",
        "displayName-count-two";
        "دولاران ليبيريان",
        "displayName-count-few";
        "دولارات ليبيرية",
        "displayName-count-many";
        "دولارًا ليبيريًا",
        "displayName-count-other";
        "دولار ليبيري",
        "symbol";
        "LRD",
        "symbol-alt-narrow";
        "$";
    }
    "LSL";
    {
        "displayName";
        "لوتي ليسوتو",
        "symbol";
        "LSL";
    }
}

```

```

"LTL";
{
  "displayName";
  "ليتا ليتوانية",
  "displayName-count-zero";
  "ليتا ليتوانية",
  "displayName-count-one";
  "ليتا ليتوانية",
  "displayName-count-two";
  "ليتا ليتوانية",
  "displayName-count-few";
  "ليتا ليتوانية",
  "displayName-count-many";
  "ليتا ليتوانية",
  "displayName-count-other";
  "ليتا ليتوانية",
  "symbol";
  "LTL",
  "symbol-alt-narrow";
  "Lt";
}
"LTT";
{
  "displayName";
  "تالوناس ليتواني",
  "symbol";
  "LTT";
}
"LUC";
{
  "displayName";
  "فرنك لوكسمبرج قابل للتحويل",
  "symbol";
  "LUC";
}
"LUF";
{
  "displayName";
  "فرنك لوكسمبرج",
  "symbol";
  "LUF";
}
"LUL";
{
  "displayName";
  "فرنك لوكسمبرج المالي",
  "symbol";
  "LUL";
}
"LVL";
{
  "displayName";
  "لاتس لاتفيا",
  "displayName-count-zero";
  "لاتس لاتفي",
  "displayName-count-one";
  "لاتس لاتفي",

```

```

        "displayName-count-two";
        "لاتس لاتفي",
        "displayName-count-few";
        "لاتس لاتفي",
        "displayName-count-many";
        "لاتس لاتفي",
        "displayName-count-other";
        "لاتس لاتفي",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "روبل لاتفيا",
        "symbol";
        "LVR";
    }
    "LYD";
    {
        "displayName";
        "دينار ليبي",
        "displayName-count-zero";
        "دينار ليبي",
        "displayName-count-one";
        "دينار ليبي",
        "displayName-count-two";
        "ديناران ليبيان",
        "displayName-count-few";
        "دينارات ليبية",
        "displayName-count-many";
        "دينارًا ليبيا",
        "displayName-count-other";
        "دينار ليبي",
        "symbol";
        "د.ل.";
    }
    "MAD";
    {
        "displayName";
        "درهم مغربي",
        "displayName-count-zero";
        "درهم مغربي",
        "displayName-count-one";
        "درهم مغربي",
        "displayName-count-two";
        "درهمان مغربيان",
        "displayName-count-few";
        "دراهم مغربية",
        "displayName-count-many";
        "درهمًا مغربيًا",
        "displayName-count-other";
        "درهم مغربي",
        "symbol";
        "د.م.";
    }

```

```

    }
    "MAF";
    {
        "displayName";
        "فرنك مغربي",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "MCF",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "MDC",
        "symbol";
        "MDC";
    }
    "MDL";
    {
        "displayName";
        "ليو مولدوفي",
        "displayName-count-zero";
        "ليو مولدوفي",
        "displayName-count-one";
        "ليو مولدوفي",
        "displayName-count-two";
        "ليو مولدوفي",
        "displayName-count-few";
        "ليو مولدوفي",
        "displayName-count-many";
        "ليو مولدوفي",
        "displayName-count-other";
        "ليو مولدوفي",
        "symbol";
        "MDL";
    }
    "MGA";
    {
        "displayName";
        "أرياري مدغشقر",
        "displayName-count-zero";
        "أرياري مدغشقر",
        "displayName-count-one";
        "أرياري مدغشقر",
        "displayName-count-two";
        "أرياري مدغشقر",
        "displayName-count-few";
        "أرياري مدغشقر",
        "displayName-count-many";
        "أرياري مدغشقر",
        "displayName-count-other";
        "أرياري مدغشقر",
    }

```

```

        "symbol";
        "MGA",
        "symbol-alt-narrow";
        "Ar";
    }
    "MGF";
    {
        "displayName";
        "فرنك مدغشقر",
        "symbol";
        "MGF";
    }
    "MKD";
    {
        "displayName";
        "دينار مقدوني",
        "displayName-count-zero";
        "دينار مقدوني",
        "displayName-count-one";
        "دينار مقدوني",
        "displayName-count-two";
        "ديناران مقدونيان",
        "displayName-count-few";
        "دينارات مقدونية",
        "displayName-count-many";
        "دينارًا مقدونيًا",
        "displayName-count-other";
        "دينار مقدوني",
        "symbol";
        "MKD";
    }
    "MKN";
    {
        "displayName";
        "MKN",
        "symbol";
        "MKN";
    }
    "MLF";
    {
        "displayName";
        "فرنك مالي",
        "symbol";
        "MLF";
    }
    "MMK";
    {
        "displayName";
        "كيات ميانمار",
        "displayName-count-zero";
        "كيات ميانمار",
        "displayName-count-one";
        "كيات ميانمار",
        "displayName-count-two";
        "كيات ميانمار",
        "displayName-count-few";
        "كيات ميانمار",
    }

```

```

        "displayName-count-many";
        "کیات میانمار",
        "displayName-count-other";
        "کیات میانمار",
        "symbol";
        "MMK",
        "symbol-alt-narrow";
        "K";
    }
    "MNT";
    {
        "displayName";
        "توغروغ منغولي",
        "displayName-count-zero";
        "توغروغ منغولي",
        "displayName-count-one";
        "توغروغ منغولي",
        "displayName-count-two";
        "توغروغ منغولي",
        "displayName-count-few";
        "توغروغ منغولي",
        "displayName-count-many";
        "توغروغ منغولي",
        "displayName-count-other";
        "توغروغ منغولي",
        "symbol";
        "MNT",
        "symbol-alt-narrow";
        "₮";
    }
    "MOP";
    {
        "displayName";
        "باتاکا ماکاوي",
        "displayName-count-zero";
        "باتاکا ماکاوي",
        "displayName-count-one";
        "باتاکا ماکاوي",
        "displayName-count-two";
        "باتاکا ماکاوي",
        "displayName-count-few";
        "باتاکا ماکاوي",
        "displayName-count-many";
        "باتاکا ماکاوي",
        "displayName-count-other";
        "باتاکا ماکاوي",
        "symbol";
        "MOP";
    }
    "MRO";
    {
        "displayName";
        "أوقية موريتانية",
        "displayName-count-zero";
        "أوقية موريتانية",
        "displayName-count-one";
        "أوقية موريتانية",

```



```

        "displayName-count-two";
        "أوقية موريتانية",
        "displayName-count-few";
        "أوقية موريتانية",
        "displayName-count-many";
        "أوقية موريتانية",
        "displayName-count-other";
        "أوقية موريتانية",
        "symbol";
        "أ.م.";
    }
    "MTL";
    {
        "displayName";
        "ليرة مالطية",
        "symbol";
        "MTL";
    }
    "MTP";
    {
        "displayName";
        "جنيه مالطي",
        "symbol";
        "MTP";
    }
    "MUR";
    {
        "displayName";
        "روبية موريشيوسية",
        "displayName-count-zero";
        "روبية موريشيوسية",
        "displayName-count-one";
        "روبية موريشيوسية",
        "displayName-count-two";
        "روبية موريشيوسية",
        "displayName-count-few";
        "روبية موريشيوسية",
        "displayName-count-many";
        "روبية موريشيوسية",
        "displayName-count-other";
        "روبية موريشيوسية",
        "symbol";
        "MUR",
        "symbol-alt-narrow";
        "Rs";
    }
    "MVP";
    {
        "displayName";
        "MVP",
        "symbol";
        "MVP";
    }
    "MVR";
    {
        "displayName";
        "روفية جزر المالديف",

```

```

        "displayName-count-zero";
        "روفيه جزر المالديف",
        "displayName-count-one";
        "روفيه جزر المالديف",
        "displayName-count-two";
        "روفيه جزر المالديف",
        "displayName-count-few";
        "روفيه جزر المالديف",
        "displayName-count-many";
        "روفيه جزر المالديف",
        "displayName-count-other";
        "روفيه جزر المالديف",
        "symbol";
        "MVR";
    }
    "MWK";
    {
        "displayName";
        "كواشا مالاوي",
        "displayName-count-zero";
        "كواشا مالاوي",
        "displayName-count-one";
        "كواشا مالاوي",
        "displayName-count-two";
        "كواشا مالاوي",
        "displayName-count-few";
        "كواشا مالاوي",
        "displayName-count-many";
        "كواشا مالاوي",
        "displayName-count-other";
        "كواشا مالاوي",
        "symbol";
        "MWK";
    }
    "MXN";
    {
        "displayName";
        "بيزو مكسيكي",
        "displayName-count-zero";
        "بيزو مكسيكي",
        "displayName-count-one";
        "بيزو مكسيكي",
        "displayName-count-two";
        "بيزو مكسيكي",
        "displayName-count-few";
        "بيزو مكسيكي",
        "displayName-count-many";
        "بيزو مكسيكي",
        "displayName-count-other";
        "بيزو مكسيكي",
        "symbol";
        "MX$",
        "symbol-alt-narrow";
        "MX$";
    }
    "MXP";
    {

```

```

        "displayName";
        "1992-1861 - بيزو فضي مكسيكي",
        "symbol";
        "MXP";
    }
    "MXV";
    {
        "displayName";
        "MXV",
        "symbol";
        "MXV";
    }
    "MYR";
    {
        "displayName";
        "رينغيت ماليزي",
        "displayName-count-zero";
        "رينغيت ماليزي",
        "displayName-count-one";
        "رينغيت ماليزي",
        "displayName-count-two";
        "رينغيت ماليزي",
        "displayName-count-few";
        "رينغيت ماليزي",
        "displayName-count-many";
        "رينغيت ماليزي",
        "displayName-count-other";
        "رينغيت ماليزي",
        "symbol";
        "MYR",
        "symbol-alt-narrow";
        "RM";
    }
    "MZE";
    {
        "displayName";
        "اسكود موزمبيقى",
        "symbol";
        "MZE";
    }
    "MZM";
    {
        "displayName";
        "MZM",
        "symbol";
        "MZM";
    }
    "MZN";
    {
        "displayName";
        "متكال موزمبيقى",
        "displayName-count-zero";
        "متكال موزمبيقى",
        "displayName-count-one";
        "متكال موزمبيقى",
        "displayName-count-two";
        "متكال موزمبيقى";
    }

```

```

        "displayName-count-few";
        "متكال موزمبيقى",
        "displayName-count-many";
        "متكال موزمبيقى",
        "displayName-count-other";
        "متكال موزمبيقى",
        "symbol";
        "MZN";
    }
    "NAD";
    {
        "displayName";
        "دولار نامىبى",
        "displayName-count-zero";
        "دولار نامىبى",
        "displayName-count-one";
        "دولار نامىبى",
        "displayName-count-two";
        "دولار نامىبى",
        "displayName-count-few";
        "دولار نامىبى",
        "displayName-count-many";
        "دولار نامىبى",
        "displayName-count-other";
        "دولار نامىبى",
        "symbol";
        "NAD",
        "symbol-alt-narrow";
        "$";
    }
    "NGN";
    {
        "displayName";
        "نايرا نيجيرى",
        "displayName-count-zero";
        "نايرا نيجيرى",
        "displayName-count-one";
        "نايرا نيجيرى",
        "displayName-count-two";
        "نايرا نيجيرى",
        "displayName-count-few";
        "نايرا نيجيرى",
        "displayName-count-many";
        "نايرا نيجيرى",
        "displayName-count-other";
        "نايرا نيجيرى",
        "symbol";
        "NGN",
        "symbol-alt-narrow";
        "₦";
    }
    "NIC";
    {
        "displayName";
        "كوردوبه نيكاراچوا",
        "symbol";
        "NIC";
    }

```

```

    }
    "NIO";
    {
        "displayName";
        "قرطبة نيكاراغوا",
        "displayName-count-zero";
        "قرطبة نيكاراغوا",
        "displayName-count-one";
        "قرطبة نيكاراغوا",
        "displayName-count-two";
        "قرطبة نيكاراغوا",
        "displayName-count-few";
        "قرطبة نيكاراغوا",
        "displayName-count-many";
        "قرطبة نيكاراغوا",
        "displayName-count-other";
        "قرطبة نيكاراغوا",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";
        "جلدر هولندي",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "كرونة نرويجية",
        "displayName-count-zero";
        "كرونة نرويجية",
        "displayName-count-one";
        "كرونة نرويجية",
        "displayName-count-two";
        "كرونة نرويجية",
        "displayName-count-few";
        "كرونة نرويجية",
        "displayName-count-many";
        "كرونة نرويجية",
        "displayName-count-other";
        "كرونة نرويجية",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "روبية نيبالي",
        "displayName-count-zero";
        "روبية نيبالي",
        "displayName-count-one";
    }

```

```

        "روبية نيبالي",
        "displayName-count-two";
        "روبية نيبالي",
        "displayName-count-few";
        "روبية نيبالي",
        "displayName-count-many";
        "روبية نيبالي",
        "displayName-count-other";
        "روبية نيبالي",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {
        "displayName";
        "دولار نيوزيلندي",
        "displayName-count-zero";
        "دولار نيوزيلندي",
        "displayName-count-one";
        "دولار نيوزيلندي",
        "displayName-count-two";
        "دولار نيوزيلندي",
        "displayName-count-few";
        "دولار نيوزيلندي",
        "displayName-count-many";
        "دولار نيوزيلندي",
        "displayName-count-other";
        "دولار نيوزيلندي",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
        "NZ$";
    }
    "OMR";
    {
        "displayName";
        "رول عماني",
        "displayName-count-zero";
        "رول عماني",
        "displayName-count-one";
        "رول عماني",
        "displayName-count-two";
        "رول عماني",
        "displayName-count-few";
        "رول عماني",
        "displayName-count-many";
        "رول عماني",
        "displayName-count-other";
        "رول عماني",
        "symbol";
        "ر.ع.";
    }
    "PAB";
    {
        "displayName";

```

```

        "بالبوا بنمي",
        "displayName-count-zero";
        "بالبوا بنمي",
        "displayName-count-one";
        "بالبوا بنمي",
        "displayName-count-two";
        "بالبوا بنمي",
        "displayName-count-few";
        "بالبوا بنمي",
        "displayName-count-many";
        "بالبوا بنمي",
        "displayName-count-other";
        "بالبوا بنمي",
        "symbol";
        "PAB";
    }
    "PEI";
    {
        "displayName";
        "PEI",
        "symbol";
        "PEI";
    }
    "PEN";
    {
        "displayName";
        "سول بيروفي",
        "displayName-count-zero";
        "سول بيروفي",
        "displayName-count-one";
        "سول بيروفي",
        "displayName-count-two";
        "سول بيروفي",
        "displayName-count-few";
        "سول بيروفي",
        "displayName-count-many";
        "سول بيروفي",
        "displayName-count-other";
        "سول بيروفي",
        "symbol";
        "PEN";
    }
    "PES";
    {
        "displayName";
        "PES",
        "symbol";
        "PES";
    }
    "PGK";
    {
        "displayName";
        "كينا بابوا غينيا الجديدة",
        "displayName-count-zero";
        "كينا بابوا غينيا الجديدة",
        "displayName-count-one";
        "كينا بابوا غينيا الجديدة",

```

```

        "displayName-count-two";
        "كينا بابوا غينيا الجديدة",
        "displayName-count-few";
        "كينا بابوا غينيا الجديدة",
        "displayName-count-many";
        "كينا بابوا غينيا الجديدة",
        "displayName-count-other";
        "كينا بابوا غينيا الجديدة",
        "symbol";
        "PGK";
    }
    "PHP";
    {
        "displayName";
        "بيزو فلبيني",
        "displayName-count-zero";
        "بيزو فلبيني",
        "displayName-count-one";
        "بيزو فلبيني",
        "displayName-count-two";
        "بيزو فلبيني",
        "displayName-count-few";
        "بيزو فلبيني",
        "displayName-count-many";
        "بيزو فلبيني",
        "displayName-count-other";
        "بيزو فلبيني",
        "symbol";
        "PHP",
        "symbol-alt-narrow";
        "₱";
    }
    "PKR";
    {
        "displayName";
        "روبية باكستاني",
        "displayName-count-zero";
        "روبية باكستاني",
        "displayName-count-one";
        "روبية باكستاني",
        "displayName-count-two";
        "روبية باكستاني",
        "displayName-count-few";
        "روبية باكستاني",
        "displayName-count-many";
        "روبية باكستاني",
        "displayName-count-other";
        "روبية باكستاني",
        "symbol";
        "PKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "PLN";
    {
        "displayName";
        "زلوتي بولندي",

```



```

        "displayName-count-zero";
        "زلوتي بولندي",
        "displayName-count-one";
        "زلوتي بولندي",
        "displayName-count-two";
        "زلوتي بولندي",
        "displayName-count-few";
        "زلوتي بولندي",
        "displayName-count-many";
        "زلوتي بولندي",
        "displayName-count-other";
        "زلوتي بولندي",
        "symbol";
        "PLN",
        "symbol-alt-narrow";
        "zł";
    }
    "PLZ";
    {
        "displayName";
        "زلوتي بولندي - 1950-1995",
        "symbol";
        "PLZ";
    }
    "PTE";
    {
        "displayName";
        "اسكود برتغالي",
        "symbol";
        "PTE";
    }
    "PYG";
    {
        "displayName";
        "غواراني باراغواي",
        "displayName-count-zero";
        "غواراني باراغواي",
        "displayName-count-one";
        "غواراني باراغواي",
        "displayName-count-two";
        "غواراني باراغواي",
        "displayName-count-few";
        "غواراني باراغواي",
        "displayName-count-many";
        "غواراني باراغواي",
        "displayName-count-other";
        "غواراني باراغواي",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "راند قطري",
        "displayName-count-zero";

```

```

        "واڤ قطري",
        "displayName-count-one";
        "واڤ قطري",
        "displayName-count-two";
        "واڤ قطري",
        "displayName-count-few";
        "واڤ قطري",
        "displayName-count-many";
        "واڤ قطري",
        "displayName-count-other";
        "واڤ قطري",
        "symbol";
        "ر.ق.";
    }
    "RHD";
    {
        "displayName";
        "دولار روڊيسي",
        "symbol";
        "RHD";
    }
    "ROL";
    {
        "displayName";
        "ليو روماني قديم",
        "symbol";
        "ROL";
    }
    "RON";
    {
        "displayName";
        "ليو روماني",
        "displayName-count-zero";
        "ليو روماني",
        "displayName-count-one";
        "ليو روماني",
        "displayName-count-two";
        "ليو روماني",
        "displayName-count-few";
        "ليو روماني",
        "displayName-count-many";
        "ليو روماني",
        "displayName-count-other";
        "ليو روماني",
        "symbol";
        "RON",
        "symbol-alt-narrow";
        "lei";
    }
    "RSD";
    {
        "displayName";
        "دينار صربي",
        "displayName-count-zero";
        "دينار صربي",
        "displayName-count-one";
        "دينار صربي",
    }

```

```

        "displayName-count-two";
        "ديناران صربيان",
        "displayName-count-few";
        "دينارات صربية",
        "displayName-count-many";
        "دينارًا صربيًا",
        "displayName-count-other";
        "دينار صربي",
        "symbol";
        "RSD";
    }
    "RUB";
    {
        "displayName";
        "روبل روسي",
        "displayName-count-zero";
        "روبل روسي",
        "displayName-count-one";
        "روبل روسي",
        "displayName-count-two";
        "روبل روسي",
        "displayName-count-few";
        "روبل روسي",
        "displayName-count-many";
        "روبل روسي",
        "displayName-count-other";
        "روبل روسي",
        "symbol";
        "RUB",
        "symbol-alt-narrow";
        "₽";
    }
    "RUR";
    {
        "displayName";
        "1998-1991 - روبل روسي",
        "symbol";
        "RUR",
        "symbol-alt-narrow";
        "p.";
    }
    "RWF";
    {
        "displayName";
        "فرنك رواندي",
        "displayName-count-zero";
        "فرنك رواندي",
        "displayName-count-one";
        "فرنك رواندي",
        "displayName-count-two";
        "فرنك رواندي",
        "displayName-count-few";
        "فرنك رواندي",
        "displayName-count-many";
        "فرنك رواندي",
        "displayName-count-other";
        "فرنك رواندي",
    }

```

```

        "symbol";
        "RWF",
        "symbol-alt-narrow";
        "RF";
    }
    "SAR";
    {
        "displayName";
        "رول سعودي",
        "displayName-count-zero";
        "رول سعودي",
        "displayName-count-one";
        "رول سعودي",
        "displayName-count-two";
        "رول سعودي",
        "displayName-count-few";
        "رول سعودي",
        "displayName-count-many";
        "رول سعودي",
        "displayName-count-other";
        "رول سعودي",
        "symbol";
        "ر.س.";
    }
    "SBD";
    {
        "displayName";
        "دولار جزر سليمان",
        "displayName-count-zero";
        "دولار جزر سليمان",
        "displayName-count-one";
        "دولار جزر سليمان",
        "displayName-count-two";
        "دولار جزر سليمان",
        "displayName-count-few";
        "دولار جزر سليمان",
        "displayName-count-many";
        "دولار جزر سليمان",
        "displayName-count-other";
        "دولار جزر سليمان",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "SB$";
    }
    "SCR";
    {
        "displayName";
        "روبية شيشلية",
        "displayName-count-zero";
        "روبية شيشلية",
        "displayName-count-one";
        "روبية شيشلية",
        "displayName-count-two";
        "روبية شيشلية",
        "displayName-count-few";
        "روبية شيشلية",
    }

```

```

        "displayName-count-many";
        "روبية سيشيلية",
        "displayName-count-other";
        "روبية سيشيلية",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";
        "دينار سوداني",
        "symbol";
        "د.س.";
    }
    "SDG";
    {
        "displayName";
        "جنيه سوداني",
        "displayName-count-zero";
        "جنيه سوداني",
        "displayName-count-one";
        "جنيه سوداني",
        "displayName-count-two";
        "جنيه سوداني",
        "displayName-count-few";
        "جنيهات سودانية",
        "displayName-count-many";
        "جنيهاً سودانياً",
        "displayName-count-other";
        "جنيه سوداني",
        "symbol";
        "ج.س.";
    }
    "SDP";
    {
        "displayName";
        "جنيه سوداني قديم",
        "symbol";
        "SDP";
    }
    "SEK";
    {
        "displayName";
        "كرونة سويدية",
        "displayName-count-zero";
        "كرونة سويدية",
        "displayName-count-one";
        "كرونة سويدية",
        "displayName-count-two";
        "كرونة سويدية",
        "displayName-count-few";
        "كرونة سويدية",
        "displayName-count-many";
        "كرونة سويدية",
        "displayName-count-other";
        "كرونة سويدية",
        "symbol";
    }

```

```

        "SEK",
        "symbol-alt-narrow";
        "kr";
    }
    "SGD";
    {
        "displayName";
        "دولار سنغافوري",
        "displayName-count-zero";
        "دولار سنغافوري",
        "displayName-count-one";
        "دولار سنغافوري",
        "displayName-count-two";
        "دولار سنغافوري",
        "displayName-count-few";
        "دولار سنغافوري",
        "displayName-count-many";
        "دولار سنغافوري",
        "displayName-count-other";
        "دولار سنغافوري",
        "symbol";
        "SGD",
        "symbol-alt-narrow";
        "$";
    }
    "SHP";
    {
        "displayName";
        "جنيه سانت هيلين",
        "displayName-count-zero";
        "جنيه سانت هيلين",
        "displayName-count-one";
        "جنيه سانت هيلين",
        "displayName-count-two";
        "جنيه سانت هيلين",
        "displayName-count-few";
        "جنيه سانت هيلين",
        "displayName-count-many";
        "جنيه سانت هيلين",
        "displayName-count-other";
        "جنيه سانت هيلين",
        "symbol";
        "SHP",
        "symbol-alt-narrow";
        "£";
    }
    "SIT";
    {
        "displayName";
        "تولار سلوفيني",
        "symbol";
        "SIT";
    }
    "SKK";
    {
        "displayName";
        "كرونة سلوفاكية",

```

```

        "symbol";
        "SKK";
    }
    "SLL";
    {
        "displayName";
        "ليون سيراليوني",
        "displayName-count-zero";
        "ليون سيراليوني",
        "displayName-count-one";
        "ليون سيراليوني",
        "displayName-count-two";
        "ليون سيراليوني",
        "displayName-count-few";
        "ليون سيراليوني",
        "displayName-count-many";
        "ليون سيراليوني",
        "displayName-count-other";
        "ليون سيراليوني",
        "symbol";
        "SLL";
    }
    "SOS";
    {
        "displayName";
        "شلن صومالي",
        "displayName-count-zero";
        "شلن صومالي",
        "displayName-count-one";
        "شلن صومالي",
        "displayName-count-two";
        "شلن صومالي",
        "displayName-count-few";
        "شلن صومالي",
        "displayName-count-many";
        "شلن صومالي",
        "displayName-count-other";
        "شلن صومالي",
        "symbol";
        "SOS";
    }
    "SRD";
    {
        "displayName";
        "دولار سورينامي",
        "displayName-count-zero";
        "دولار سورينامي",
        "displayName-count-one";
        "دولار سورينامي",
        "displayName-count-two";
        "دولار سورينامي",
        "displayName-count-few";
        "دولار سورينامي",
        "displayName-count-many";
        "دولار سورينامي",
        "displayName-count-other";
        "دولار سورينامي",
    }

```

```

        "symbol";
        "SRD",
        "symbol-alt-narrow";
        "SR$";
    }
    "SRG";
    {
        "displayName";
        "جلدر سورينامي",
        "symbol";
        "SRG";
    }
    "SSP";
    {
        "displayName";
        "جنيه جنوب السودان",
        "displayName-count-zero";
        "جنيه جنوب السودان",
        "displayName-count-one";
        "جنيه جنوب السودان",
        "displayName-count-two";
        "جنيهان جنوب السودان",
        "displayName-count-few";
        "جنيهات جنوب السودان",
        "displayName-count-many";
        "جنيها جنوب السودان",
        "displayName-count-other";
        "جنيه جنوب السودان",
        "symbol";
        "SSP",
        "symbol-alt-narrow";
        "£";
    }
    "STD";
    {
        "displayName";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-zero";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-one";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-two";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-few";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-many";
        "دوبرا ساو تومي وبرينسيبي",
        "displayName-count-other";
        "دوبرا ساو تومي وبرينسيبي",
        "symbol";
        "STD",
        "symbol-alt-narrow";
        "Db";
    }
    "STN";
    {
        "displayName";

```



```

        "STN",
        "symbol";
        "STN";
    }
    "SUR";
    {
        "displayName";
        "روبل سوفيتي",
        "symbol";
        "SUR";
    }
    "SVC";
    {
        "displayName";
        "كولون سلفادوري",
        "symbol";
        "SVC";
    }
    "SYP";
    {
        "displayName";
        "ليرة سورية",
        "displayName-count-zero";
        "ليرة سورية",
        "displayName-count-one";
        "ليرة سورية",
        "displayName-count-two";
        "ليرة سورية",
        "displayName-count-few";
        "ليرة سورية",
        "displayName-count-many";
        "ليرة سورية",
        "displayName-count-other";
        "ليرة سورية",
        "symbol";
        "ل.س.",
        "symbol-alt-narrow";
        "£";
    }
    "SZL";
    {
        "displayName";
        "ليلانجيني سوازيلندي",
        "displayName-count-zero";
        "ليلانجيني سوازيلندي",
        "displayName-count-one";
        "ليلانجيني سوازيلندي",
        "displayName-count-two";
        "ليلانجيني سوازيلندي",
        "displayName-count-few";
        "ليلانجيني سوازيلندي",
        "displayName-count-many";
        "ليلانجيني سوازيلندي",
        "displayName-count-other";
        "ليلانجيني سوازيلندي",
        "symbol";
        "SZL";
    }

```

```

    }
    "THB";
    {
        "displayName";
        "باخت تایلاندي",
        "displayName-count-zero";
        "باخت تایلاندي",
        "displayName-count-one";
        "باخت تایلاندي",
        "displayName-count-two";
        "باخت تایلاندي",
        "displayName-count-few";
        "باخت تایلاندي",
        "displayName-count-many";
        "باخت تایلاندي",
        "displayName-count-other";
        "باخت تایلاندي",
        "symbol";
        "฿",
        "symbol-alt-narrow";
        "฿";
    }
    "TJR";
    {
        "displayName";
        "روبل طاجیکستانی",
        "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "سومونی طاجیکستانی",
        "displayName-count-zero";
        "سومونی طاجیکستانی",
        "displayName-count-one";
        "سومونی طاجیکستانی",
        "displayName-count-two";
        "سومونی طاجیکستانی",
        "displayName-count-few";
        "سومونی طاجیکستانی",
        "displayName-count-many";
        "سومونی طاجیکستانی",
        "displayName-count-other";
        "سومونی طاجیکستانی",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "مانات ترکمنستانی",
        "symbol";
        "TMM";
    }
    "TMT";
    {

```

```

        "displayName";
        "مانات ترکمانستان",
        "displayName-count-zero";
        "مانات ترکمانستان",
        "displayName-count-one";
        "مانات ترکمانستان",
        "displayName-count-two";
        "مانات ترکمانستان",
        "displayName-count-few";
        "مانات ترکمانستان",
        "displayName-count-many";
        "مانات ترکمانستان",
        "displayName-count-other";
        "مانات ترکمانستان",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "دينار تونسي",
        "displayName-count-zero";
        "دينار تونسي",
        "displayName-count-one";
        "دينار تونسي",
        "displayName-count-two";
        "ديناران تونسيان",
        "displayName-count-few";
        "دينارات تونسية",
        "displayName-count-many";
        "دينارًا تونسيًا",
        "displayName-count-other";
        "دينار تونسي",
        "symbol";
        "د.ت.";
    }
    "TOP";
    {
        "displayName";
        "بانغا تونغا",
        "displayName-count-zero";
        "بانغا تونغا",
        "displayName-count-one";
        "بانغا تونغا",
        "displayName-count-two";
        "بانغا تونغا",
        "displayName-count-few";
        "بانغا تونغا",
        "displayName-count-many";
        "بانغا تونغا",
        "displayName-count-other";
        "بانغا تونغا",
        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
}

```

```

    "TPE";
    {
      "displayName";
      "اسكود تيموري",
      "symbol";
      "TPE";
    }
    "TRL";
    {
      "displayName";
      "ليرة تركي",
      "symbol";
      "TRL";
    }
    "TRY";
    {
      "displayName";
      "ليرة تركية",
      "displayName-count-zero";
      "ليرة تركية",
      "displayName-count-one";
      "ليرة تركية",
      "displayName-count-two";
      "ليرة تركية",
      "displayName-count-few";
      "ليرة تركية",
      "displayName-count-many";
      "ليرة تركية",
      "displayName-count-other";
      "ليرة تركية",
      "symbol";
      "TRY",
      "symbol-alt-narrow";
      "₺",
      "symbol-alt-variant";
      "TL";
    }
    "TTD";
    {
      "displayName";
      "دولار ترينداد وتوباغو",
      "displayName-count-zero";
      "دولار ترينداد وتوباغو",
      "displayName-count-one";
      "دولار ترينداد وتوباغو",
      "displayName-count-two";
      "دولار ترينداد وتوباغو",
      "displayName-count-few";
      "دولار ترينداد وتوباغو",
      "displayName-count-many";
      "دولار ترينداد وتوباغو",
      "displayName-count-other";
      "دولار ترينداد وتوباغو",
      "symbol";
      "TTD",
      "symbol-alt-narrow";
      "TT$";
    }

```

```

    }
    "TWD";
    {
        "displayName";
        "دولار تایوانی",
        "displayName-count-zero";
        "دولار تایوانی",
        "displayName-count-one";
        "دولار تایوانی",
        "displayName-count-two";
        "دولار تایوانی",
        "displayName-count-few";
        "دولار تایوانی",
        "displayName-count-many";
        "دولار تایوانی",
        "displayName-count-other";
        "دولار تایوانی",
        "symbol";
        "NT$";
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";
    {
        "displayName";
        "شلن تنزانی",
        "displayName-count-zero";
        "شلن تنزانی",
        "displayName-count-one";
        "شلن تنزانی",
        "displayName-count-two";
        "شلن تنزانی",
        "displayName-count-few";
        "شلن تنزانی",
        "displayName-count-many";
        "شلن تنزانی",
        "displayName-count-other";
        "شلن تنزانی",
        "symbol";
        "TZS";
    }
    "UAH";
    {
        "displayName";
        "هریفنیا اوکرانی",
        "displayName-count-zero";
        "هریفنیا اوکرانی",
        "displayName-count-one";
        "هریفنیا اوکرانی",
        "displayName-count-two";
        "هریفنیا اوکرانی",
        "displayName-count-few";
        "هریفنیا اوکرانی",
        "displayName-count-many";
        "هریفنیا اوکرانی",
        "displayName-count-other";
        "هریفنیا اوکرانی",
    }

```

```

        "symbol";
        "UAH",
        "symbol-alt-narrow";
        "₴";
    }
    "UAK";
    {
        "displayName";
        "UAK",
        "symbol";
        "UAK";
    }
    "UGS";
    {
        "displayName";
        "شلن أوغندي - 1966-1987",
        "symbol";
        "UGS";
    }
    "UGX";
    {
        "displayName";
        "شلن أوغندي",
        "displayName-count-zero";
        "شلن أوغندي",
        "displayName-count-one";
        "شلن أوغندي",
        "displayName-count-two";
        "شلن أوغندي",
        "displayName-count-few";
        "شلن أوغندي",
        "displayName-count-many";
        "شلن أوغندي",
        "displayName-count-other";
        "شلن أوغندي",
        "symbol";
        "UGX";
    }
    "USD";
    {
        "displayName";
        "دولار أمريكي",
        "displayName-count-zero";
        "دولار أمريكي",
        "displayName-count-one";
        "دولار أمريكي",
        "displayName-count-two";
        "دولار أمريكي",
        "displayName-count-few";
        "دولار أمريكي",
        "displayName-count-many";
        "دولار أمريكي",
        "displayName-count-other";
        "دولار أمريكي",
        "symbol";
        "US$",
        "symbol-alt-narrow";
    }

```

```

        "US$";
    }
    "USN";
    {
        "displayName";
        "دولار أمريكي (اليوم التالي)",
        "symbol";
        "USN";
    }
    "USS";
    {
        "displayName";
        "دولار أمريكي (نفس اليوم)",
        "symbol";
        "USS";
    }
    "UYI";
    {
        "displayName";
        "UYI",
        "symbol";
        "UYI";
    }
    "UYP";
    {
        "displayName";
        "بيزو أوروغواي - 1975-1993",
        "symbol";
        "UYP";
    }
    "UYU";
    {
        "displayName";
        "بيزو اوروغواي",
        "displayName-count-zero";
        "بيزو اوروغواي",
        "displayName-count-one";
        "بيزو اوروغواي",
        "displayName-count-two";
        "بيزو اوروغواي",
        "displayName-count-few";
        "بيزو اوروغواي",
        "displayName-count-many";
        "بيزو اوروغواي",
        "displayName-count-other";
        "بيزو اوروغواي",
        "symbol";
        "UYU",
        "symbol-alt-narrow";
        "UY$";
    }
    "UZS";
    {
        "displayName";
        "سوم أوزبكستاني",
        "displayName-count-zero";
        "سوم أوزبكستاني",

```

```

        "displayName-count-one";
        "سوم أوزبكستاني",
        "displayName-count-two";
        "سوم أوزبكستاني",
        "displayName-count-few";
        "سوم أوزبكستاني",
        "displayName-count-many";
        "سوم أوزبكستاني",
        "displayName-count-other";
        "سوم أوزبكستاني",
        "symbol";
        "UZS";
    }
    "VEB";
    {
        "displayName";
        "بوليفار فنزويلي - 2008-1871",
        "symbol";
        "VEB";
    }
    "VEF";
    {
        "displayName";
        "بوليفار فنزويلي",
        "displayName-count-zero";
        "بوليفار فنزويلي",
        "displayName-count-one";
        "بوليفار فنزويلي",
        "displayName-count-two";
        "بوليفار فنزويلي",
        "displayName-count-few";
        "بوليفار فنزويلي",
        "displayName-count-many";
        "بوليفار فنزويلي",
        "displayName-count-other";
        "بوليفار فنزويلي",
        "symbol";
        "VEF",
        "symbol-alt-narrow";
        "Bs";
    }
    "VND";
    {
        "displayName";
        "دونج فيتنامي",
        "displayName-count-zero";
        "دونج فيتنامي",
        "displayName-count-one";
        "دونج فيتنامي",
        "displayName-count-two";
        "دونج فيتنامي",
        "displayName-count-few";
        "دونج فيتنامي",
        "displayName-count-many";
        "دونج فيتنامي",
        "displayName-count-other";
        "دونج فيتنامي",
    }

```



```

        "symbol";
        "₯",
        "symbol-alt-narrow";
        "₯";
    }
    "VNN";
    {
        "displayName";
        "VNN",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "فاتو فانواتو",
        "displayName-count-zero";
        "فاتو فانواتو",
        "displayName-count-one";
        "فاتو فانواتو",
        "displayName-count-two";
        "فاتو فانواتو",
        "displayName-count-few";
        "فاتو فانواتو",
        "displayName-count-many";
        "فاتو فانواتو",
        "displayName-count-other";
        "فاتو فانواتو",
        "symbol";
        "VUV";
    }
    "WST";
    {
        "displayName";
        "تالا ساموا",
        "displayName-count-zero";
        "تالا ساموا",
        "displayName-count-one";
        "تالا ساموا",
        "displayName-count-two";
        "تالا ساموا",
        "displayName-count-few";
        "تالا ساموا",
        "displayName-count-many";
        "تالا ساموا",
        "displayName-count-other";
        "تالا ساموا",
        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "فرنك وسط أفريقي",
        "displayName-count-zero";
        "فرنك وسط أفريقي",
        "displayName-count-one";
    }

```

```

        "فرنك وسط أفريقي",
        "displayName-count-two";
        "فرنك وسط أفريقي",
        "displayName-count-few";
        "فرنك وسط أفريقي",
        "displayName-count-many";
        "فرنك وسط أفريقي",
        "displayName-count-other";
        "فرنك وسط أفريقي",
        "symbol";
        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "فضة",
        "symbol";
        "XAG";
    }
    "XAU";
    {
        "displayName";
        "ذهب",
        "symbol";
        "XAU";
    }
    "XBA";
    {
        "displayName";
        "الوحدة الأوروبية المركبة",
        "symbol";
        "XBA";
    }
    "XBB";
    {
        "displayName";
        "الوحدة المالية الأوروبية",
        "symbol";
        "XBB";
    }
    "XBC";
    {
        "displayName";
        "الوحدة الحسابية الأوروبية",
        "symbol";
        "XBC";
    }
    "XBD";
    {
        "displayName";
        "وحدة الحساب الأوروبية (XBD)",
        "symbol";
        "XBD";
    }
    "XCD";
    {
        "displayName";

```

```

        "دولار شرق الكاريبي",
        "displayName-count-zero";
        "دولار شرق الكاريبي",
        "displayName-count-one";
        "دولار شرق الكاريبي",
        "displayName-count-two";
        "دولار شرق الكاريبي",
        "displayName-count-few";
        "دولار شرق الكاريبي",
        "displayName-count-many";
        "دولار شرق الكاريبي",
        "displayName-count-other";
        "دولار شرق الكاريبي",
        "symbol";
        "EC$";
        "symbol-alt-narrow";
        "$";
    }
    "XDR";
    {
        "displayName";
        "حقوق السحب الخاصة",
        "symbol";
        "XDR";
    }
    "XEU";
    {
        "displayName";
        "وحدة النقد الأوروبية",
        "symbol";
        "XEU";
    }
    "XFO";
    {
        "displayName";
        "فرنك فرنسي ذهبي",
        "symbol";
        "XFO";
    }
    "XFU";
    {
        "displayName";
        "فرنك فرنسي (UIC)",
        "symbol";
        "XFU";
    }
    "XOF";
    {
        "displayName";
        "فرنك غرب أفريقي",
        "displayName-count-zero";
        "فرنك غرب أفريقي",
        "displayName-count-one";
        "فرنك غرب أفريقي",
        "displayName-count-two";
        "فرنك غرب أفريقي",
        "displayName-count-few";
    }

```

```

        "فرنك غرب أفريقي",
        "displayName-count-many";
        "فرنك غرب أفريقي",
        "displayName-count-other";
        "فرنك غرب أفريقي",
        "symbol";
        "CFA";
    }
    "XPD";
    {
        "displayName";
        "بالاديوم",
        "symbol";
        "XPD";
    }
    "XPF";
    {
        "displayName";
        "فرنك سي إف بي",
        "displayName-count-zero";
        "فرنك سي إف بي",
        "displayName-count-one";
        "فرنك سي إف بي",
        "displayName-count-two";
        "فرنك سي إف بي",
        "displayName-count-few";
        "فرنك سي إف بي",
        "displayName-count-many";
        "فرنك سي إف بي",
        "displayName-count-other";
        "فرنك سي إف بي",
        "symbol";
        "CFPF";
    }
    "XPT";
    {
        "displayName";
        "البلاطين",
        "symbol";
        "XPT";
    }
    "XRE";
    {
        "displayName";
        "XRE",
        "symbol";
        "XRE";
    }
    "XSU";
    {
        "displayName";
        "XSU",
        "symbol";
        "XSU";
    }
    "XTS";
    {

```

```

        "displayName";
        "كود اختبار العملة",
        "symbol";
        "XTS";
    }
    "XUA";
    {
        "displayName";
        "XUA",
        "symbol";
        "XUA";
    }
    "XXX";
    {
        "displayName";
        "عملة غير معروفة",
        "displayName-count-zero";
        "(عملة غير معروفة)",
        "displayName-count-one";
        "(عملة غير معروفة)",
        "displayName-count-two";
        "(عملة غير معروفة)",
        "displayName-count-few";
        "(عملة غير معروفة)",
        "displayName-count-many";
        "(عملة غير معروفة)",
        "displayName-count-other";
        "(عملة غير معروفة)",
        "symbol";
        "****";
    }
    "YDD";
    {
        "displayName";
        "دينار يمني",
        "symbol";
        "YDD";
    }
    "YER";
    {
        "displayName";
        "ول يمني",
        "displayName-count-zero";
        "ول يمني",
        "displayName-count-one";
        "ول يمني",
        "displayName-count-two";
        "ول يمني",
        "displayName-count-few";
        "ول يمني",
        "displayName-count-many";
        "ول يمني",
        "displayName-count-other";
        "ول يمني",
        "symbol";
        "ر.ي.";
    }
}

```

```

        "YUD";
        {
            "displayName";
            "دينار يوغسلافي",
            "symbol";
            "YUD";
        }
        "YUM";
        {
            "displayName";
            "YUM",
            "symbol";
            "YUM";
        }
        "YUN";
        {
            "displayName";
            "دينار يوغسلافي قابل للتحويل",
            "symbol";
            "YUN";
        }
        "YUR";
        {
            "displayName";
            "YUR",
            "symbol";
            "YUR";
        }
        "ZAL";
        {
            "displayName";
            "راند جنوب أفريقيا -مالي",
            "symbol";
            "ZAL";
        }
        "ZAR";
        {
            "displayName";
            "راند جنوب أفريقيا",
            "displayName-count-zero";
            "راند جنوب أفريقيا",
            "displayName-count-one";
            "راند جنوب أفريقيا",
            "displayName-count-two";
            "راند جنوب أفريقيا",
            "displayName-count-few";
            "راند جنوب أفريقيا",
            "displayName-count-many";
            "راند جنوب أفريقيا",
            "displayName-count-other";
            "راند جنوب أفريقيا",
            "symbol";
            "ZAR",
            "symbol-alt-narrow";
            "R";
        }
        "ZMK";
    
```

```

    {
      "displayName";
      "2012-1968 - كواشا زامبي",
      "displayName-count-zero";
      "2012-1968 - كواشا زامبي",
      "displayName-count-one";
      "2012-1968 - كواشا زامبي",
      "displayName-count-two";
      "2012-1968 - كواشا زامبي",
      "displayName-count-few";
      "2012-1968 - كواشا زامبي",
      "displayName-count-many";
      "2012-1968 - كواشا زامبي",
      "displayName-count-other";
      "2012-1968 - كواشا زامبي",
      "symbol";
      "ZMK";
    }
    "ZMW";
    {
      "displayName";
      "كواشا زامبي",
      "displayName-count-zero";
      "كواشا زامبي",
      "displayName-count-one";
      "كواشا زامبي",
      "displayName-count-two";
      "كواشا زامبي",
      "displayName-count-few";
      "كواشا زامبي",
      "displayName-count-many";
      "كواشا زامبي",
      "displayName-count-other";
      "كواشا زامبي",
      "symbol";
      "ZMW",
      "symbol-alt-narrow";
      "ZK";
    }
    "ZRN";
    {
      "displayName";
      "زائير زائيري جديد",
      "symbol";
      "ZRN";
    }
    "ZRZ";
    {
      "displayName";
      "زائير زائيري",
      "symbol";
      "ZRZ";
    }
    "ZWD";
    {
      "displayName";
      "دولار زمبابوي",

```

```
        "symbol";
    }
    "ZWD";
}
{
    "displayName";
    "دولار زمبابوي 2009",
    "symbol";
    "ZWL";
}
"ZWR";
{
    "displayName";
    "ZWR",
    "symbol";
    "ZWR";
}
}
}
}
```

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
    'ar': {
        'datetimepicker': {
            placeholder: 'حدد التاريخ والوقت',
            today: 'اليوم'
        }
    }
});
class App extends React.Component {
    render() {
        return <DateTimePickerComponent id="datetimepicker" enableRTL={true}
        locale='ar' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as React from 'react';
```



```

import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
  'ar': {
    'datetimepicker': {
      placeholder: 'حدد التاريخ والوقت',
      today: 'اليوم'
    }
  }
});
class App extends React.Component<{}>, {}> {
  render() {
    return <DateTimePickerComponent id="datetimepicker" enableRtl={true}
    locale='ar' />
  }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

NUMBERINGSYSTEMS.JSON

```

{
  "supplemental": {
    "version": {
      "_number": "$Revision: 12732 $",
      "_unicodeVersion": "9.0.0",
      "_cldrVersion": "31"
    },
    "numberingSystems": {
      "adlm": {
        "_digits": "ᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩ",
        "_type": "numeric"
      },
      "ahom": {
        "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
        "_type": "numeric"
      },
      "arab": {
        "_digits": "٠١٢٣٤٥٦٧٨٩",
        "_type": "numeric"
      },
      "arabext": {
        "_digits": "٠١٢٣٤٥٦٧٨٩",
        "_type": "numeric"
      },
      "armn": {
        "_rules": "armenian-upper",
        "_type": "algorithmic"
      }
    }
  }
}

```

```

},
"armnlow": {
  "_rules": "armenian-lower",
  "_type": "algorithmic"
},
"bali": {
  "_digits": "ᬀᬁᬂᬃᬄᬅᬆᬇᬈᬉᬐᬑᬒᬓᬔᬕᬖᬗᬘᬙᬚᬛᬞᬟᬠᬡᬢᬣᬤᬥᬦᬧᬨᬩᬪᬫᬬᬭᬮᬯᬰᬱᬲᬳ᬴ᬵᬶᬷᬸᬹᬺᬻᬼᬽᬾᬿ",
  "_type": "numeric"
},
"beng": {
  "_digits": "০১২৩৪৫৬৭৮৯",
  "_type": "numeric"
},
"bhks": {
  "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱹᱯᱰᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type": "numeric"
},
"brah": {
  "_digits": "·᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱹᱯᱰᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type": "numeric"
},
"cakm": {
  "_digits": "ᨀᨁᨂᨃᨄᨅᨆᨇᨈᨉᨐᨑᨒᨓᨔᨕᨖᨘᨗᨙᨚᨛ᨜᨝᨞᨟ᨠᨡᨢᨣᨤᨥᨦᨧᨨᨩᨪᨫᨬᨭᨮᨯᨰᨱᨲᨳᨴᨵᨶᨷᨸᨹᨺᨻᨼᨽᨾᨿ",
  "_type": "numeric"
},
"cham": {
  "_digits": "ᬀᬁᬂᬃᬄᬅᬆᬇᬈᬉᬐᬑᬒᬓᬔᬕᬖᬗᬘᬙᬚᬛᬞᬟᬠᬡᬢᬣᬤᬥᬦᬧᬨᬩᬪᬫᬬᬭᬮᬯᬰᬱᬲᬳ᬴ᬵᬶᬷᬸᬹᬺᬻᬼᬽᬾᬿ",
  "_type": "numeric"
},
"cyrl": {
  "_rules": "cyrillic-lower",
  "_type": "algorithmic"
},
"deva": {
  "_digits": "०१२३४५६७८९",
  "_type": "numeric"
},
"ethi": {
  "_rules": "ethiopic",
  "_type": "algorithmic"
},
"fullwide": {
  "_digits": "0 1 2 3 4 5 6 7 8 9",
  "_type": "numeric"
},
"geor": {
  "_rules": "georgian",
  "_type": "algorithmic"
},
"grek": {
  "_rules": "greek-upper",
  "_type": "algorithmic"
},
"greklow": {
  "_rules": "greek-lower",
  "_type": "algorithmic"
}

```

```

    },
    "gujr": {
      "_digits": "૦૧૨૩૪૫૬૭૮૯",
      "_type": "numeric"
    },
    "guru": {
      "_digits": "੦੧੨੩੪੫੬੭੮੯",
      "_type": "numeric"
    },
    "hanidays": {
      "_rules": "zh/SpelloutRules/spellout-numbering-days",
      "_type": "algorithmic"
    },
    "hanidec": {
      "_digits": "〇一二三四五六七八九",
      "_type": "numeric"
    },
    "hans": {
      "_rules": "zh/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hansfin": {
      "_rules": "zh/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hant": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hantfin": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hebr": {
      "_rules": "hebrew",
      "_type": "algorithmic"
    },
    "hmng": {
      "_digits": "᠐᠑᠒᠓᠐ᠠᠨᠣᠭ",
      "_type": "numeric"
    },
    "java": {
      "_digits": "᠐ᠠᠨᠠᠵᠤᠯᠤᠰᠤᠨᠠᠵᠤᠯᠤᠰ",
      "_type": "numeric"
    },
    "jpan": {
      "_rules": "ja/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "jpanfin": {
      "_rules": "ja/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "kali": {
      "_digits": "᠐ᠠᠨᠠᠵᠤᠯᠤᠰᠤᠨᠠᠵᠤᠯᠤᠰ",

```

```

    "_type": "numeric"
  },
  "khmr": {
    "_digits": "០១២៣៤៥៦៧៨៩",
    "_type": "numeric"
  },
  "knda": {
    "_digits": "೦೧೨೩೪೫೬೭೮೯",
    "_type": "numeric"
  },
  "lana": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "lanatham": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "laoo": {
    "_digits": "໐໑໒໓໔໕໖໗໘໑",
    "_type": "numeric"
  },
  "latn": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "lepc": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "limb": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mathbold": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathdbl": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathmono": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsanb": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsans": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mlym": {

```

```

    "_digits": "൦൧൨൩൪൫൬൭൮൯",
    "_type": "numeric"
  },
  "modi": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mong": {
    "_digits": "᠐᠑᠒᠓᠐ᠠᠨᠨᠠᠭ",
    "_type": "numeric"
  },
  "mroo": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mtei": {
    "_digits": "ႤႬႬႬႬႬႬႬႬႬႬ",
    "_type": "numeric"
  },
  "mymr": {
    "_digits": "၀၁၂၃၄၅၆၇၈",
    "_type": "numeric"
  },
  "mymrshan": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mymrtlng": {
    "_digits": "0ႤႬႬႬႬႬႬႬႬႬ",
    "_type": "numeric"
  },
  "newa": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "nkoo": {
    "_digits": "ႦႧႨႩႪႫႬႭႮ",
    "_type": "numeric"
  },
  "olck": {
    "_digits": "0ႭႬႬႬႬႬႬႬႬႬ",
    "_type": "numeric"
  },
  "orya": {
    "_digits": "୦୧୨୩୪୫୬୭୮୯",
    "_type": "numeric"
  },
  "osma": {
    "_digits": "ႤႬႬႬႬႬႬႬႬႬ",
    "_type": "numeric"
  },
  "roman": {
    "_rules": "roman-upper",

```

```

    "_type": "algorithmic"
  },
  "romanlow": {
    "_rules": "roman-lower",
    "_type": "algorithmic"
  },
  "saur": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "shrd": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "sind": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "sinh": {
    "_digits": "ඒ෧෨෩෪෫෬෭෮෯",
    "_type": "numeric"
  },
  "sora": {
    "_digits": "୦୧୨୩୪୫୬୭୮୯",
    "_type": "numeric"
  },
  "sund": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "takr": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "talv": {
    "_digits": "೦೧೨೩೪೫೬೭೮೯",
    "_type": "numeric"
  },
  "taml": {
    "_rules": "tamil",
    "_type": "algorithmic"
  },
  "tamldec": {
    "_digits": "0௧௨௩௪௫௬௭௮௯",
    "_type": "numeric"
  },
  "telu": {
    "_digits": "౦౧౨౩౪౫౬౭౮౯",
    "_type": "numeric"
  },
  "thai": {
    "_digits": "๐๑๒๓๔๕๖๗๘๙",
    "_type": "numeric"
  },
  "tibet": {

```

```

    "_digits": "ႠႡႢႣႤႥႦႧ",
    "_type": "numeric"
  },
  "tirh": {
    "_digits": "ႠႡႢႣႤႥႦႧ",
    "_type": "numeric"
  },
  "vaih": {
    "_digits": "ႠႡႢႣႤႥႦႧ",
    "_type": "numeric"
  },
  "wara": {
    "_digits": "ႠႡႢႣႤႥႦႧ",
    "_type": "numeric"
  }
}
}
}

```

NUMBERINGSYSTEMS.JSX

```

{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
    "numberingSystems";
    {
      "adlm";
      {
        "_digits";
        "ႠႡႢႣႤႥႦႧ",
        "_type";
        "numeric";
      }
      "ahom";
      {
        "_digits";
        "ႠႡႢႣႤႥႦႧ",
        "_type";
        "numeric";
      }
      "arab";
      {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
      }
    }
  }
}

```

```
"arabext";
{
  "_digits";
  "٠١٢٣٤٥٦٧٨٩",
  "_type";
  "numeric";
}
"armn";
{
  "_rules";
  "armenian-upper",
  "_type";
  "algorithmic";
}
"armnlow";
{
  "_rules";
  "armenian-lower",
  "_type";
  "algorithmic";
}
"bali";
{
  "_digits";
  "ᮘᮙᮚᮛᮜᮝᮞᮟ",
  "_type";
  "numeric";
}
"beng";
{
  "_digits";
  "০১২৩৪৫৬৭৮৯",
  "_type";
  "numeric";
}
"bhks";
{
  "_digits";
  "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
  "_type";
  "numeric";
}
"brah";
{
  "_digits";
  "᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐",
  "_type";
  "numeric";
}
"cakm";
{
  "_digits";
  "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
  "_type";
  "numeric";
}
```



```
"cham";
{
  "_digits";
  "ႤႬႬႬႬႬႬႬႬႬ",
  "_type";
  "numeric";
}
"cyrl";
{
  "_rules";
  "cyrillic-lower",
  "_type";
  "algorithmic";
}
"deva";
{
  "_digits";
  "ॐ१२३४५६७८९",
  "_type";
  "numeric";
}
"ethi";
{
  "_rules";
  "ethiopic",
  "_type";
  "algorithmic";
}
"fullwide";
{
  "_digits";
  "ᲀ ᲁ ᲂ ᲃ ᲄ ᲅ ᲆ ᲇ ᲈ Ᲊ",
  "_type";
  "numeric";
}
"geor";
{
  "_rules";
  "georgian",
  "_type";
  "algorithmic";
}
"grek";
{
  "_rules";
  "greek-upper",
  "_type";
  "algorithmic";
}
"greklow";
{
  "_rules";
  "greek-lower",
  "_type";
  "algorithmic";
}
```

```

"gujr";
{
  "_digits";
  "૦૧૨૩૪૫૬૭૮૯",
  "_type";
  "numeric";
}
"guru";
{
  "_digits";
  "੦੧੨੩੪੫੬੭੮੯",
  "_type";
  "numeric";
}
"hanidays";
{
  "_rules";
  "zh/SpelloutRules/spellout-numbering-days",
  "_type";
  "algorithmic";
}
"hanidec";
{
  "_digits";
  "〇一二三四五六七八九",
  "_type";
  "numeric";
}
"hans";
{
  "_rules";
  "zh/SpelloutRules/spellout-cardinal",
  "_type";
  "algorithmic";
}
"hansfin";
{
  "_rules";
  "zh/SpelloutRules/spellout-cardinal-financial",
  "_type";
  "algorithmic";
}
"hant";
{
  "_rules";
  "zh_Hant/SpelloutRules/spellout-cardinal",
  "_type";
  "algorithmic";
}
"hantfin";
{
  "_rules";
  "zh_Hant/SpelloutRules/spellout-cardinal-financial",
  "_type";
  "algorithmic";
}

```

```

    "hebr";
    {
      "_rules";
      "hebrew",
      "_type";
      "algorithmic";
    }
    "hmng";
    {
      "_digits";
      "□□□□□□□□□□",
      "_type";
      "numeric";
    }
    "java";
    {
      "_digits";
      "௦௧௨௩௪௫௬௭௮௯",
      "_type";
      "numeric";
    }
    "jpan";
    {
      "_rules";
      "ja/SpelloutRules/spellout-cardinal",
      "_type";
      "algorithmic";
    }
    "jpanfin";
    {
      "_rules";
      "ja/SpelloutRules/spellout-cardinal-financial",
      "_type";
      "algorithmic";
    }
    "kali";
    {
      "_digits";
      "□□□□□□□□□□",
      "_type";
      "numeric";
    }
    "khmr";
    {
      "_digits";
      "០១២៣៤៥៦៧៨៩",
      "_type";
      "numeric";
    }
    "knda";
    {
      "_digits";
      "೦೧೨೩೪೫೬೭೮೯",
      "_type";
      "numeric";
    }

```

```

    }
    "lana";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "lanatham";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "laoo";
    {
        "_digits";
        "໐໑໒໓໔໕໖໗໘໙",
        "_type";
        "numeric";
    }
    "latn";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "lepc";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "limb";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mathbold";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathdbl";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }

```

```

    }
    "mathmono";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsanb";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsans";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mlym";
    {
        "_digits";
        "ഘറനർറ്റനൗവുൻ",
        "_type";
        "numeric";
    }
    "modi";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mong";
    {
        "_digits";
        "᠐᠑᠒᠓᠐ᠠᠨᠨᠠᠭ",
        "_type";
        "numeric";
    }
    "mroo";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mtei";
    {
        "_digits";
        "ႤႬႬႬႬႬႬႬႬႬႬႬ",
        "_type";
        "numeric";
    }

```

```

    }
    "mymr";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mymrshan";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mymrtlng";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "newa";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "nkoo";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "olck";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "orya";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "osma";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",

```

```

        "_type";
        "numeric";
    }
    "roman";
    {
        "_rules";
        "roman-upper",
        "_type";
        "algorithmic";
    }
    "romanlow";
    {
        "_rules";
        "roman-lower",
        "_type";
        "algorithmic";
    }
    "saur";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "shrd";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sind";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "sinh";
    {
        "_digits";
        "⁂௧௧௧௧௧௧௧௧௧",
        "_type";
        "numeric";
    }
    "sora";
    {
        "_digits";
        "0௧௨௩௪௫௬௭",
        "_type";
        "numeric";
    }
    "sund";
    {
        "_digits";
        "□□□□□□□□",

```

```

        "_type";
        "numeric";
    }
    "takr";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "talu";
    {
        "_digits";
        "ᱠᱟᱹᱨᱢᱟᱱ",
        "_type";
        "numeric";
    }
    "taml";
    {
        "_rules";
        "tamil",
        "_type";
        "algorithmic";
    }
    "tamldec";
    {
        "_digits";
        "ᱠᱟᱹᱨᱢᱟᱱ",
        "_type";
        "numeric";
    }
    "telu";
    {
        "_digits";
        "ᱠᱟᱹᱨᱢᱟᱱ",
        "_type";
        "numeric";
    }
    "thai";
    {
        "_digits";
        "ᱠᱟᱹᱨᱢᱟᱱ",
        "_type";
        "numeric";
    }
    "tibb";
    {
        "_digits";
        "ᱠᱟᱹᱨᱢᱟᱱ",
        "_type";
        "numeric";
    }
    "tirh";
    {
        "_digits";
        "□□□□□□□□□□",

```



```

        "_type";
        "numeric";
    }
    "vaii";
    {
        "_digits";
        "᠑᠒᠓᠔᠕᠖᠗᠘᠐᠑",
        "_type";
        "numeric";
    }
    "wara";
    {
        "_digits";
        "᠑᠒᠓᠔᠕᠖᠗᠘᠐᠑",
        "_type";
        "numeric";
    }
    }
}

```

NUMBERS.JSON

```

{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "numbers": {
        "defaultNumberingSystem": "arab",
        "otherNumberingSystems": {
          "native": "arab"
        },
        "minimumGroupingDigits": "1",
        "symbols-numberSystem-arab": {
          "decimal": ",",
          "group": ",",
          "list": ";",
          "percentSign": "%",
          "plusSign": "+",
          "minusSign": "-",
          "exponential": "٭",
          "superscriptingExponent": "×",
          "perMille": "‰",
          "infinity": "∞",
          "nan": "ليس رقم",
          "timeSeparator": ":"
        },
        "symbols-numberSystem-latn": {
          "decimal": ".",
          "group": ",",

```

```

    "list": ";",
    "percentSign": "%",
    "plusSign": "+",
    "minusSign": "-",
    "exponential": "E",
    "superscriptingExponent": "×",
    "perMille": "‰",
    "infinity": "∞",
    "nan": "ليس رقمًا",
    "timeSeparator": ":"
  },
  "decimalFormats-numberSystem-arab": {
    "standard": "#,##0.###",
    "long": {
      "decimalFormat": {
        "1000-count-zero": "0 ألف",
        "1000-count-one": "0 ألف",
        "1000-count-two": "0 ألف",
        "1000-count-few": "0 آلاف",
        "1000-count-many": "0 ألف",
        "1000-count-other": "0 ألف",
        "10000-count-zero": "00 ألف",
        "10000-count-one": "00 ألف",
        "10000-count-two": "00 ألف",
        "10000-count-few": "00 ألف",
        "10000-count-many": "00 ألف",
        "10000-count-other": "00 ألف",
        "100000-count-zero": "000 ألف",
        "100000-count-one": "000 ألف",
        "100000-count-two": "000 ألف",
        "100000-count-few": "000 ألف",
        "100000-count-many": "000 ألف",
        "100000-count-other": "000 ألف",
        "1000000-count-zero": "0 مليون",
        "1000000-count-one": "0 مليون",
        "1000000-count-two": "0 مليون",
        "1000000-count-few": "0 ملايين",
        "1000000-count-many": "0 مليون",
        "1000000-count-other": "0 مليون",
        "10000000-count-zero": "00 مليون",
        "10000000-count-one": "00 مليون",
        "10000000-count-two": "00 مليون",
        "10000000-count-few": "00 ملايين",
        "10000000-count-many": "00 مليون",
        "10000000-count-other": "00 مليون",
        "100000000-count-zero": "000 مليون",
        "100000000-count-one": "000 مليون",
        "100000000-count-two": "000 مليون",
        "100000000-count-few": "000 مليون",
        "100000000-count-many": "000 مليون",
        "100000000-count-other": "000 مليون",
        "1000000000-count-zero": "0 مليار",
        "1000000000-count-one": "0 مليار",
        "1000000000-count-two": "0 مليار",
        "1000000000-count-few": "0 مليار",
        "1000000000-count-many": "0 مليار",
        "1000000000-count-other": "0 مليار",
      }
    }
  }
}

```

```

        "10000000000-count-zero": "00 مليار",
        "10000000000-count-one": "00 مليار",
        "10000000000-count-two": "00 مليار",
        "10000000000-count-few": "00 مليار",
        "10000000000-count-many": "00 مليار",
        "10000000000-count-other": "00 مليار",
        "100000000000-count-zero": "000 مليار",
        "100000000000-count-one": "000 مليار",
        "100000000000-count-two": "000 مليار",
        "100000000000-count-few": "000 مليار",
        "100000000000-count-many": "000 مليار",
        "100000000000-count-other": "000 مليار",
        "1000000000000-count-zero": "0 ترليون",
        "1000000000000-count-one": "0 ترليون",
        "1000000000000-count-two": "0 ترليون",
        "1000000000000-count-few": "0 ترليون",
        "1000000000000-count-many": "0 ترليون",
        "1000000000000-count-other": "0 ترليون",
        "10000000000000-count-zero": "00 ترليون",
        "10000000000000-count-one": "00 ترليون",
        "10000000000000-count-two": "00 ترليون",
        "10000000000000-count-few": "00 ترليون",
        "10000000000000-count-many": "00 ترليون",
        "10000000000000-count-other": "00 ترليون",
        "100000000000000-count-zero": "000 ترليون",
        "100000000000000-count-one": "000 ترليون",
        "100000000000000-count-two": "000 ترليون",
        "100000000000000-count-few": "000 ترليون",
        "100000000000000-count-many": "000 ترليون",
        "100000000000000-count-other": "000 ترليون"
    },
    },
    "short": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",
            "1000-count-one": "0 ألف",
            "1000-count-two": "0 ألف",
            "1000-count-few": "0 آلاف",
            "1000-count-many": "0 ألف",
            "1000-count-other": "0 ألف",
            "10000-count-zero": "00 ألف",
            "10000-count-one": "00 ألف",
            "10000-count-two": "00 ألف",
            "10000-count-few": "00 ألف",
            "10000-count-many": "00 ألف",
            "10000-count-other": "00 ألف",
            "100000-count-zero": "000 ألف",
            "100000-count-one": "000 ألف",
            "100000-count-two": "000 ألف",
            "100000-count-few": "000 ألف",
            "100000-count-many": "000 ألف",
            "100000-count-other": "000 ألف",
            "1000000-count-zero": "0 مليون",
            "1000000-count-one": "0 مليون",
            "1000000-count-two": "0 مليون",
            "1000000-count-few": "0 مليون",
            "1000000-count-many": "0 مليون",

```

```

        "1000000-count-other": "0 مليون",
        "1000000-count-zero": "00 مليون",
        "1000000-count-one": "00 مليون",
        "1000000-count-two": "00 مليون",
        "1000000-count-few": "00 مليون",
        "1000000-count-many": "00 مليون",
        "1000000-count-other": "00 مليون",
        "10000000-count-zero": "000 مليون",
        "10000000-count-one": "000 مليون",
        "10000000-count-two": "000 مليون",
        "10000000-count-few": "000 مليون",
        "10000000-count-many": "000 مليون",
        "10000000-count-other": "000 مليون",
        "100000000-count-zero": "0 مليار",
        "100000000-count-one": "0 مليار",
        "100000000-count-two": "0 مليار",
        "100000000-count-few": "0 مليار",
        "100000000-count-many": "0 مليار",
        "100000000-count-other": "0 مليار",
        "1000000000-count-zero": "00 مليار",
        "1000000000-count-one": "00 مليار",
        "1000000000-count-two": "00 مليار",
        "1000000000-count-few": "00 مليار",
        "1000000000-count-many": "00 مليار",
        "1000000000-count-other": "00 مليار",
        "10000000000-count-zero": "000 مليار",
        "10000000000-count-one": "000 مليار",
        "10000000000-count-two": "000 مليار",
        "10000000000-count-few": "000 مليار",
        "10000000000-count-many": "000 مليار",
        "10000000000-count-other": "000 مليار",
        "100000000000-count-zero": "0 ترليون",
        "100000000000-count-one": "0 ترليون",
        "100000000000-count-two": "0 ترليون",
        "100000000000-count-few": "0 ترليون",
        "100000000000-count-many": "0 ترليون",
        "100000000000-count-other": "0 ترليون",
        "1000000000000-count-zero": "00 ترليون",
        "1000000000000-count-one": "00 ترليون",
        "1000000000000-count-two": "00 ترليون",
        "1000000000000-count-few": "00 ترليون",
        "1000000000000-count-many": "00 ترليون",
        "1000000000000-count-other": "00 ترليون",
        "10000000000000-count-zero": "000 ترليون",
        "10000000000000-count-one": "000 ترليون",
        "10000000000000-count-two": "000 ترليون",
        "10000000000000-count-few": "000 ترليون",
        "10000000000000-count-many": "000 ترليون",
        "10000000000000-count-other": "000 ترليون"
    }
}
},
"decimalFormats-numberSystem-latn": {
    "standard": "#,##0.###",
    "long": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",

```

```

"1000-count-one": "0 ألف",
"1000-count-two": "0 ألف",
"1000-count-few": "0 آلاف",
"1000-count-many": "0 ألف",
"1000-count-other": "0 ألف",
"10000-count-zero": "00 ألف",
"10000-count-one": "00 ألف",
"10000-count-two": "00 ألف",
"10000-count-few": "00 ألف",
"10000-count-many": "00 ألف",
"10000-count-other": "00 ألف",
"100000-count-zero": "000 ألف",
"100000-count-one": "000 ألف",
"100000-count-two": "000 ألف",
"100000-count-few": "000 ألف",
"100000-count-many": "000 ألف",
"100000-count-other": "000 ألف",
"1000000-count-zero": "0 مليون",
"1000000-count-one": "0 مليون",
"1000000-count-two": "0 مليون",
"1000000-count-few": "0 ملايين",
"1000000-count-many": "0 مليون",
"1000000-count-other": "0 مليون",
"10000000-count-zero": "00 مليون",
"10000000-count-one": "00 مليون",
"10000000-count-two": "00 مليون",
"10000000-count-few": "00 ملايين",
"10000000-count-many": "00 مليون",
"10000000-count-other": "00 مليون",
"100000000-count-zero": "000 مليون",
"100000000-count-one": "000 مليون",
"100000000-count-two": "000 مليون",
"100000000-count-few": "000 مليون",
"100000000-count-many": "000 مليون",
"100000000-count-other": "000 مليون",
"1000000000-count-zero": "0 مليار",
"1000000000-count-one": "0 مليار",
"1000000000-count-two": "0 مليار",
"1000000000-count-few": "0 مليار",
"1000000000-count-many": "0 مليار",
"1000000000-count-other": "0 مليار",
"10000000000-count-zero": "00 مليار",
"10000000000-count-one": "00 مليار",
"10000000000-count-two": "00 مليار",
"10000000000-count-few": "00 مليار",
"10000000000-count-many": "00 مليار",
"10000000000-count-other": "00 مليار",
"100000000000-count-zero": "000 مليار",
"100000000000-count-one": "000 مليار",
"100000000000-count-two": "000 مليار",
"100000000000-count-few": "000 مليار",
"100000000000-count-many": "000 مليار",
"100000000000-count-other": "000 مليار",
"1000000000000-count-zero": "0 ترليون",
"1000000000000-count-one": "0 ترليون",
"1000000000000-count-two": "0 ترليون",
"1000000000000-count-few": "0 ترليون",

```

```

        "1000000000000-count-many": "0 ترليون",
        "1000000000000-count-other": "0 ترليون",
        "1000000000000-count-zero": "00 ترليون",
        "1000000000000-count-one": "00 ترليون",
        "1000000000000-count-two": "00 ترليون",
        "1000000000000-count-few": "00 ترليون",
        "1000000000000-count-many": "00 ترليون",
        "1000000000000-count-other": "00 ترليون",
        "1000000000000-count-zero": "000 ترليون",
        "1000000000000-count-one": "000 ترليون",
        "1000000000000-count-two": "000 ترليون",
        "1000000000000-count-few": "000 ترليون",
        "1000000000000-count-many": "000 ترليون",
        "1000000000000-count-other": "000 ترليون"
    },
    },
    "short": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",
            "1000-count-one": "0 ألف",
            "1000-count-two": "0 ألف",
            "1000-count-few": "0 آلاف",
            "1000-count-many": "0 ألف",
            "1000-count-other": "0 ألف",
            "10000-count-zero": "00 ألف",
            "10000-count-one": "00 ألف",
            "10000-count-two": "00 ألف",
            "10000-count-few": "00 ألف",
            "10000-count-many": "00 ألف",
            "10000-count-other": "00 ألف",
            "100000-count-zero": "000 ألف",
            "100000-count-one": "000 ألف",
            "100000-count-two": "000 ألف",
            "100000-count-few": "000 ألف",
            "100000-count-many": "000 ألف",
            "100000-count-other": "000 ألف",
            "1000000-count-zero": "0 مليون",
            "1000000-count-one": "0 مليون",
            "1000000-count-two": "0 مليون",
            "1000000-count-few": "0 مليون",
            "1000000-count-many": "0 مليون",
            "1000000-count-other": "0 مليون",
            "10000000-count-zero": "00 مليون",
            "10000000-count-one": "00 مليون",
            "10000000-count-two": "00 مليون",
            "10000000-count-few": "00 مليون",
            "10000000-count-many": "00 مليون",
            "10000000-count-other": "00 مليون",
            "100000000-count-zero": "000 مليون",
            "100000000-count-one": "000 مليون",
            "100000000-count-two": "000 مليون",
            "100000000-count-few": "000 مليون",
            "100000000-count-many": "000 مليون",
            "100000000-count-other": "000 مليون",
            "1000000000-count-zero": "0 مليار",
            "1000000000-count-one": "0 مليار",
            "1000000000-count-two": "0 مليار",

```

```

        "1000000000-count-few": "0 مليار",
        "1000000000-count-many": "0 مليار",
        "1000000000-count-other": "0 مليار",
        "1000000000-count-zero": "00 مليار",
        "1000000000-count-one": "00 مليار",
        "1000000000-count-two": "00 مليار",
        "1000000000-count-few": "00 مليار",
        "1000000000-count-many": "00 مليار",
        "1000000000-count-other": "00 مليار",
        "1000000000-count-zero": "000 مليار",
        "1000000000-count-one": "000 مليار",
        "1000000000-count-two": "000 مليار",
        "1000000000-count-few": "000 مليار",
        "1000000000-count-many": "000 مليار",
        "1000000000-count-other": "000 مليار",
        "100000000000-count-zero": "0 ترليون",
        "100000000000-count-one": "0 ترليون",
        "100000000000-count-two": "0 ترليون",
        "100000000000-count-few": "0 ترليون",
        "100000000000-count-many": "0 ترليون",
        "100000000000-count-other": "0 ترليون",
        "1000000000000-count-zero": "00 ترليون",
        "1000000000000-count-one": "00 ترليون",
        "1000000000000-count-two": "00 ترليون",
        "1000000000000-count-few": "00 ترليون",
        "1000000000000-count-many": "00 ترليون",
        "1000000000000-count-other": "00 ترليون",
        "10000000000000-count-zero": "000 ترليون",
        "10000000000000-count-one": "000 ترليون",
        "10000000000000-count-two": "000 ترليون",
        "10000000000000-count-few": "000 ترليون",
        "10000000000000-count-many": "000 ترليون",
        "10000000000000-count-other": "000 ترليون"
    }
}
},
"scientificFormats-numberSystem-arab": {
    "standard": "#E0"
},
"scientificFormats-numberSystem-latn": {
    "standard": "#E0"
},
"percentFormats-numberSystem-arab": {
    "standard": "#,##0 %"
},
"percentFormats-numberSystem-latn": {
    "standard": "#,##0%"
},
"currencyFormats-numberSystem-arab": {
    "currencySpacing": {
        "beforeCurrency": {
            "currencyMatch": "[:^S:]",
            "surroundingMatch": "[:digit:]",
            "insertBetween": " "
        },
        "afterCurrency": {
            "currencyMatch": "[:^S:]",

```

```

        "surroundingMatch": "[:digit:]",
        "insertBetween": " "
    },
    },
    "standard": "#,##0.00 ¤",
    "accounting": "#,##0.00 ¤",
    "unitPattern-count-zero": "{0} {1}",
    "unitPattern-count-one": "{0} {1}",
    "unitPattern-count-two": "{0} {1}",
    "unitPattern-count-few": "{0} {1}",
    "unitPattern-count-many": "{0} {1}",
    "unitPattern-count-other": "{0} {1}"
},
"currencyFormats-numberSystem-latn": {
    "currencySpacing": {
        "beforeCurrency": {
            "currencyMatch": "[:^S:]",
            "surroundingMatch": "[:digit:]",
            "insertBetween": " "
        },
        "afterCurrency": {
            "currencyMatch": "[:^S:]",
            "surroundingMatch": "[:digit:]",
            "insertBetween": " "
        }
    },
    "standard": "¤ #,##0.00",
    "accounting": "¤#,##0.00; (¤#,##0.00)",
    "short": {
        "standard": {
            "1000-count-zero": "¤ 0 ألف",
            "1000-count-one": "¤ 0 ألف",
            "1000-count-two": "¤ 0 ألف",
            "1000-count-few": "¤ 0 ألف",
            "1000-count-many": "¤ 0 ألف",
            "1000-count-other": "¤ 0 ألف",
            "10000-count-zero": "¤ 00 ألف",
            "10000-count-one": "¤ 00 ألف",
            "10000-count-two": "¤ 00 ألف",
            "10000-count-few": "¤ 00 ألف",
            "10000-count-many": "¤ 00 ألف",
            "10000-count-other": "¤ 00 ألف",
            "100000-count-zero": "¤ 000 ألف",
            "100000-count-one": "¤ 000 ألف",
            "100000-count-two": "¤ 000 ألف",
            "100000-count-few": "¤ 000 ألف",
            "100000-count-many": "¤ 000 ألف",
            "100000-count-other": "¤ 000 ألف",
            "1000000-count-zero": "¤ 0 مليون",
            "1000000-count-one": "¤ 0 مليون",
            "1000000-count-two": "¤ 0 مليون",
            "1000000-count-few": "¤ 0 مليون",
            "1000000-count-many": "¤ 0 مليون",
            "1000000-count-other": "¤ 0 مليون",
            "10000000-count-zero": "¤ 00 مليون",
            "10000000-count-one": "¤ 00 مليون",
            "10000000-count-two": "¤ 00 مليون",

```



```

        "10000000-count-few": "٠٠ مليون",
        "10000000-count-many": "٠٠ مليون",
        "10000000-count-other": "٠٠ مليون",
        "100000000-count-zero": "٠٠٠ مليون",
        "100000000-count-one": "٠٠٠ مليون",
        "100000000-count-two": "٠٠٠ مليون",
        "100000000-count-few": "٠٠٠ مليون",
        "100000000-count-many": "٠٠٠ مليون",
        "100000000-count-other": "٠٠٠ مليون",
        "1000000000-count-zero": "٠ مليار",
        "1000000000-count-one": "٠ مليار",
        "1000000000-count-two": "٠ مليار",
        "1000000000-count-few": "٠ مليار",
        "1000000000-count-many": "٠ مليار",
        "1000000000-count-other": "٠ مليار",
        "10000000000-count-zero": "٠٠ مليار",
        "10000000000-count-one": "٠٠ مليار",
        "10000000000-count-two": "٠٠ مليار",
        "10000000000-count-few": "٠٠ مليار",
        "10000000000-count-many": "٠٠ مليار",
        "10000000000-count-other": "٠٠ مليار",
        "100000000000-count-zero": "٠٠٠ مليار",
        "100000000000-count-one": "٠٠٠ مليار",
        "100000000000-count-two": "٠٠٠ مليار",
        "100000000000-count-few": "٠٠٠ مليار",
        "100000000000-count-many": "٠٠٠ مليار",
        "100000000000-count-other": "٠٠٠ مليار",
        "1000000000000-count-zero": "٠ ترليون",
        "1000000000000-count-one": "٠ ترليون",
        "1000000000000-count-two": "٠ ترليون",
        "1000000000000-count-few": "٠ ترليون",
        "1000000000000-count-many": "٠ ترليون",
        "1000000000000-count-other": "٠ ترليون",
        "10000000000000-count-zero": "٠٠ ترليون",
        "10000000000000-count-one": "٠٠ ترليون",
        "10000000000000-count-two": "٠٠ ترليون",
        "10000000000000-count-few": "٠٠ ترليون",
        "10000000000000-count-many": "٠٠ ترليون",
        "10000000000000-count-other": "٠٠ ترليون",
        "100000000000000-count-zero": "٠٠٠ ترليون",
        "100000000000000-count-one": "٠٠٠ ترليون",
        "100000000000000-count-two": "٠٠٠ ترليون",
        "100000000000000-count-few": "٠٠٠ ترليون",
        "100000000000000-count-many": "٠٠٠ ترليون",
        "100000000000000-count-other": "٠٠٠ ترليون"
    },
    },
    "unitPattern-count-zero": "{0} {1}",
    "unitPattern-count-one": "{0} {1}",
    "unitPattern-count-two": "{0} {1}",
    "unitPattern-count-few": "{0} {1}",
    "unitPattern-count-many": "{0} {1}",
    "unitPattern-count-other": "{0} {1}"
},
"miscPatterns-numberSystem-arab": {
    "atLeast": "+{0}",
    "range": "{0}-{1}"
}

```

```

    },
    "miscPatterns-numberSystem-latn": {
      "atLeast": "+{0}",
      "range": "{0}-{1}"
    },
    "minimalPairs": {
      "pluralMinimalPairs-count-zero": "{0} كتاب",
      "pluralMinimalPairs-count-one": "ولد واحد حضر",
      "pluralMinimalPairs-count-two": "ولدان حضرا",
      "pluralMinimalPairs-count-few": "{0} أولاد حضروا",
      "pluralMinimalPairs-count-many": "{0} ولدًا حضروا",
      "pluralMinimalPairs-count-other": "{0} ولد حضروا",
      "other": "اتجه إلى المنعطف الـ {0} يمينًا."
    }
  }
}
}
}

```

NUMBERS.JSX

```

{
  "main";
  {
    "ar";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13686 $",
          "_cldrVersion";
          "32";
        }
        "language";
        "ar";
      }
      "numbers";
      {
        "defaultNumberingSystem";
        "arab",
        "otherNumberingSystems";
        {
          "native";
          "arab";
        }
        "minimumGroupingDigits";
        "1",
        "symbols-numberSystem-arab";
        {
          "decimal";
          ",",
          "group";
          ",",
          "list";
        }
      }
    }
  }
}

```

```

        ":",
        "percentSign";
        "%",
        "plusSign";
        "+",
        "minusSign";
        "-",
        "exponential";
        "١٠",
        "superscriptingExponent";
        "×",
        "perMille";
        "‰",
        "infinity";
        "∞",
        "nan";
        "ليس رقم",
        "timeSeparator";
        ":";
    }
    "symbols-numberSystem-latn";
    {
        "decimal";
        ".",
        "group";
        ",",
        "list";
        ";",
        "percentSign";
        "‰",
        "plusSign";
        "+",
        "minusSign";
        "-",
        "exponential";
        "E",
        "superscriptingExponent";
        "×",
        "perMille";
        "‰",
        "infinity";
        "∞",
        "nan";
        "ليس رقمًا",
        "timeSeparator";
        ":";
    }
    "decimalFormats-numberSystem-arab";
    {
        "standard";
        "#,##0.###",
        "long";
        {
            "decimalFormat";
            {
                "1000-count-zero";
                "ألف 0",
            }
        }
    }

```

```

        "1000-count-one";
    "0 ألف",
        "1000-count-two";
    "0 ألف",
        "1000-count-few";
    "0 آلاف",
        "1000-count-many";
    "0 ألف",
        "1000-count-other";
    "0 ألف",
        "10000-count-zero";
    "00 ألف",
        "10000-count-one";
    "00 ألف",
        "10000-count-two";
    "00 ألف",
        "10000-count-few";
    "00 ألف",
        "10000-count-many";
    "00 ألف",
        "10000-count-other";
    "00 ألف",
        "100000-count-zero";
    "000 ألف",
        "100000-count-one";
    "000 ألف",
        "100000-count-two";
    "000 ألف",
        "100000-count-few";
    "000 ألف",
        "100000-count-many";
    "000 ألف",
        "100000-count-other";
    "000 ألف",
        "1000000-count-zero";
    "0 مليون",
        "1000000-count-one";
    "0 مليون",
        "1000000-count-two";
    "0 مليون",
        "1000000-count-few";
    "0 ملايين",
        "1000000-count-many";
    "0 مليون",
        "1000000-count-other";
    "0 مليون",
        "10000000-count-zero";
    "00 مليون",
        "10000000-count-one";
    "00 مليون",
        "10000000-count-two";
    "00 مليون",
        "10000000-count-few";
    "00 ملايين",
        "10000000-count-many";
    "00 مليون",
        "10000000-count-other";

```

```

"00 مليون",
  "100000000-count-zero";
"000 مليون",
  "100000000-count-one";
"000 مليون",
  "100000000-count-two";
"000 مليون",
  "100000000-count-few";
"000 مليون",
  "100000000-count-many";
"000 مليون",
  "100000000-count-other";
"000 مليون",
  "1000000000-count-zero";
"0 مليار",
  "1000000000-count-one";
"0 مليار",
  "1000000000-count-two";
"0 مليار",
  "1000000000-count-few";
"0 مليار",
  "1000000000-count-many";
"0 مليار",
  "1000000000-count-other";
"0 مليار",
  "10000000000-count-zero";
"00 مليار",
  "10000000000-count-one";
"00 مليار",
  "10000000000-count-two";
"00 مليار",
  "10000000000-count-few";
"00 مليار",
  "10000000000-count-many";
"00 مليار",
  "10000000000-count-other";
"00 مليار",
  "100000000000-count-zero";
"000 مليار",
  "100000000000-count-one";
"000 مليار",
  "100000000000-count-two";
"000 مليار",
  "100000000000-count-few";
"000 مليار",
  "100000000000-count-many";
"000 مليار",
  "100000000000-count-other";
"000 مليار",
  "1000000000000-count-zero";
"0 ترليون",
  "1000000000000-count-one";
"0 ترليون",
  "1000000000000-count-two";
"0 ترليون",
  "1000000000000-count-few";
"0 ترليون",

```

```

        "1000000000000-count-many";
    "0 ترليون",
    "1000000000000-count-other";
    "0 ترليون",
    "1000000000000-count-zero";
    "00 ترليون",
    "1000000000000-count-one";
    "00 ترليون",
    "1000000000000-count-two";
    "00 ترليون",
    "1000000000000-count-few";
    "00 ترليون",
    "1000000000000-count-many";
    "00 ترليون",
    "1000000000000-count-other";
    "00 ترليون",
    "1000000000000-count-zero";
    "000 ترليون",
    "1000000000000-count-one";
    "000 ترليون",
    "1000000000000-count-two";
    "000 ترليون",
    "1000000000000-count-few";
    "000 ترليون",
    "1000000000000-count-many";
    "000 ترليون",
    "1000000000000-count-other";
    "000 ترليون";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-zero";
        "0 ألف",
        "1000-count-one";
        "0 ألف",
        "1000-count-two";
        "0 ألف",
        "1000-count-few";
        "0 آلاف",
        "1000-count-many";
        "0 ألف",
        "1000-count-other";
        "0 ألف",
        "10000-count-zero";
        "00 ألف",
        "10000-count-one";
        "00 ألف",
        "10000-count-two";
        "00 ألف",
        "10000-count-few";
        "00 ألف",
        "10000-count-many";
        "00 ألف",
        "10000-count-other";
    }
}

```

```

"00 ألف",
    "100000-count-zero";
"000 ألف",
    "100000-count-one";
"000 ألف",
    "100000-count-two";
"000 ألف",
    "100000-count-few";
"000 ألف",
    "100000-count-many";
"000 ألف",
    "100000-count-other";
"000 ألف",
    "1000000-count-zero";
"0 مليون",
    "1000000-count-one";
"0 مليون",
    "1000000-count-two";
"0 مليون",
    "1000000-count-few";
"0 مليون",
    "1000000-count-many";
"0 مليون",
    "1000000-count-other";
"0 مليون",
    "10000000-count-zero";
"00 مليون",
    "10000000-count-one";
"00 مليون",
    "10000000-count-two";
"00 مليون",
    "10000000-count-few";
"00 مليون",
    "10000000-count-many";
"00 مليون",
    "10000000-count-other";
"00 مليون",
    "100000000-count-zero";
"000 مليون",
    "100000000-count-one";
"000 مليون",
    "100000000-count-two";
"000 مليون",
    "100000000-count-few";
"000 مليون",
    "100000000-count-many";
"000 مليون",
    "100000000-count-other";
"000 مليون",
    "1000000000-count-zero";
"0 مليار",
    "1000000000-count-one";
"0 مليار",
    "1000000000-count-two";
"0 مليار",
    "1000000000-count-few";
"0 مليار",

```

```

        "1000000000-count-many";
        "0 مليار",
        "1000000000-count-other";
        "0 مليار",
        "10000000000-count-zero";
        "00 مليار",
        "10000000000-count-one";
        "00 مليار",
        "10000000000-count-two";
        "00 مليار",
        "10000000000-count-few";
        "00 مليار",
        "10000000000-count-many";
        "00 مليار",
        "10000000000-count-other";
        "00 مليار",
        "100000000000-count-zero";
        "000 مليار",
        "100000000000-count-one";
        "000 مليار",
        "100000000000-count-two";
        "000 مليار",
        "100000000000-count-few";
        "000 مليار",
        "100000000000-count-many";
        "000 مليار",
        "100000000000-count-other";
        "000 مليار",
        "1000000000000-count-zero";
        "0 ترليون",
        "1000000000000-count-one";
        "0 ترليون",
        "1000000000000-count-two";
        "0 ترليون",
        "1000000000000-count-few";
        "0 ترليون",
        "1000000000000-count-many";
        "0 ترليون",
        "1000000000000-count-other";
        "0 ترليون",
        "10000000000000-count-zero";
        "00 ترليون",
        "10000000000000-count-one";
        "00 ترليون",
        "10000000000000-count-two";
        "00 ترليون",
        "10000000000000-count-few";
        "00 ترليون",
        "10000000000000-count-many";
        "00 ترليون",
        "10000000000000-count-other";
        "00 ترليون",
        "100000000000000-count-zero";
        "000 ترليون",
        "100000000000000-count-one";
        "000 ترليون",
        "100000000000000-count-two";

```



```

        "000 ترليون",
        "1000000000000000-count-few";
        "000 ترليون",
        "1000000000000000-count-many";
        "000 ترليون",
        "1000000000000000-count-other";
        "000 ترليون";
    }
}
}
"decimalFormats-numberSystem-latn";
{
    "standard";
    "#,##0.###",
    "long";
    {
        "decimalFormat";
        {
            "1000-count-zero";
            "0 ألف",
            "1000-count-one";
            "0 ألف",
            "1000-count-two";
            "0 ألف",
            "1000-count-few";
            "0 آلاف",
            "1000-count-many";
            "0 ألف",
            "1000-count-other";
            "0 ألف",
            "10000-count-zero";
            "00 ألف",
            "10000-count-one";
            "00 ألف",
            "10000-count-two";
            "00 ألف",
            "10000-count-few";
            "00 ألف",
            "10000-count-many";
            "00 ألف",
            "10000-count-other";
            "00 ألف",
            "100000-count-zero";
            "000 ألف",
            "100000-count-one";
            "000 ألف",
            "100000-count-two";
            "000 ألف",
            "100000-count-few";
            "000 ألف",
            "100000-count-many";
            "000 ألف",
            "100000-count-other";
            "000 ألف",
            "1000000-count-zero";
            "0 مليون",
            "1000000-count-one";

```

```

"0 مليون",
  "1000000-count-two";
"0 مليون",
  "1000000-count-few";
"0 ملايين",
  "1000000-count-many";
"0 مليون",
  "1000000-count-other";
"0 مليون",
  "10000000-count-zero";
"00 مليون",
  "10000000-count-one";
"00 مليون",
  "10000000-count-two";
"00 مليون",
  "10000000-count-few";
"00 ملايين",
  "10000000-count-many";
"00 مليون",
  "10000000-count-other";
"00 مليون",
  "100000000-count-zero";
"000 مليون",
  "100000000-count-one";
"000 مليون",
  "100000000-count-two";
"000 مليون",
  "100000000-count-few";
"000 مليون",
  "100000000-count-many";
"000 مليون",
  "100000000-count-other";
"000 مليون",
  "1000000000-count-zero";
"0 مليار",
  "1000000000-count-one";
"0 مليار",
  "1000000000-count-two";
"0 مليار",
  "1000000000-count-few";
"0 مليار",
  "1000000000-count-many";
"0 مليار",
  "1000000000-count-other";
"0 مليار",
  "10000000000-count-zero";
"00 مليار",
  "10000000000-count-one";
"00 مليار",
  "10000000000-count-two";
"00 مليار",
  "10000000000-count-few";
"00 مليار",
  "10000000000-count-many";
"00 مليار",
  "10000000000-count-other";
"00 مليار",

```

```

        "100000000000-count-zero";
        "000 مليار",
        "100000000000-count-one";
        "000 مليار",
        "100000000000-count-two";
        "000 مليار",
        "100000000000-count-few";
        "000 مليار",
        "100000000000-count-many";
        "000 مليار",
        "100000000000-count-other";
        "000 مليار",
        "100000000000-count-zero";
        "0 ترليون",
        "100000000000-count-one";
        "0 ترليون",
        "100000000000-count-two";
        "0 ترليون",
        "100000000000-count-few";
        "0 ترليون",
        "100000000000-count-many";
        "0 ترليون",
        "100000000000-count-other";
        "0 ترليون",
        "100000000000-count-zero";
        "00 ترليون",
        "100000000000-count-one";
        "00 ترليون",
        "100000000000-count-two";
        "00 ترليون",
        "100000000000-count-few";
        "00 ترليون",
        "100000000000-count-many";
        "00 ترليون",
        "100000000000-count-other";
        "00 ترليون",
        "100000000000-count-zero";
        "000 ترليون",
        "100000000000-count-one";
        "000 ترليون",
        "100000000000-count-two";
        "000 ترليون",
        "100000000000-count-few";
        "000 ترليون",
        "100000000000-count-many";
        "000 ترليون",
        "100000000000-count-other";
        "000 ترليون";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-zero";
        "0 ألف",
        "1000-count-one";
    }
}

```

```

"0 ألف",
  "1000-count-two";
"0 ألف",
  "1000-count-few";
"0 آلاف",
  "1000-count-many";
"0 ألف",
  "1000-count-other";
"0 ألف",
  "10000-count-zero";
"00 ألف",
  "10000-count-one";
"00 ألف",
  "10000-count-two";
"00 ألف",
  "10000-count-few";
"00 ألف",
  "10000-count-many";
"00 ألف",
  "10000-count-other";
"00 ألف",
  "100000-count-zero";
"000 ألف",
  "100000-count-one";
"000 ألف",
  "100000-count-two";
"000 ألف",
  "100000-count-few";
"000 ألف",
  "100000-count-many";
"000 ألف",
  "100000-count-other";
"000 ألف",
  "1000000-count-zero";
"0 مليون",
  "1000000-count-one";
"0 مليون",
  "1000000-count-two";
"0 مليون",
  "1000000-count-few";
"0 مليون",
  "1000000-count-many";
"0 مليون",
  "1000000-count-other";
"0 مليون",
  "10000000-count-zero";
"00 مليون",
  "10000000-count-one";
"00 مليون",
  "10000000-count-two";
"00 مليون",
  "10000000-count-few";
"00 مليون",
  "10000000-count-many";
"00 مليون",
  "10000000-count-other";
"00 مليون",

```

```

        "100000000-count-zero";
    "000 مليون",
        "100000000-count-one";
    "000 مليون",
        "100000000-count-two";
    "000 مليون",
        "100000000-count-few";
    "000 مليون",
        "100000000-count-many";
    "000 مليون",
        "100000000-count-other";
    "000 مليون",
        "1000000000-count-zero";
    "0 مليار",
        "1000000000-count-one";
    "0 مليار",
        "1000000000-count-two";
    "0 مليار",
        "1000000000-count-few";
    "0 مليار",
        "1000000000-count-many";
    "0 مليار",
        "1000000000-count-other";
    "0 مليار",
        "10000000000-count-zero";
    "00 مليار",
        "10000000000-count-one";
    "00 مليار",
        "10000000000-count-two";
    "00 مليار",
        "10000000000-count-few";
    "00 مليار",
        "10000000000-count-many";
    "00 مليار",
        "10000000000-count-other";
    "00 مليار",
        "100000000000-count-zero";
    "000 مليار",
        "100000000000-count-one";
    "000 مليار",
        "100000000000-count-two";
    "000 مليار",
        "100000000000-count-few";
    "000 مليار",
        "100000000000-count-many";
    "000 مليار",
        "100000000000-count-other";
    "000 مليار",
        "1000000000000-count-zero";
    "0 ترليون",
        "1000000000000-count-one";
    "0 ترليون",
        "1000000000000-count-two";
    "0 ترليون",
        "1000000000000-count-few";
    "0 ترليون",
        "1000000000000-count-many";

```

```

        "0 ترليون",
        "1000000000000-count-other";
        "0 ترليون",
        "1000000000000-count-zero";
        "00 ترليون",
        "1000000000000-count-one";
        "00 ترليون",
        "1000000000000-count-two";
        "00 ترليون",
        "1000000000000-count-few";
        "00 ترليون",
        "1000000000000-count-many";
        "00 ترليون",
        "1000000000000-count-other";
        "00 ترليون",
        "1000000000000-count-zero";
        "000 ترليون",
        "1000000000000-count-one";
        "000 ترليون",
        "1000000000000-count-two";
        "000 ترليون",
        "1000000000000-count-few";
        "000 ترليون",
        "1000000000000-count-many";
        "000 ترليون",
        "1000000000000-count-other";
        "000 ترليون";
    }
}
"scientificFormats-numberSystem-arab";
{
    "standard";
    "#E0";
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}
"percentFormats-numberSystem-arab";
{
    "standard";
    "#,##0 %";
}
"percentFormats-numberSystem-latn";
{
    "standard";
    "#,##0%";
}
"currencyFormats-numberSystem-arab";
{
    "currencySpacing";
    {
        "beforeCurrency";
        {
            "currencyMatch";

```

```

        "[:^S:]",
        "surroundingMatch";
        "[:digit:]",
        "insertBetween";
        " ";
    }
    "afterCurrency";
    {
        "currencyMatch";
        "[:^S:]",
        "surroundingMatch";
        "[:digit:]",
        "insertBetween";
        " ";
    }
}
"standard";
"#,##0.00 ¤",
    "accounting";
"#,##0.00 ¤",
    "unitPattern-count-zero";
"{0} {1}",
    "unitPattern-count-one";
"{0} {1}",
    "unitPattern-count-two";
"{0} {1}",
    "unitPattern-count-few";
"{0} {1}",
    "unitPattern-count-many";
"{0} {1}",
    "unitPattern-count-other";
"{0} {1}";
}
"currencyFormats-numberSystem-latn";
{
    "currencySpacing";
    {
        "beforeCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
        "afterCurrency";
        {
            "currencyMatch";
            "[:^S:]",
            "surroundingMatch";
            "[:digit:]",
            "insertBetween";
            " ";
        }
    }
}
"standard";

```

```

"¤ #,##0.00",
  "accounting";
"¤#,##0.00; (¤#,##0.00) ",
  "short";
{
  "standard";
  {
    "1000-count-zero";
    "¤ 0 ألف",
    "1000-count-one";
    "¤ 0 ألف",
    "1000-count-two";
    "¤ 0 ألف",
    "1000-count-few";
    "¤ 0 ألف",
    "1000-count-many";
    "¤ 0 ألف",
    "1000-count-other";
    "¤ 0 ألف",
    "10000-count-zero";
    "¤ 00 ألف",
    "10000-count-one";
    "¤ 00 ألف",
    "10000-count-two";
    "¤ 00 ألف",
    "10000-count-few";
    "¤ 00 ألف",
    "10000-count-many";
    "¤ 00 ألف",
    "10000-count-other";
    "¤ 00 ألف",
    "100000-count-zero";
    "¤ 000 ألف",
    "100000-count-one";
    "¤ 000 ألف",
    "100000-count-two";
    "¤ 000 ألف",
    "100000-count-few";
    "¤ 000 ألف",
    "100000-count-many";
    "¤ 000 ألف",
    "100000-count-other";
    "¤ 000 ألف",
    "1000000-count-zero";
    "¤ 0 مليون",
    "1000000-count-one";
    "¤ 0 مليون",
    "1000000-count-two";
    "¤ 0 مليون",
    "1000000-count-few";
    "¤ 0 مليون",
    "1000000-count-many";
    "¤ 0 مليون",
    "1000000-count-other";
    "¤ 0 مليون",
    "10000000-count-zero";
    "¤ 00 مليون",

```



```

        "10000000-count-one";
        "¤ 00 مليون",
        "10000000-count-two";
        "¤ 00 مليون",
        "10000000-count-few";
        "¤ 00 مليون",
        "10000000-count-many";
        "¤ 00 مليون",
        "10000000-count-other";
        "¤ 00 مليون",
        "100000000-count-zero";
        "¤ 000 مليون",
        "100000000-count-one";
        "¤ 000 مليون",
        "100000000-count-two";
        "¤ 000 مليون",
        "100000000-count-few";
        "¤ 000 مليون",
        "100000000-count-many";
        "¤ 000 مليون",
        "100000000-count-other";
        "¤ 000 مليون",
        "1000000000-count-zero";
        "¤ 0 مليار",
        "1000000000-count-one";
        "¤ 0 مليار",
        "1000000000-count-two";
        "¤ 0 مليار",
        "1000000000-count-few";
        "¤ 0 مليار",
        "1000000000-count-many";
        "¤ 0 مليار",
        "1000000000-count-other";
        "¤ 0 مليار",
        "10000000000-count-zero";
        "¤ 00 مليار",
        "10000000000-count-one";
        "¤ 00 مليار",
        "10000000000-count-two";
        "¤ 00 مليار",
        "10000000000-count-few";
        "¤ 00 مليار",
        "10000000000-count-many";
        "¤ 00 مليار",
        "10000000000-count-other";
        "¤ 00 مليار",
        "100000000000-count-zero";
        "¤ 000 مليار",
        "100000000000-count-one";
        "¤ 000 مليار",
        "100000000000-count-two";
        "¤ 000 مليار",
        "100000000000-count-few";
        "¤ 000 مليار",
        "100000000000-count-many";
        "¤ 000 مليار",
        "100000000000-count-other";

```

```

        "٠ ٠٠٠ مليار",
        "1000000000000-count-zero";
        "٠ ترليون",
        "1000000000000-count-one";
        "٠ ترليون",
        "1000000000000-count-two";
        "٠ ترليون",
        "1000000000000-count-few";
        "٠ ترليون",
        "1000000000000-count-many";
        "٠ ترليون",
        "1000000000000-count-other";
        "٠ ترليون",
        "1000000000000-count-zero";
        "٠٠ ترليون",
        "1000000000000-count-one";
        "٠٠ ترليون",
        "1000000000000-count-two";
        "٠٠ ترليون",
        "1000000000000-count-few";
        "٠٠ ترليون",
        "1000000000000-count-many";
        "٠٠ ترليون",
        "1000000000000-count-other";
        "٠٠ ترليون",
        "1000000000000-count-zero";
        "٠٠٠ ترليون",
        "1000000000000-count-one";
        "٠٠٠ ترليون",
        "1000000000000-count-two";
        "٠٠٠ ترليون",
        "1000000000000-count-few";
        "٠٠٠ ترليون",
        "1000000000000-count-many";
        "٠٠٠ ترليون",
        "1000000000000-count-other";
        "٠٠٠ ترليون";
    }
}
"unitPattern-count-zero";
"{0} {1}",
    "unitPattern-count-one";
"{0} {1}",
    "unitPattern-count-two";
"{0} {1}",
    "unitPattern-count-few";
"{0} {1}",
    "unitPattern-count-many";
"{0} {1}",
    "unitPattern-count-other";
"{0} {1}";
}
"miscPatterns-numberSystem-arab";
{
    "atLeast";
    "+{0}",
    "range";
}

```

```

        "{0}-{1}";
    }
    "miscPatterns-numberSystem-latn";
    {
        "atLeast";
        "+{0}",
        "range";
        "{0}-{1}";
    }
    "minimalPairs";
    {
        "pluralMinimalPairs-count-zero";
        "{0} كتاب",
        "pluralMinimalPairs-count-one";
        "ولد واحد حضر",
        "pluralMinimalPairs-count-two";
        "ولدان حضرا",
        "pluralMinimalPairs-count-few";
        "{0} أولاد حضروا",
        "pluralMinimalPairs-count-many";
        "{0} ولدا حضروا",
        "pluralMinimalPairs-count-other";
        "{0} ولد حضروا",
        "other";
        "اتجه إلى المنعطف الـ {0} يميناً";
    }
    }
}

```

TIMEZONENAMES.JSON

```

{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "dates": {
        "timeZoneNames": {
          "hourFormat": "+HH:mm;-HH:mm",
          "gmtFormat": "0{غرينتش}",
          "gmtZeroFormat": "غرينتش",
          "regionFormat": "0{توقيت}",
          "regionFormat-type-daylight": "0{الصيفي} توقيت",
          "regionFormat-type-standard": "0{الرسمي} توقيت",
          "fallbackFormat": "{1} ({0})",
          "zone": {
            "America": {
              "Adak": {
                "exemplarCity": "أداك"
              }
            }
          }
        }
      }
    }
  }
}

```

```
    },  
    "Anchorage": {  
      "exemplarCity": "أنشوراج"  
    },  
    "Anguilla": {  
      "exemplarCity": "أنغويلا"  
    },  
    "Antigua": {  
      "exemplarCity": "أنتيغوا"  
    },  
    "Araguaina": {  
      "exemplarCity": "أروجوانيا"  
    },  
    "Argentina": {  
      "Rio_Gallegos": {  
        "exemplarCity": "ريو جالييوس"  
      },  
      "San_Juan": {  
        "exemplarCity": "سان خوان"  
      },  
      "Ushuaia": {  
        "exemplarCity": "أشوا"  
      },  
      "La_Rioja": {  
        "exemplarCity": "لا ريوجا"  
      },  
      "San_Luis": {  
        "exemplarCity": "سان لويس"  
      },  
      "Salta": {  
        "exemplarCity": "سالطا"  
      },  
      "Tucuman": {  
        "exemplarCity": "تاكمان"  
      }  
    },  
    "Aruba": {  
      "exemplarCity": "أروبا"  
    },  
    "Asuncion": {  
      "exemplarCity": "أسونسيون"  
    },  
    "Bahia": {  
      "exemplarCity": "باهيا"  
    },  
    "Bahia_Banderas": {  
      "exemplarCity": "باهيا بانديراس"  
    },  
    "Barbados": {  
      "exemplarCity": "بربادوس"  
    },  
    "Belem": {  
      "exemplarCity": "بلم"  
    },  
    "Belize": {  
      "exemplarCity": "بليز"  
    },  
  },  
}
```

```
"Blanc-Sablon": {
  "exemplarCity": "بلانك-سابلون"
},
"Boa_Vista": {
  "exemplarCity": "باو فيستا"
},
"Bogota": {
  "exemplarCity": "بوغوتا"
},
"Boise": {
  "exemplarCity": "بويس"
},
"Buenos_Aires": {
  "exemplarCity": "بوينوس آيرس"
},
"Cambridge_Bay": {
  "exemplarCity": "كامبرديج باي"
},
"Campo_Grande": {
  "exemplarCity": "كومبو جراند"
},
"Cancun": {
  "exemplarCity": "كانكون"
},
"Caracas": {
  "exemplarCity": "كاراكاس"
},
"Catamarca": {
  "exemplarCity": "كاتاماركا"
},
"Cayenne": {
  "exemplarCity": "كايين"
},
"Cayman": {
  "exemplarCity": "كايمان"
},
"Chicago": {
  "exemplarCity": "شيكاغو"
},
"Chihuahua": {
  "exemplarCity": "تشيوواوا"
},
"Coral_Harbour": {
  "exemplarCity": "كoral هاربر"
},
"Cordoba": {
  "exemplarCity": "كوردوبا"
},
"Costa_Rica": {
  "exemplarCity": "كوستاريكا"
},
"Creston": {
  "exemplarCity": "كريستون"
},
"Cuiaba": {
  "exemplarCity": "كيابا"
},
}
```

```
"Curacao": {
  "exemplarCity": "كوراساو"
},
"Danmarkshavn": {
  "exemplarCity": "دانمرك شافن"
},
"Dawson": {
  "exemplarCity": "داوسان"
},
"Dawson_Creek": {
  "exemplarCity": "داوسن كريك"
},
"Denver": {
  "exemplarCity": "دنفر"
},
"Detroit": {
  "exemplarCity": "ديترويت"
},
"Dominica": {
  "exemplarCity": "دومينيكا"
},
"Edmonton": {
  "exemplarCity": "ايدمونتون"
},
"Eirunepe": {
  "exemplarCity": "ايرونبي"
},
"El_Salvador": {
  "exemplarCity": "السلفادور"
},
"Fort_Nelson": {
  "exemplarCity": "فورت نيلسون"
},
"Fortaleza": {
  "exemplarCity": "فورتاليزا"
},
"Glace_Bay": {
  "exemplarCity": "جلاس باي"
},
"Godthab": {
  "exemplarCity": "غودثاب"
},
"Goose_Bay": {
  "exemplarCity": "جوس باي"
},
"Grand_Turk": {
  "exemplarCity": "غراند ترك"
},
"Grenada": {
  "exemplarCity": "جرينادا"
},
"Guadeloupe": {
  "exemplarCity": "غوادلوب"
},
"Guatemala": {
  "exemplarCity": "غواتيمالا"
},
}
```

```

"Guayaquil": {
  "exemplarCity": "غواياكويل"
},
"Guyana": {
  "exemplarCity": "غيانا"
},
"Halifax": {
  "exemplarCity": "هاليفاكس"
},
"Havana": {
  "exemplarCity": "هافانا"
},
"Hermosillo": {
  "exemplarCity": "هيرموسيلو"
},
"Indiana": {
  "Vincennes": {
    "exemplarCity": "فينسينس"
  },
  "Petersburg": {
    "exemplarCity": "بيتربيرغ"
  },
  "Tell_City": {
    "exemplarCity": "مدينة تل، إنديانا"
  },
  "Knox": {
    "exemplarCity": "كونكس"
  },
  "Winamac": {
    "exemplarCity": "ويناماك"
  },
  "Marengo": {
    "exemplarCity": "مارنجو"
  },
  "Vevay": {
    "exemplarCity": "فيفاي"
  }
},
"Indianapolis": {
  "exemplarCity": "إنديانا بوليس"
},
"Inuvik": {
  "exemplarCity": "اينوفيك"
},
"Iqaluit": {
  "exemplarCity": "اكويلت"
},
"Jamaica": {
  "exemplarCity": "جامايكا"
},
"Jujuy": {
  "exemplarCity": "جوجو"
},
"Juneau": {
  "exemplarCity": "جونى"
},
"Kentucky": {

```

```

    "Monticello": {
      "exemplarCity": "مونتي سيلو"
    },
    "Kralendijk": {
      "exemplarCity": "كرالنديك"
    },
    "La_Paz": {
      "exemplarCity": "لا باز"
    },
    "Lima": {
      "exemplarCity": "ليما"
    },
    "Los_Angeles": {
      "exemplarCity": "لوس انجلوس"
    },
    "Louisville": {
      "exemplarCity": "لويس فيل"
    },
    "Lower_Princes": {
      "exemplarCity": "حي الأمير السفلي"
    },
    "Maceio": {
      "exemplarCity": "ماشيو"
    },
    "Managua": {
      "exemplarCity": "ماناغوا"
    },
    "Manaus": {
      "exemplarCity": "ماناوس"
    },
    "Marigot": {
      "exemplarCity": "ماريغوت"
    },
    "Martinique": {
      "exemplarCity": "المارتينيك"
    },
    "Matamoros": {
      "exemplarCity": "ماتاموروس"
    },
    "Mazatlan": {
      "exemplarCity": "مازاتلان"
    },
    "Mendoza": {
      "exemplarCity": "ميندوزا"
    },
    "Menominee": {
      "exemplarCity": "مينوميني"
    },
    "Merida": {
      "exemplarCity": "ميريديا"
    },
    "Metlakatla": {
      "exemplarCity": "ميتلاكاتلا"
    },
    "Mexico_City": {
      "exemplarCity": "مدينة المكسيك"
    }
  }

```



```

    },
    "Miquelon": {
      "exemplarCity": "مكويلون"
    },
    "Moncton": {
      "exemplarCity": "وينكتون"
    },
    "Monterrey": {
      "exemplarCity": "مونتيري"
    },
    "Montevideo": {
      "exemplarCity": "مونتيفيديو"
    },
    "Montserrat": {
      "exemplarCity": "مونتسيرات"
    },
    "Nassau": {
      "exemplarCity": "ناسو"
    },
    "New_York": {
      "exemplarCity": "نيويورك"
    },
    "Nipigon": {
      "exemplarCity": "نيبيجون"
    },
    "Nome": {
      "exemplarCity": "نوم"
    },
    "Noronha": {
      "exemplarCity": "نوروناه"
    },
    "North_Dakota": {
      "Beulah": {
        "exemplarCity": "بيولا، داكوتا الشمالية"
      },
      "New_Salem": {
        "exemplarCity": "نيو ساليم"
      },
      "Center": {
        "exemplarCity": "سنتر"
      }
    },
    "Ojinaga": {
      "exemplarCity": "أوجيناغا"
    },
    "Panama": {
      "exemplarCity": "بنما"
    },
    "Pangnirtung": {
      "exemplarCity": "بانجينتينج"
    },
    "Paramaribo": {
      "exemplarCity": "باراماريبو"
    },
    "Phoenix": {
      "exemplarCity": "فينكس"
    },
  },

```

```

"Port-au-Prince": {
  "exemplarCity": "بورت أو برنس"
},
"Port_of_Spain": {
  "exemplarCity": "بورت أوف سبين"
},
"Porto_Velho": {
  "exemplarCity": "بورتو فيلو"
},
"Puerto_Rico": {
  "exemplarCity": "بورتوريكو"
},
"Punta_Arenas": {
  "exemplarCity": "بونتأ أريناز"
},
"Rainy_River": {
  "exemplarCity": "راني ريفر"
},
"Rankin_Inlet": {
  "exemplarCity": "رانكن انلت"
},
"Recife": {
  "exemplarCity": "ريسيف"
},
"Regina": {
  "exemplarCity": "ريجينا"
},
"Resolute": {
  "exemplarCity": "ريزولوت"
},
"Rio_Branco": {
  "exemplarCity": "ريوبرانكو"
},
"Santa_Isabel": {
  "exemplarCity": "سانتا إيزابيل"
},
"Santarem": {
  "exemplarCity": "سانتاريم"
},
"Santiago": {
  "exemplarCity": "سانتياغو"
},
"Santo_Domingo": {
  "exemplarCity": "سانتو دومينغو"
},
"Sao_Paulo": {
  "exemplarCity": "ساو باولو"
},
"Scoresbysund": {
  "exemplarCity": "سكورسبيسند"
},
"Sitka": {
  "exemplarCity": "سيتكا"
},
"St_Barthelemy": {
  "exemplarCity": "سانت بارتيليمي"
},

```

```

    "St_Johns": {
      "exemplarCity": "سانت جونس"
    },
    "St_Kitts": {
      "exemplarCity": "سانت كيتس"
    },
    "St_Lucia": {
      "exemplarCity": "سانت لوشيا"
    },
    "St_Thomas": {
      "exemplarCity": "سانت توماس"
    },
    "St_Vincent": {
      "exemplarCity": "سانت فنسنت"
    },
    "Swift_Current": {
      "exemplarCity": "سوفت كارنت"
    },
    "Tegucigalpa": {
      "exemplarCity": "تيغوسيغالبا"
    },
    "Thule": {
      "exemplarCity": "ثيل"
    },
    "Thunder_Bay": {
      "exemplarCity": "ثندر باي"
    },
    "Tijuana": {
      "exemplarCity": "تيخوانا"
    },
    "Toronto": {
      "exemplarCity": "تورونتو"
    },
    "Tortola": {
      "exemplarCity": "تورتولا"
    },
    "Vancouver": {
      "exemplarCity": "فانكوفر"
    },
    "Whitehorse": {
      "exemplarCity": "وايت هورس"
    },
    "Winnipeg": {
      "exemplarCity": "وينيبيج"
    },
    "Yakutat": {
      "exemplarCity": "ياكوتات"
    },
    "Yellowknife": {
      "exemplarCity": "يلونيف"
    }
  },
  "Atlantic": {
    "Azores": {
      "exemplarCity": "أزورس"
    },
    "Bermuda": {

```

```

        "exemplarCity": "برمودا"
      },
      "Canary": {
        "exemplarCity": "كناري"
      },
      "Cape_Verde": {
        "exemplarCity": "الرأس الأخضر"
      },
      "Faeroe": {
        "exemplarCity": "فارو"
      },
      "Madeira": {
        "exemplarCity": "ماديرا"
      },
      "Reykjavik": {
        "exemplarCity": "ريكيافيك"
      },
      "South_Georgia": {
        "exemplarCity": "جورجيا الجنوبية"
      },
      "St_Helena": {
        "exemplarCity": "سانت هيلينا"
      },
      "Stanley": {
        "exemplarCity": "استانلي"
      }
    },
    "Europe": {
      "Amsterdam": {
        "exemplarCity": "أمستردام"
      },
      "Andorra": {
        "exemplarCity": "أندورا"
      },
      "Astrakhan": {
        "exemplarCity": "أستراخان"
      },
      "Athens": {
        "exemplarCity": "أثينا"
      },
      "Belgrade": {
        "exemplarCity": "بلغراد"
      },
      "Berlin": {
        "exemplarCity": "برلين"
      },
      "Bratislava": {
        "exemplarCity": "براتيسلافا"
      },
      "Brussels": {
        "exemplarCity": "بروكسل"
      },
      "Bucharest": {
        "exemplarCity": "بوخارست"
      },
      "Budapest": {
        "exemplarCity": "بودابست"
      }
    }
  }

```

```
    },  
    "Busingen": {  
      "exemplarCity": "بوسنغن"  
    },  
    "Chisinau": {  
      "exemplarCity": "تشيسيناو"  
    },  
    "Copenhagen": {  
      "exemplarCity": "كوبنهاغن"  
    },  
    "Dublin": {  
      "long": {  
        "daylight": "توقيت أيرلندا الرسمي"  
      },  
      "exemplarCity": "دبلن"  
    },  
    "Gibraltar": {  
      "exemplarCity": "جبل طارق"  
    },  
    "Guernsey": {  
      "exemplarCity": "غيرنسي"  
    },  
    "Helsinki": {  
      "exemplarCity": "هلسنكي"  
    },  
    "Isle_of_Man": {  
      "exemplarCity": "جزيرة مان"  
    },  
    "Istanbul": {  
      "exemplarCity": "إسطنبول"  
    },  
    "Jersey": {  
      "exemplarCity": "جيرسي"  
    },  
    "Kaliningrad": {  
      "exemplarCity": "كالينجراد"  
    },  
    "Kiev": {  
      "exemplarCity": "كييف"  
    },  
    "Kirov": {  
      "exemplarCity": "كيروف"  
    },  
    "Lisbon": {  
      "exemplarCity": "لشبونة"  
    },  
    "Ljubljana": {  
      "exemplarCity": "ليوبليانا"  
    },  
    "London": {  
      "long": {  
        "daylight": "توقيت بريطانيا الصيفي"  
      },  
      "exemplarCity": "لندن"  
    },  
    "Luxembourg": {  
      "exemplarCity": "لوكسمبورغ"
```

```
    },  
    "Madrid": {  
      "exemplarCity": "مدريد"  
    },  
    "Malta": {  
      "exemplarCity": "مالمطة"  
    },  
    "Mariehamn": {  
      "exemplarCity": "ماريهامن"  
    },  
    "Minsk": {  
      "exemplarCity": "مينسك"  
    },  
    "Monaco": {  
      "exemplarCity": "موناكو"  
    },  
    "Moscow": {  
      "exemplarCity": "موسكو"  
    },  
    "Oslo": {  
      "exemplarCity": "أوسلو"  
    },  
    "Paris": {  
      "exemplarCity": "باريس"  
    },  
    "Podgorica": {  
      "exemplarCity": "بودغوريكا"  
    },  
    "Prague": {  
      "exemplarCity": "براغ"  
    },  
    "Riga": {  
      "exemplarCity": "ريغا"  
    },  
    "Rome": {  
      "exemplarCity": "روما"  
    },  
    "Samara": {  
      "exemplarCity": "سمراء"  
    },  
    "San_Marino": {  
      "exemplarCity": "سان مارينو"  
    },  
    "Sarajevo": {  
      "exemplarCity": "سراييفو"  
    },  
    "Saratov": {  
      "exemplarCity": "ساراتوف"  
    },  
    "Simferopol": {  
      "exemplarCity": "سيمفروبول"  
    },  
    "Skopje": {  
      "exemplarCity": "سكوبي"  
    },  
    "Sofia": {  
      "exemplarCity": "صوفيا"
```

```
    },
    "Stockholm": {
      "exemplarCity": "ستوكهولم"
    },
    "Tallinn": {
      "exemplarCity": "تالين"
    },
    "Tirane": {
      "exemplarCity": "تيرانا"
    },
    "Ulyanovsk": {
      "exemplarCity": "أوليانوفسك"
    },
    "Uzhgorod": {
      "exemplarCity": "أوزجروود"
    },
    "Vaduz": {
      "exemplarCity": "فادوز"
    },
    "Vatican": {
      "exemplarCity": "الفاتيكان"
    },
    "Vienna": {
      "exemplarCity": "فيينا"
    },
    "Vilnius": {
      "exemplarCity": "فيلنيوس"
    },
    "Volgograd": {
      "exemplarCity": "فولجوراد"
    },
    "Warsaw": {
      "exemplarCity": "وارسو"
    },
    "Zagreb": {
      "exemplarCity": "زغرب"
    },
    "Zaporozhye": {
      "exemplarCity": "زابوروزي"
    },
    "Zurich": {
      "exemplarCity": "زيورخ"
    }
  },
  "Africa": {
    "Abidjan": {
      "exemplarCity": "أبيدجان"
    },
    "Accra": {
      "exemplarCity": "أكرا"
    },
    "Addis_Ababa": {
      "exemplarCity": "أديس أبابا"
    },
    "Algiers": {
      "exemplarCity": "الجزائر"
    }
  },
```

```
"Asmera": {
  "exemplarCity": "أسمرّة"
},
"Bamako": {
  "exemplarCity": "باماكو"
},
"Bangui": {
  "exemplarCity": "بانغوي"
},
"Banjul": {
  "exemplarCity": "بانجول"
},
"Bissau": {
  "exemplarCity": "بيساو"
},
"Blantyre": {
  "exemplarCity": "بلانتيير"
},
"Brazzaville": {
  "exemplarCity": "برازافيل"
},
"Bujumbura": {
  "exemplarCity": "بوجومبورا"
},
"Cairo": {
  "exemplarCity": "القاهرة"
},
"Casablanca": {
  "exemplarCity": "الدار البيضاء"
},
"Ceuta": {
  "exemplarCity": "سيّتا"
},
"Conakry": {
  "exemplarCity": "كوناكري"
},
"Dakar": {
  "exemplarCity": "داكار"
},
"Dar_es_Salaam": {
  "exemplarCity": "دار السلام"
},
"Djibouti": {
  "exemplarCity": "جيبوتي"
},
"Douala": {
  "exemplarCity": "دوالا"
},
"El_Aaiun": {
  "exemplarCity": "العيون"
},
"Freetown": {
  "exemplarCity": "فري تاون"
},
"Gaborone": {
  "exemplarCity": "غابورون"
},
}
```



```
"Harare": {
  "exemplarCity": "هراري"
},
"Johannesburg": {
  "exemplarCity": "جوهانسبرغ"
},
"Juba": {
  "exemplarCity": "جوبا"
},
"Kampala": {
  "exemplarCity": "كامبالا"
},
"Khartoum": {
  "exemplarCity": "الخرطوم"
},
"Kigali": {
  "exemplarCity": "كيغالي"
},
"Kinshasa": {
  "exemplarCity": "كينشاسا"
},
"Lagos": {
  "exemplarCity": "لاغوس"
},
"Libreville": {
  "exemplarCity": "ليبرفيل"
},
"Lome": {
  "exemplarCity": "لومي"
},
"Luanda": {
  "exemplarCity": "لواندا"
},
"Lubumbashi": {
  "exemplarCity": "لومبباشا"
},
"Lusaka": {
  "exemplarCity": "لوساكا"
},
"Malabo": {
  "exemplarCity": "مالابو"
},
"Maputo": {
  "exemplarCity": "مابوتو"
},
"Maseru": {
  "exemplarCity": "ماسيرو"
},
"Mbabane": {
  "exemplarCity": "مباباني"
},
"Mogadishu": {
  "exemplarCity": "مقديشو"
},
"Monrovia": {
  "exemplarCity": "مونروفيا"
},
},
```

```

"Nairobi": {
  "exemplarCity": "نairobi"
},
"Ndjamena": {
  "exemplarCity": "نجامينا"
},
"Niamey": {
  "exemplarCity": "نيامي"
},
"Nouakchott": {
  "exemplarCity": "نواكشوط"
},
"Ouagadougou": {
  "exemplarCity": "واغادوغو"
},
"Porto-Novo": {
  "exemplarCity": "بورتو نوفو"
},
"Sao_Tome": {
  "exemplarCity": "ساو تومي"
},
"Tripoli": {
  "exemplarCity": "طرابلس"
},
"Tunis": {
  "exemplarCity": "تونس"
},
"Windhoek": {
  "exemplarCity": "ويندهوك"
}
},
"Asia": {
  "Aden": {
    "exemplarCity": "عدن"
  },
  "Almaty": {
    "exemplarCity": "ألماتي"
  },
  "Amman": {
    "exemplarCity": "عمان"
  },
  "Anadyr": {
    "exemplarCity": "أندير"
  },
  "Aqtan": {
    "exemplarCity": "أكتان"
  },
  "Aqtobe": {
    "exemplarCity": "أكتوب"
  },
  "Ashgabat": {
    "exemplarCity": "عشق آباد"
  },
  "Atyrau": {
    "exemplarCity": "أтираو"
  },
  "Baghdad": {

```

```
    "exemplarCity": "بغداد"
  },
  "Bahrain": {
    "exemplarCity": "البحرين"
  },
  "Baku": {
    "exemplarCity": "باكو"
  },
  "Bangkok": {
    "exemplarCity": "بانكوك"
  },
  "Barnaul": {
    "exemplarCity": "بارناول"
  },
  "Beirut": {
    "exemplarCity": "بيروت"
  },
  "Bishkek": {
    "exemplarCity": "بشكيك"
  },
  "Brunei": {
    "exemplarCity": "بروناي"
  },
  "Calcutta": {
    "exemplarCity": "كالكتا"
  },
  "Chita": {
    "exemplarCity": "تشيتا"
  },
  "Choibalsan": {
    "exemplarCity": "تشوبالسان"
  },
  "Colombo": {
    "exemplarCity": "كولومبو"
  },
  "Damascus": {
    "exemplarCity": "دمشق"
  },
  "Dhaka": {
    "exemplarCity": "دكا"
  },
  "Dili": {
    "exemplarCity": "ديلي"
  },
  "Dubai": {
    "exemplarCity": "دبي"
  },
  "Dushanbe": {
    "exemplarCity": "دوشانبي"
  },
  "Famagusta": {
    "exemplarCity": "فاماغوستا"
  },
  "Gaza": {
    "exemplarCity": "غزة"
  },
  "Hebron": {
```

```

    "exemplarCity": "(هيبرون) مدينة الخليل"
  },
  "Hong_Kong": {
    "exemplarCity": "هونغ كونغ"
  },
  "Hovd": {
    "exemplarCity": "هوفد"
  },
  "Irkutsk": {
    "exemplarCity": "ايركيتسك"
  },
  "Jakarta": {
    "exemplarCity": "جاكرتا"
  },
  "Jayapura": {
    "exemplarCity": "جايا بورا"
  },
  "Jerusalem": {
    "exemplarCity": "القدس"
  },
  "Kabul": {
    "exemplarCity": "كابول"
  },
  "Kamchatka": {
    "exemplarCity": "كامتشاتكا"
  },
  "Karachi": {
    "exemplarCity": "كراتشي"
  },
  "Katmandu": {
    "exemplarCity": "كاتماندو"
  },
  "Khandyga": {
    "exemplarCity": "خانديجا"
  },
  "Krasnoyarsk": {
    "exemplarCity": "كراسنويارسك"
  },
  "Kuala_Lumpur": {
    "exemplarCity": "كوالا لامبور"
  },
  "Kuching": {
    "exemplarCity": "كيشينج"
  },
  "Kuwait": {
    "exemplarCity": "الكويت"
  },
  "Macau": {
    "exemplarCity": "ماكاو"
  },
  "Magadan": {
    "exemplarCity": "مجادن"
  },
  "Makassar": {
    "exemplarCity": "ماكسار"
  },
  "Manila": {

```

```
    "exemplarCity": "مانیلا"
  },
  "Muscat": {
    "exemplarCity": "مسقط"
  },
  "Nicosia": {
    "exemplarCity": "نیقوسیا"
  },
  "Novokuznetsk": {
    "exemplarCity": "نوفوکوزنتسک"
  },
  "Novosibirsk": {
    "exemplarCity": "نوفوسبیرسک"
  },
  "Omsk": {
    "exemplarCity": "أومسک"
  },
  "Oral": {
    "exemplarCity": "أورال"
  },
  "Phnom_Penh": {
    "exemplarCity": "بنوم بنه"
  },
  "Pontianak": {
    "exemplarCity": "بونتیانک"
  },
  "Pyongyang": {
    "exemplarCity": "بیونگ یانگ"
  },
  "Qatar": {
    "exemplarCity": "قطر"
  },
  "Qyzylorda": {
    "exemplarCity": "کیزیلوردا"
  },
  "Rangoon": {
    "exemplarCity": "رانگون"
  },
  "Riyadh": {
    "exemplarCity": "الریاض"
  },
  "Saigon": {
    "exemplarCity": "مدینة هو تشي منه"
  },
  "Sakhalin": {
    "exemplarCity": "سکالین"
  },
  "Samarkand": {
    "exemplarCity": "سمرقند"
  },
  "Seoul": {
    "exemplarCity": "سول"
  },
  "Shanghai": {
    "exemplarCity": "شنغهاي"
  },
  "Singapore": {
```

```
    "exemplarCity": "سنغافورة"
  },
  "Srednekolymsk": {
    "exemplarCity": "سريدنكوليمسك"
  },
  "Taipei": {
    "exemplarCity": "تايبيه"
  },
  "Tashkent": {
    "exemplarCity": "تشقند"
  },
  "Tbilisi": {
    "exemplarCity": "تبليسي"
  },
  "Tehran": {
    "exemplarCity": "تهران"
  },
  "Thimphu": {
    "exemplarCity": "تيمفو"
  },
  "Tokyo": {
    "exemplarCity": "طوكيو"
  },
  "Tomsk": {
    "exemplarCity": "تومسك"
  },
  "Ulaanbaatar": {
    "exemplarCity": "آلانباتار"
  },
  "Urumqi": {
    "exemplarCity": "أرومكي"
  },
  "Ust-Nera": {
    "exemplarCity": "أوست نيرا"
  },
  "Vientiane": {
    "exemplarCity": "فيانتيان"
  },
  "Vladivostok": {
    "exemplarCity": "فلاديفوستك"
  },
  "Yakutsk": {
    "exemplarCity": "ياكتسك"
  },
  "Yekaterinburg": {
    "exemplarCity": "يكاترينبيرج"
  },
  "Yerevan": {
    "exemplarCity": "يريفان"
  }
},
"Indian": {
  "Antananarivo": {
    "exemplarCity": "أنتاناناريفو"
  },
  "Chagos": {
    "exemplarCity": "تشاغوس"
  }
}
```

```
    },  
    "Christmas": {  
      "exemplarCity": "كريسماس"  
    },  
    "Cocos": {  
      "exemplarCity": "كوكوس"  
    },  
    "Comoro": {  
      "exemplarCity": "جزر القمر"  
    },  
    "Kerguelen": {  
      "exemplarCity": "كيرغويلين"  
    },  
    "Mahe": {  
      "exemplarCity": "ماهي"  
    },  
    "Maldives": {  
      "exemplarCity": "المالديف"  
    },  
    "Mauritius": {  
      "exemplarCity": "موريشيوس"  
    },  
    "Mayotte": {  
      "exemplarCity": "مايوت"  
    },  
    "Reunion": {  
      "exemplarCity": "ريونيون"  
    }  
  },  
  "Australia": {  
    "Adelaide": {  
      "exemplarCity": "أديليد"  
    },  
    "Brisbane": {  
      "exemplarCity": "برسبان"  
    },  
    "Broken_Hill": {  
      "exemplarCity": "بروكن هيل"  
    },  
    "Currie": {  
      "exemplarCity": "كوري"  
    },  
    "Darwin": {  
      "exemplarCity": "دارون"  
    },  
    "Eucla": {  
      "exemplarCity": "أو كلا"  
    },  
    "Hobart": {  
      "exemplarCity": "هوبارت"  
    },  
    "Lindeman": {  
      "exemplarCity": "ليندمان"  
    },  
    "Lord_Howe": {  
      "exemplarCity": "لورد هاو"  
    }  
  },  
}
```

```
"Melbourne": {
  "exemplarCity": "ميلبورن"
},
"Perth": {
  "exemplarCity": "برثا"
},
"Sydney": {
  "exemplarCity": "سيدني"
}
},
"Pacific": {
  "Apia": {
    "exemplarCity": "أبيا"
  },
  "Auckland": {
    "exemplarCity": "أوكلاند"
  },
  "Bougainville": {
    "exemplarCity": "بوغانفيل"
  },
  "Chatham": {
    "exemplarCity": "تشاثام"
  },
  "Easter": {
    "exemplarCity": "استر"
  },
  "Efate": {
    "exemplarCity": "إيفات"
  },
  "Enderbury": {
    "exemplarCity": "اندربيرج"
  },
  "Fakaofo": {
    "exemplarCity": "فاكاوفو"
  },
  "Fiji": {
    "exemplarCity": "فيجي"
  },
  "Funafuti": {
    "exemplarCity": "فونافوتي"
  },
  "Galapagos": {
    "exemplarCity": "جلاپاجوس"
  },
  "Gambier": {
    "exemplarCity": "جامبير"
  },
  "Guadalcanal": {
    "exemplarCity": "غواد الكانال"
  },
  "Guam": {
    "exemplarCity": "غوام"
  },
  "Honolulu": {
    "exemplarCity": "هونولولو"
  },
  "Johnston": {
```



```
    "exemplarCity": "جونستون"
  },
  "Kiritimati": {
    "exemplarCity": "كيريتي ماتي"
  },
  "Kosrae": {
    "exemplarCity": "كوسرا"
  },
  "Kwajalein": {
    "exemplarCity": "كواجالين"
  },
  "Majuro": {
    "exemplarCity": "ماجورو"
  },
  "Marquesas": {
    "exemplarCity": "ماركيساس"
  },
  "Midway": {
    "exemplarCity": "ميدواي"
  },
  "Nauru": {
    "exemplarCity": "ناورو"
  },
  "Niue": {
    "exemplarCity": "نيوي"
  },
  "Norfolk": {
    "exemplarCity": "نورفولك"
  },
  "Noumea": {
    "exemplarCity": "نوميا"
  },
  "Pago_Pago": {
    "exemplarCity": "باغو باغو"
  },
  "Palau": {
    "exemplarCity": "بالاو"
  },
  "Pitcairn": {
    "exemplarCity": "بيتكيرن"
  },
  "Ponape": {
    "exemplarCity": "باناب"
  },
  "Port_Moresby": {
    "exemplarCity": "بور مورسبي"
  },
  "Rarotonga": {
    "exemplarCity": "راروتونغا"
  },
  "Saipan": {
    "exemplarCity": "سايبان"
  },
  "Tahiti": {
    "exemplarCity": "تاهيتي"
  },
  "Tarawa": {
```

```
    "exemplarCity": "تاراوا",
  },
  "Tongatapu": {
    "exemplarCity": "تونغاتابو",
  },
  "Truk": {
    "exemplarCity": "ترك",
  },
  "Wake": {
    "exemplarCity": "واك",
  },
  "Wallis": {
    "exemplarCity": "واليس",
  },
},
"Arctic": {
  "Longyearbyen": {
    "exemplarCity": "لونجيربين",
  },
},
"Antarctica": {
  "Casey": {
    "exemplarCity": "كاساي",
  },
  "Davis": {
    "exemplarCity": "دافيز",
  },
  "DumontDUrville": {
    "exemplarCity": "دي مونت دو روفيل",
  },
  "Macquarie": {
    "exemplarCity": "ماكوارى",
  },
  "Mawson": {
    "exemplarCity": "ماوسون",
  },
  "McMurdo": {
    "exemplarCity": "ماك موردو",
  },
  "Palmer": {
    "exemplarCity": "بالمير",
  },
  "Rothera": {
    "exemplarCity": "روثيرا",
  },
  "Syowa": {
    "exemplarCity": "سايووا",
  },
  "Troll": {
    "exemplarCity": "ترول",
  },
  "Vostok": {
    "exemplarCity": "فوستوك",
  },
},
"Etc": {
  "UTC": {
```

```

        "long": {
            "standard": "التوقيت العالمي المنسق"
        },
        "short": {
            "standard": "UTC"
        }
    },
    "Unknown": {
        "exemplarCity": "مدينة غير معروفة"
    }
},
"metazone": {
    "Afghanistan": {
        "long": {
            "standard": "توقيت أفغانستان"
        }
    },
    "Africa_Central": {
        "long": {
            "standard": "توقيت وسط أفريقيا"
        }
    },
    "Africa_Eastern": {
        "long": {
            "standard": "توقيت شرق أفريقيا"
        }
    },
    "Africa_Southern": {
        "long": {
            "standard": "توقيت جنوب أفريقيا"
        }
    },
    "Africa_Western": {
        "long": {
            "generic": "توقيت غرب أفريقيا",
            "standard": "توقيت غرب أفريقيا الرسمي",
            "daylight": "توقيت غرب أفريقيا الصيفي"
        }
    },
    "Alaska": {
        "long": {
            "generic": "توقيت ألاسكا",
            "standard": "التوقيت الرسمي لألاسكا",
            "daylight": "توقيت ألاسكا الصيفي"
        }
    },
    "Amazon": {
        "long": {
            "generic": "توقيت الأمازون",
            "standard": "توقيت الأمازون الرسمي",
            "daylight": "توقيت الأمازون الصيفي"
        }
    },
    "America_Central": {
        "long": {
            "generic": "التوقيت المركزي لأمريكا الشمالية",

```

```

        "standard": "التوقيت الرسمي المركزي لأمريكا الشمالية",
        "daylight": "التوقيت الصيفي المركزي لأمريكا الشمالية"
    },
    },
    "America_Eastern": {
        "long": {
            "generic": "التوقيت الشرقي لأمريكا الشمالية",
            "standard": "التوقيت الرسمي الشرقي لأمريكا الشمالية",
            "daylight": "التوقيت الصيفي الشرقي لأمريكا الشمالية"
        }
    },
    "America_Mountain": {
        "long": {
            "generic": "التوقيت الجبلي لأمريكا الشمالية",
            "standard": "التوقيت الجبلي الرسمي لأمريكا الشمالية",
            "daylight": "التوقيت الجبلي الصيفي لأمريكا الشمالية"
        }
    },
    "America_Pacific": {
        "long": {
            "generic": "توقيت المحيط الهادي",
            "standard": "توقيت المحيط الهادي الرسمي",
            "daylight": "توقيت المحيط الهادي الصيفي"
        }
    },
    "Anadyr": {
        "long": {
            "generic": "توقيت أنادير",
            "standard": "توقيت أنادير الرسمي",
            "daylight": "التوقيت الصيفي لأنادير"
        }
    },
    "Apia": {
        "long": {
            "generic": "توقيت آبيا",
            "standard": "التوقيت الرسمي لآبيا",
            "daylight": "التوقيت الصيفي لآبيا"
        }
    },
    "Arabian": {
        "long": {
            "generic": "التوقيت العربي",
            "standard": "التوقيت العربي الرسمي",
            "daylight": "التوقيت العربي الصيفي"
        }
    },
    "Argentina": {
        "long": {
            "generic": "توقيت الأرجنتين",
            "standard": "توقيت الأرجنتين الرسمي",
            "daylight": "توقيت الأرجنتين الصيفي"
        }
    },
    "Argentina_Western": {
        "long": {
            "generic": "توقيت غرب الأرجنتين",
            "standard": "توقيت غرب الأرجنتين الرسمي",

```

```

        "daylight": "توقيت غرب الأرجنتين الصيفي"
    },
},
"Armenia": {
    "long": {
        "generic": "توقيت أرمينيا",
        "standard": "توقيت أرمينيا الرسمي",
        "daylight": "توقيت أرمينيا الصيفي"
    }
},
"Atlantic": {
    "long": {
        "generic": "توقيت الأطلسي",
        "standard": "التوقيت الرسمي الأطلسي",
        "daylight": "التوقيت الصيفي الأطلسي"
    }
},
"Australia_Central": {
    "long": {
        "generic": "توقيت وسط أستراليا",
        "standard": "توقيت وسط أستراليا الرسمي",
        "daylight": "توقيت وسط أستراليا الصيفي"
    }
},
"Australia_CentralWestern": {
    "long": {
        "generic": "توقيت غرب وسط أستراليا",
        "standard": "توقيت غرب وسط أستراليا الرسمي",
        "daylight": "توقيت غرب وسط أستراليا الصيفي"
    }
},
"Australia_Eastern": {
    "long": {
        "generic": "توقيت شرق أستراليا",
        "standard": "توقيت شرق أستراليا الرسمي",
        "daylight": "توقيت شرق أستراليا الصيفي"
    }
},
"Australia_Western": {
    "long": {
        "generic": "توقيت غرب أستراليا",
        "standard": "توقيت غرب أستراليا الرسمي",
        "daylight": "توقيت غرب أستراليا الصيفي"
    }
},
"Azerbaijan": {
    "long": {
        "generic": "توقيت أذربيجان",
        "standard": "توقيت أذربيجان الرسمي",
        "daylight": "توقيت أذربيجان الصيفي"
    }
},
"Azores": {
    "long": {
        "generic": "توقيت أزورس",
        "standard": "توقيت أزورس الرسمي",
        "daylight": "توقيت أزورس الصيفي"
    }
}

```

```

    }
  },
  "Bangladesh": {
    "long": {
      "generic": "توقيت بنغلاديش",
      "standard": "توقيت بنغلاديش الرسمي",
      "daylight": "توقيت بنغلاديش الصيفي"
    }
  },
  "Bhutan": {
    "long": {
      "standard": "توقيت بوتان"
    }
  },
  "Bolivia": {
    "long": {
      "standard": "توقيت بوليفيا"
    }
  },
  "Brasilia": {
    "long": {
      "generic": "توقيت برازيليا",
      "standard": "توقيت برازيليا الرسمي",
      "daylight": "توقيت برازيليا الصيفي"
    }
  },
  "Brunei": {
    "long": {
      "standard": "توقيت بروناي"
    }
  },
  "Cape_Verde": {
    "long": {
      "generic": "توقيت الرأس الأخضر",
      "standard": "توقيت الرأس الأخضر الرسمي",
      "daylight": "توقيت الرأس الأخضر الصيفي"
    }
  },
  "Chamorro": {
    "long": {
      "standard": "توقيت تشامورو"
    }
  },
  "Chatham": {
    "long": {
      "generic": "توقيت تشاتام",
      "standard": "توقيت تشاتام الرسمي",
      "daylight": "توقيت تشاتام الصيفي"
    }
  },
  "Chile": {
    "long": {
      "generic": "توقيت شيلي",
      "standard": "توقيت شيلي الرسمي",
      "daylight": "توقيت شيلي الصيفي"
    }
  },
},

```

```

"China": {
  "long": {
    "generic": "توقيت الصين",
    "standard": "توقيت الصين الرسمي",
    "daylight": "توقيت الصين الصيفي"
  }
},
"Choibalsan": {
  "long": {
    "generic": "توقيت شويبالسان",
    "standard": "توقيت شويبالسان الرسمي",
    "daylight": "التوقيت الصيفي لشويبالسان"
  }
},
"Christmas": {
  "long": {
    "standard": "توقيت جزر الكريسماس"
  }
},
"Cocos": {
  "long": {
    "standard": "توقيت جزر كوكوس"
  }
},
"Colombia": {
  "long": {
    "generic": "توقيت كولومبيا",
    "standard": "توقيت كولومبيا الرسمي",
    "daylight": "توقيت كولومبيا الصيفي"
  }
},
"Cook": {
  "long": {
    "generic": "توقيت جزر كوك",
    "standard": "توقيت جزر كوك الرسمي",
    "daylight": "توقيت جزر كوك الصيفي"
  }
},
"Cuba": {
  "long": {
    "generic": "توقيت كوبا",
    "standard": "توقيت كوبا الرسمي",
    "daylight": "توقيت كوبا الصيفي"
  }
},
"Davis": {
  "long": {
    "standard": "توقيت دافيز"
  }
},
"DumontDUrville": {
  "long": {
    "standard": "توقيت دي مونت دو روفيل"
  }
},
"East_Timor": {
  "long": {

```

```

        "standard": "توقيت تيمور الشرقية"
    },
    },
    "Easter": {
        "long": {
            "generic": "توقيت جزيرة استر",
            "standard": "توقيت جزيرة استر الرسمي",
            "daylight": "توقيت جزيرة استر الصيفي"
        }
    },
    "Ecuador": {
        "long": {
            "standard": "توقيت الإكوادور"
        }
    },
    "Europe_Central": {
        "long": {
            "generic": "توقيت وسط أوروبا",
            "standard": "توقيت وسط أوروبا الرسمي",
            "daylight": "توقيت وسط أوروبا الصيفي"
        }
    },
    "Europe_Eastern": {
        "long": {
            "generic": "توقيت شرق أوروبا",
            "standard": "توقيت شرق أوروبا الرسمي",
            "daylight": "توقيت شرق أوروبا الصيفي"
        }
    },
    "Europe_Further_Eastern": {
        "long": {
            "standard": "التوقيت الأوروبي (أكثر شرقًا)"
        }
    },
    "Europe_Western": {
        "long": {
            "generic": "توقيت غرب أوروبا",
            "standard": "توقيت غرب أوروبا الرسمي",
            "daylight": "توقيت غرب أوروبا الصيفي"
        }
    },
    "Falkland": {
        "long": {
            "generic": "توقيت جزر فوكلاند",
            "standard": "توقيت جزر فوكلاند الرسمي",
            "daylight": "توقيت جزر فوكلاند الصيفي"
        }
    },
    "Fiji": {
        "long": {
            "generic": "توقيت فيجي",
            "standard": "توقيت فيجي الرسمي",
            "daylight": "توقيت فيجي الصيفي"
        }
    },
    "French_Guiana": {
        "long": {

```



```

        "standard": "توقيت غايانا الفرنسية"
    },
    },
    "French_Southern": {
        "long": {
            "standard": "توقيت المقاطعات الفرنسية الجنوبية والأنتارتيكية"
        }
    },
    "Galapagos": {
        "long": {
            "standard": "توقيت غلاباغوس"
        }
    },
    "Gambier": {
        "long": {
            "standard": "توقيت جامبير"
        }
    },
    "Georgia": {
        "long": {
            "generic": "توقيت جورجيا",
            "standard": "توقيت جورجيا الرسمي",
            "daylight": "توقيت جورجيا الصيفي"
        }
    },
    "Gilbert_Islands": {
        "long": {
            "standard": "توقيت جزر جيلبرت"
        }
    },
    "GMT": {
        "long": {
            "standard": "توقيت غرينتش"
        }
    },
    "Greenland_Eastern": {
        "long": {
            "generic": "توقيت شرق غرينلاند",
            "standard": "توقيت شرق غرينلاند الرسمي",
            "daylight": "توقيت شرق غرينلاند الصيفي"
        }
    },
    "Greenland_Western": {
        "long": {
            "generic": "توقيت غرب غرينلاند",
            "standard": "توقيت غرب غرينلاند الرسمي",
            "daylight": "توقيت غرب غرينلاند الصيفي"
        }
    },
    "Guam": {
        "long": {
            "standard": "توقيت غوام"
        }
    },
    "Gulf": {
        "long": {
            "standard": "توقيت الخليج"
        }
    }

```

```

    }
  },
  "Guyana": {
    "long": {
      "standard": "توقيت غيانا"
    }
  },
  "Hawaii_Aleutian": {
    "long": {
      "generic": "توقيت هاواي ألوتيان",
      "standard": "توقيت هاواي ألوتيان الرسمي",
      "daylight": "توقيت هاواي ألوتيان الصيفي"
    }
  },
  "Hong_Kong": {
    "long": {
      "generic": "توقيت هونغ كونغ",
      "standard": "توقيت هونغ كونغ الرسمي",
      "daylight": "توقيت هونغ كونغ الصيفي"
    }
  },
  "Hovd": {
    "long": {
      "generic": "توقيت هوفد",
      "standard": "توقيت هوفد الرسمي",
      "daylight": "توقيت هوفد الصيفي"
    }
  },
  "India": {
    "long": {
      "standard": "توقيت الهند"
    }
  },
  "Indian_Ocean": {
    "long": {
      "standard": "توقيت المحيط الهندي"
    }
  },
  "Indochina": {
    "long": {
      "standard": "توقيت الهند الصينية"
    }
  },
  "Indonesia_Central": {
    "long": {
      "standard": "توقيت وسط إندونيسيا"
    }
  },
  "Indonesia_Eastern": {
    "long": {
      "standard": "توقيت شرق إندونيسيا"
    }
  },
  "Indonesia_Western": {
    "long": {
      "standard": "توقيت غرب إندونيسيا"
    }
  }
}

```

```

    },
    "Iran": {
      "long": {
        "generic": "توقيت إيران",
        "standard": "توقيت إيران الرسمي",
        "daylight": "توقيت إيران الصيفي"
      }
    },
    "Irkutsk": {
      "long": {
        "generic": "توقيت إركوتسك",
        "standard": "توقيت إركوتسك الرسمي",
        "daylight": "توقيت إركوتسك الصيفي"
      }
    },
    "Israel": {
      "long": {
        "generic": "توقيت إسرائيل",
        "standard": "توقيت إسرائيل الرسمي",
        "daylight": "توقيت إسرائيل الصيفي"
      }
    },
    "Japan": {
      "long": {
        "generic": "توقيت اليابان",
        "standard": "توقيت اليابان الرسمي",
        "daylight": "توقيت اليابان الصيفي"
      }
    },
    "Kamchatka": {
      "long": {
        "generic": "توقيت كامشاتكا",
        "standard": "توقيت بيتروبافلوفسك-كامتشاتسكي",
        "daylight": "توقيت بيتروبافلوفسك-كامتشاتسكي الصيفي"
      }
    },
    "Kazakhstan_Eastern": {
      "long": {
        "standard": "توقيت شرق كازاخستان"
      }
    },
    "Kazakhstan_Western": {
      "long": {
        "standard": "توقيت غرب كازاخستان"
      }
    },
    "Korea": {
      "long": {
        "generic": "توقيت كوريا",
        "standard": "توقيت كوريا الرسمي",
        "daylight": "توقيت كوريا الصيفي"
      }
    },
    "Kosrae": {
      "long": {
        "standard": "توقيت كوسرا"
      }
    }
  }

```

```

    },
    "Krasnoyarsk": {
      "long": {
        "generic": "توقيت كراسنويارسك",
        "standard": "توقيت كراسنويارسك الرسمي",
        "daylight": "التوقيت الصيفي لكراسنويارسك"
      }
    },
    "Kyrgystan": {
      "long": {
        "standard": "توقيت قيرغيزستان"
      }
    },
    "Line_Islands": {
      "long": {
        "standard": "توقيت جزر لاين"
      }
    },
    "Lord_Howe": {
      "long": {
        "generic": "توقيت لورد هاو",
        "standard": "توقيت لورد هاو الرسمي",
        "daylight": "التوقيت الصيفي للورد هاو"
      }
    },
    "Macquarie": {
      "long": {
        "standard": "توقيت ماكوارى"
      }
    },
    "Magadan": {
      "long": {
        "generic": "توقيت ماغادان",
        "standard": "توقيت ماغادان الرسمي",
        "daylight": "توقيت ماغادان الصيفي"
      }
    },
    "Malaysia": {
      "long": {
        "standard": "توقيت ماليزيا"
      }
    },
    "Maldives": {
      "long": {
        "standard": "توقيت المالديف"
      }
    },
    "Marquesas": {
      "long": {
        "standard": "توقيت ماركيساس"
      }
    },
    "Marshall_Islands": {
      "long": {
        "standard": "توقيت جزر مارشال"
      }
    }
  },

```

```

"Mauritius": {
  "long": {
    "generic": "توقيت موريشيوس",
    "standard": "توقيت موريشيوس الرسمي",
    "daylight": "توقيت موريشيوس الصيفي"
  }
},
"Mawson": {
  "long": {
    "standard": "توقيت ماوسون"
  }
},
"Mexico_Northwest": {
  "long": {
    "generic": "توقيت شمال غرب المكسيك",
    "standard": "التوقيت الرسمي لشمال غرب المكسيك",
    "daylight": "التوقيت الصيفي لشمال غرب المكسيك"
  }
},
"Mexico_Pacific": {
  "long": {
    "generic": "توقيت المحيط الهادي للمكسيك",
    "standard": "توقيت المحيط الهادي الرسمي للمكسيك",
    "daylight": "توقيت المحيط الهادي الصيفي للمكسيك"
  }
},
"Mongolia": {
  "long": {
    "generic": "توقيت أولان باتور",
    "standard": "توقيت أولان باتور الرسمي",
    "daylight": "توقيت أولان باتور الصيفي"
  }
},
"Moscow": {
  "long": {
    "generic": "توقيت موسكو",
    "standard": "توقيت موسكو الرسمي",
    "daylight": "توقيت موسكو الصيفي"
  }
},
"Myanmar": {
  "long": {
    "standard": "توقيت ميانمار"
  }
},
"Nauru": {
  "long": {
    "standard": "توقيت ناورو"
  }
},
"Nepal": {
  "long": {
    "standard": "توقيت نيبال"
  }
},
"New_Caledonia": {
  "long": {

```

```

        "generic": "توقيت كاليدونيا الجديدة",
        "standard": "توقيت كاليدونيا الجديدة الرسمي",
        "daylight": "توقيت كاليدونيا الجديدة الصيفي"
    },
},
"New_Zealand": {
    "long": {
        "generic": "توقيت نيوزيلندا",
        "standard": "توقيت نيوزيلندا الرسمي",
        "daylight": "توقيت نيوزيلندا الصيفي"
    }
},
"Newfoundland": {
    "long": {
        "generic": "توقيت نيوفاوندلاند",
        "standard": "توقيت نيوفاوندلاند الرسمي",
        "daylight": "توقيت نيوفاوندلاند الصيفي"
    }
},
"Niue": {
    "long": {
        "standard": "توقيت نيوي"
    }
},
},
"Norfolk": {
    "long": {
        "standard": "توقيت جزيرة نورفولك"
    }
},
},
"Noronha": {
    "long": {
        "generic": "توقيت فيرناندو دي نورونها",
        "standard": "توقيت فرناندو دي نورونها الرسمي",
        "daylight": "توقيت فرناندو دي نورونها الصيفي"
    }
},
},
"North_Mariana": {
    "long": {
        "standard": "توقيت جزر ماريانا الشمالية"
    }
},
},
"Novosibirsk": {
    "long": {
        "generic": "توقيت نوفوسيبيرسك",
        "standard": "توقيت نوفوسيبيرسك الرسمي",
        "daylight": "توقيت نوفوسيبيرسك الصيفي"
    }
},
},
"Omsk": {
    "long": {
        "generic": "توقيت أومسك",
        "standard": "توقيت أومسك الرسمي",
        "daylight": "توقيت أومسك الصيفي"
    }
},
},
"Pakistan": {
    "long": {

```

```

        "generic": "توقيت باكستان",
        "standard": "توقيت باكستان الرسمي",
        "daylight": "توقيت باكستان الصيفي"
    },
},
"Palau": {
    "long": {
        "standard": "توقيت بالاو"
    }
},
"Papua_New_Guinea": {
    "long": {
        "standard": "توقيت بابوا غينيا الجديدة"
    }
},
"Paraguay": {
    "long": {
        "generic": "توقيت باراغواي",
        "standard": "توقيت باراغواي الرسمي",
        "daylight": "توقيت باراغواي الصيفي"
    }
},
"Peru": {
    "long": {
        "generic": "توقيت بيرو",
        "standard": "توقيت بيرو الرسمي",
        "daylight": "توقيت بيرو الصيفي"
    }
},
"Philippines": {
    "long": {
        "generic": "توقيت الفلبين",
        "standard": "توقيت الفلبين الرسمي",
        "daylight": "توقيت الفلبين الصيفي"
    }
},
"Phoenix_Islands": {
    "long": {
        "standard": "توقيت جزر فينكس"
    }
},
"Pierre_Miquelon": {
    "long": {
        "generic": "توقيت سانت بيير وميكلون",
        "standard": "توقيت سانت بيير وميكلون الرسمي",
        "daylight": "توقيت سانت بيير وميكلون الصيفي"
    }
},
"Pitcairn": {
    "long": {
        "standard": "توقيت بيتكيرن"
    }
},
"Ponape": {
    "long": {
        "standard": "توقيت بونابي"
    }
}

```

```

    },
    "Pyongyang": {
      "long": {
        "standard": "توقيت بيونغ يانغ"
      }
    },
    "Reunion": {
      "long": {
        "standard": "توقيت ريونيون"
      }
    },
    "Rothera": {
      "long": {
        "standard": "توقيت روثيرا"
      }
    },
    "Sakhalin": {
      "long": {
        "generic": "توقيت ساخالين",
        "standard": "توقيت ساخالين الرسمي",
        "daylight": "توقيت ساخالين الصيفي"
      }
    },
    "Samara": {
      "long": {
        "generic": "توقيت سامارا",
        "standard": "توقيت سمارة",
        "daylight": "توقيت سمارة الصيفي"
      }
    },
    "Samoa": {
      "long": {
        "generic": "توقيت ساموا",
        "standard": "توقيت ساموا الرسمي",
        "daylight": "توقيت ساموا الصيفي"
      }
    },
    "Seychelles": {
      "long": {
        "standard": "توقيت سيشل"
      }
    },
    "Singapore": {
      "long": {
        "standard": "توقيت سنغافورة"
      }
    },
    "Solomon": {
      "long": {
        "standard": "توقيت جزر سليمان"
      }
    },
    "South_Georgia": {
      "long": {
        "standard": "توقيت جنوب جورجيا"
      }
    }
  },

```



```
"Suriname": {
  "long": {
    "standard": "توقيت سورينام"
  }
},
"Syowa": {
  "long": {
    "standard": "توقيت سايووا"
  }
},
"Tahiti": {
  "long": {
    "standard": "توقيت تاهيتي"
  }
},
"Taipei": {
  "long": {
    "generic": "توقيت تايبه",
    "standard": "توقيت تايبه الرسمي",
    "daylight": "توقيت تايبه الصيفي"
  }
},
"Tajikistan": {
  "long": {
    "standard": "توقيت طاجكستان"
  }
},
"Tokelau": {
  "long": {
    "standard": "توقيت توكيلاو"
  }
},
"Tonga": {
  "long": {
    "generic": "توقيت تونغا",
    "standard": "توقيت تونغا الرسمي",
    "daylight": "توقيت تونغا الصيفي"
  }
},
"Truk": {
  "long": {
    "standard": "توقيت شوك"
  }
},
"Turkmenistan": {
  "long": {
    "generic": "توقيت تركمانستان",
    "standard": "توقيت تركمانستان الرسمي",
    "daylight": "توقيت تركمانستان الصيفي"
  }
},
"Tuvalu": {
  "long": {
    "standard": "توقيت توفالو"
  }
},
"Uruguay": {
```

```

    "long": {
      "generic": "توقيت أوروغواي",
      "standard": "توقيت أوروغواي الرسمي",
      "daylight": "توقيت أوروغواي الصيفي"
    }
  },
  "Uzbekistan": {
    "long": {
      "generic": "توقيت أوزبكستان",
      "standard": "توقيت أوزبكستان الرسمي",
      "daylight": "توقيت أوزبكستان الصيفي"
    }
  },
  "Vanuatu": {
    "long": {
      "generic": "توقيت فانواتو",
      "standard": "توقيت فانواتو الرسمي",
      "daylight": "توقيت فانواتو الصيفي"
    }
  },
  "Venezuela": {
    "long": {
      "standard": "توقيت فنزويلا"
    }
  },
  "Vladivostok": {
    "long": {
      "generic": "توقيت فلاديفوستوك",
      "standard": "توقيت فلاديفوستوك الرسمي",
      "daylight": "توقيت فلاديفوستوك الصيفي"
    }
  },
  "Volgograd": {
    "long": {
      "generic": "توقيت فولغوغراد",
      "standard": "توقيت فولغوغراد الرسمي",
      "daylight": "توقيت فولغوغراد الصيفي"
    }
  },
  "Vostok": {
    "long": {
      "standard": "توقيت فوستوك"
    }
  },
  "Wake": {
    "long": {
      "standard": "توقيت جزيرة ويك"
    }
  },
  "Wallis": {
    "long": {
      "standard": "توقيت واليس و فوتونا"
    }
  },
  "Yakutsk": {
    "long": {
      "generic": "توقيت ياكوتسك",

```

```

        "standard": "توقيت ياكوتسك الرسمي",
        "daylight": "توقيت ياكوتسك الصيفي"
    },
    "Yekaterinburg": {
        "long": {
            "generic": "توقيت يكاترينبورغ",
            "standard": "توقيت يكاترينبورغ الرسمي",
            "daylight": "توقيت يكاترينبورغ الصيفي"
        }
    }
}

```

TIMEZONENAMES.JSX

```

{
    "main";
    {
        "ar";
        {
            "identity";
            {
                "version";
                {
                    "_number";
                    "$Revision: 13686 $",
                    "_cldrVersion";
                    "32";
                }
                "language";
                "ar";
            }
            "dates";
            {
                "timeZoneNames";
                {
                    "hourFormat";
                    "+HH:mm;-HH:mm",
                    "gmtFormat";
                    "0{غرينتش",
                    "gmtZeroFormat";
                    "غرينتش",
                    "regionFormat";
                    "0{توقيت",
                    "regionFormat-type-daylight";
                    "توقيت 0{الصيفي",
                    "regionFormat-type-standard";
                    "توقيت 0{الرسمي",
                    "fallbackFormat";
                    "{1} ({0})",
                    "zone";
                }
            }
        }
    }
}

```

```
{
  "America";
  {
    "Adak";
    {
      "exemplarCity";
      "أداك";
    }
    "Anchorage";
    {
      "exemplarCity";
      "أنشوراج";
    }
    "Anguilla";
    {
      "exemplarCity";
      "أنغويلا";
    }
    "Antigua";
    {
      "exemplarCity";
      "أنتيغوا";
    }
    "Araguaina";
    {
      "exemplarCity";
      "أروجوانيا";
    }
    "Argentina";
    {
      "Rio_Gallegos";
      {
        "exemplarCity";
        "ريو جالييوس";
      }
      "San_Juan";
      {
        "exemplarCity";
        "سان خوان";
      }
      "Ushuaia";
      {
        "exemplarCity";
        "أشوا";
      }
      "La_Rioja";
      {
        "exemplarCity";
        "لا ريوجا";
      }
      "San_Luis";
      {
        "exemplarCity";
        "سان لويس";
      }
      "Salta";
      {
```

```
        "exemplarCity";
        "سالتا";
    }
    "Tucuman";
    {
        "exemplarCity";
        "تاكمان";
    }
}
"Aruba";
{
    "exemplarCity";
    "أروبا";
}
"Asuncion";
{
    "exemplarCity";
    "أسونسيون";
}
"Bahia";
{
    "exemplarCity";
    "باهيا";
}
"Bahia_Banderas";
{
    "exemplarCity";
    "باهيا بانديراس";
}
"Barbados";
{
    "exemplarCity";
    "بربادوس";
}
"Belem";
{
    "exemplarCity";
    "بلم";
}
"Belize";
{
    "exemplarCity";
    "بليز";
}
"Blanc-Sablon";
{
    "exemplarCity";
    "بلانك-سابلون";
}
"Boa_Vista";
{
    "exemplarCity";
    "باو فيستا";
}
"Bogota";
{
    "exemplarCity";
```

```
        "بوغوتا";
    }
    "Boise";
    {
        "exemplarCity";
        "بويس";
    }
    "Buenos_Aires";
    {
        "exemplarCity";
        "بوينوس آيرس";
    }
    "Cambridge_Bay";
    {
        "exemplarCity";
        "كامبرديج باي";
    }
    "Campo_Grande";
    {
        "exemplarCity";
        "كومبو جراند";
    }
    "Cancun";
    {
        "exemplarCity";
        "كانكون";
    }
    "Caracas";
    {
        "exemplarCity";
        "كاراكاس";
    }
    "Catamarca";
    {
        "exemplarCity";
        "كاتاماركا";
    }
    "Cayenne";
    {
        "exemplarCity";
        "كايين";
    }
    "Cayman";
    {
        "exemplarCity";
        "كايمان";
    }
    "Chicago";
    {
        "exemplarCity";
        "شيكاغو";
    }
    "Chihuahua";
    {
        "exemplarCity";
        "تشياواوا";
    }
}
```

```
"Coral_Harbour";
{
  "exemplarCity";
  "کورال ہاربر";
}
"Cordoba";
{
  "exemplarCity";
  "کوردوبا";
}
"Costa_Rica";
{
  "exemplarCity";
  "کوستاریکا";
}
"Creston";
{
  "exemplarCity";
  "کریستون";
}
"Cuiaba";
{
  "exemplarCity";
  "کیابا";
}
"Curacao";
{
  "exemplarCity";
  "کوراساو";
}
"Danmarkshavn";
{
  "exemplarCity";
  "دانمرک شافن";
}
"Dawson";
{
  "exemplarCity";
  "داوسان";
}
"Dawson_Creek";
{
  "exemplarCity";
  "داوسن کریک";
}
"Denver";
{
  "exemplarCity";
  "دنفر";
}
"Detroit";
{
  "exemplarCity";
  "دیٹرویت";
}
"Dominica";
{
```

```
        "exemplarCity";
        "دومينیکا";
    }
    "Edmonton";
    {
        "exemplarCity";
        "ایدمونتون";
    }
    "Eirunepe";
    {
        "exemplarCity";
        "ایرونپی";
    }
    "El_Salvador";
    {
        "exemplarCity";
        "السلفادور";
    }
    "Fort_Nelson";
    {
        "exemplarCity";
        "فورت نیلسون";
    }
    "Fortaleza";
    {
        "exemplarCity";
        "فورتالیزا";
    }
    "Glace_Bay";
    {
        "exemplarCity";
        "جلاس بای";
    }
    "Godthab";
    {
        "exemplarCity";
        "غودثاب";
    }
    "Goose_Bay";
    {
        "exemplarCity";
        "جوس بای";
    }
    "Grand_Turk";
    {
        "exemplarCity";
        "گرانڈ ترک";
    }
    "Grenada";
    {
        "exemplarCity";
        "گرینادا";
    }
    "Guadeloupe";
    {
        "exemplarCity";
        "گواڈلوب";
    }
```



```
}
"Guatemala";
{
  "exemplarCity";
  "غواتيمالا";
}
"Guayaquil";
{
  "exemplarCity";
  "غواياكويل";
}
"Guyana";
{
  "exemplarCity";
  "غيانا";
}
"Halifax";
{
  "exemplarCity";
  "هاليفاكس";
}
"Havana";
{
  "exemplarCity";
  "هافانا";
}
"Hermosillo";
{
  "exemplarCity";
  "هيرموسيلو";
}
"Indiana";
{
  "Vincennes";
  {
    "exemplarCity";
    "فينسينس";
  }
  "Petersburg";
  {
    "exemplarCity";
    "بيتربيرغ";
  }
  "Tell_City";
  {
    "exemplarCity";
    "مدينة تل، إنديانا";
  }
  "Knox";
  {
    "exemplarCity";
    "كونكس";
  }
  "Winamac";
  {
    "exemplarCity";
    "ويناماك";
  }
}
```

```
}
  "Marengo";
  {
    "exemplarCity";
    "مارنجو";
  }
  "Vevay";
  {
    "exemplarCity";
    "فيفاي";
  }
}
"Indianapolis";
{
  "exemplarCity";
  "إنديانا بوليس";
}
"Inuvik";
{
  "exemplarCity";
  "اينوفيك";
}
"Iqaluit";
{
  "exemplarCity";
  "اكويلت";
}
"Jamaica";
{
  "exemplarCity";
  "جاما يكا";
}
"Jujuy";
{
  "exemplarCity";
  "جوجو";
}
"Juneau";
{
  "exemplarCity";
  "جونى";
}
"Kentucky";
{
  "Monticello";
  {
    "exemplarCity";
    "مونتي سيلو";
  }
}
"Kralendijk";
{
  "exemplarCity";
  "كرالنديك";
}
"La_Paz";
{
```

```
        "exemplarCity";
        "لا باز";
    }
    "Lima";
    {
        "exemplarCity";
        "ليما";
    }
    "Los_Angeles";
    {
        "exemplarCity";
        "لوس انجلوس";
    }
    "Louisville";
    {
        "exemplarCity";
        "لويس فيل";
    }
    "Lower_Princes";
    {
        "exemplarCity";
        "حي الأمير السفلي";
    }
    "Maceio";
    {
        "exemplarCity";
        "ماشيو";
    }
    "Managua";
    {
        "exemplarCity";
        "ماناغوا";
    }
    "Manaus";
    {
        "exemplarCity";
        "ماناوس";
    }
    "Marigot";
    {
        "exemplarCity";
        "ماريغوت";
    }
    "Martinique";
    {
        "exemplarCity";
        "المارتينيك";
    }
    "Matamoros";
    {
        "exemplarCity";
        "ماتاموروس";
    }
    "Mazatlan";
    {
        "exemplarCity";
        "مازاتلان";
    }
```

```
}
"Mendoza";
{
  "exemplarCity";
  "ميندوزا";
}
"Menominee";
{
  "exemplarCity";
  "مينوميني";
}
"Merida";
{
  "exemplarCity";
  "ميريدا";
}
"Metlakatla";
{
  "exemplarCity";
  "ميتلاكاتلا";
}
"Mexico_City";
{
  "exemplarCity";
  "مدينة المكسيك";
}
"Miquelon";
{
  "exemplarCity";
  "مكويلون";
}
"Moncton";
{
  "exemplarCity";
  "وينكتون";
}
"Monterrey";
{
  "exemplarCity";
  "مونتيري";
}
"Montevideo";
{
  "exemplarCity";
  "مونتيفيديو";
}
"Montserrat";
{
  "exemplarCity";
  "مونتسيرات";
}
"Nassau";
{
  "exemplarCity";
  "ناسو";
}
"New_York";
```

```

{
    "exemplarCity";
    "نيويورك";
}
"Nipigon";
{
    "exemplarCity";
    "نيبيجون";
}
"Nome";
{
    "exemplarCity";
    "نوم";
}
"Noronha";
{
    "exemplarCity";
    "نوروناه";
}
"North_Dakota";
{
    "Beulah";
    {
        "exemplarCity";
        "بيولا، داكوتا الشمالية";
    }
    "New_Salem";
    {
        "exemplarCity";
        "نيو ساليم";
    }
    "Center";
    {
        "exemplarCity";
        "سنتر";
    }
}
"Ojinaga";
{
    "exemplarCity";
    "أوجيناغا";
}
"Panama";
{
    "exemplarCity";
    "بنما";
}
"Pangnirtung";
{
    "exemplarCity";
    "بانجينتینگ";
}
"Paramaribo";
{
    "exemplarCity";
    "باراماريبو";
}

```

```
"Phoenix";
{
  "exemplarCity";
  "فينكس";
}
"Port-au-Prince";
{
  "exemplarCity";
  "بورت أو برنس";
}
"Port_of_Spain";
{
  "exemplarCity";
  "بورت أوف سبين";
}
"Porto_Velho";
{
  "exemplarCity";
  "بورتو فيلو";
}
"Puerto_Rico";
{
  "exemplarCity";
  "بورتوريكو";
}
"Punta_Arenas";
{
  "exemplarCity";
  "بونتأ أريناز";
}
"Rainy_River";
{
  "exemplarCity";
  "راني ريفر";
}
"Rankin_Inlet";
{
  "exemplarCity";
  "رانكن انلت";
}
"Recife";
{
  "exemplarCity";
  "ريسيف";
}
"Regina";
{
  "exemplarCity";
  "ريجينا";
}
"Resolute";
{
  "exemplarCity";
  "ريزولوت";
}
"Rio_Branco";
{
```

```

        "exemplarCity";
        "ريوبرانكو";
    }
    "Santa_Isabel";
    {
        "exemplarCity";
        "سانتا إيزابيل";
    }
    "Santarem";
    {
        "exemplarCity";
        "سانتاريم";
    }
    "Santiago";
    {
        "exemplarCity";
        "سانتياغو";
    }
    "Santo_Domingo";
    {
        "exemplarCity";
        "سانتو دومينغو";
    }
    "Sao_Paulo";
    {
        "exemplarCity";
        "ساو باولو";
    }
    "Scoresbysund";
    {
        "exemplarCity";
        "سكورسبيسند";
    }
    "Sitka";
    {
        "exemplarCity";
        "سيتكا";
    }
    "St_Barthelemy";
    {
        "exemplarCity";
        "سانت بارتيليمي";
    }
    "St_Johns";
    {
        "exemplarCity";
        "سانت جونز";
    }
    "St_Kitts";
    {
        "exemplarCity";
        "سانت كيتس";
    }
    "St_Lucia";
    {
        "exemplarCity";
        "سانت لوشيا";
    }

```

```
}
"St_Thomas";
{
  "exemplarCity";
  "سانت توماس";
}
"St_Vincent";
{
  "exemplarCity";
  "سانت فنسنت";
}
"Swift_Current";
{
  "exemplarCity";
  "سوفت كارنت";
}
"Tegucigalpa";
{
  "exemplarCity";
  "تيغوسيغالبا";
}
"Thule";
{
  "exemplarCity";
  "ثيل";
}
"Thunder_Bay";
{
  "exemplarCity";
  "ثندر باي";
}
"Tijuana";
{
  "exemplarCity";
  "تيخوانا";
}
"Toronto";
{
  "exemplarCity";
  "تورونتو";
}
"Tortola";
{
  "exemplarCity";
  "تورتولا";
}
"Vancouver";
{
  "exemplarCity";
  "فانكوفر";
}
"Whitehorse";
{
  "exemplarCity";
  "وايت هورس";
}
"Winnipeg";
```



```
{
  "exemplarCity";
  "وينيبيج";
}
"Yakutat";
{
  "exemplarCity";
  "ياكوتات";
}
"Yellowknife";
{
  "exemplarCity";
  "يلونيف";
}
}
"Atlantic";
{
  "Azores";
  {
    "exemplarCity";
    "أزورس";
  }
  "Bermuda";
  {
    "exemplarCity";
    "برمودا";
  }
  "Canary";
  {
    "exemplarCity";
    "كناري";
  }
  "Cape_Verde";
  {
    "exemplarCity";
    "الرأس الأخضر";
  }
  "Faeroe";
  {
    "exemplarCity";
    "فارو";
  }
  "Madeira";
  {
    "exemplarCity";
    "ماديرا";
  }
  "Reykjavik";
  {
    "exemplarCity";
    "ريكيافيك";
  }
  "South_Georgia";
  {
    "exemplarCity";
    "جورجيا الجنوبية";
  }
}
```

```
        "St_Helena";
        {
            "exemplarCity";
            "سانت هيلينا";
        }
        "Stanley";
        {
            "exemplarCity";
            "استانلي";
        }
    }
    "Europe";
    {
        "Amsterdam";
        {
            "exemplarCity";
            "أمستردام";
        }
        "Andorra";
        {
            "exemplarCity";
            "أندورا";
        }
        "Astrakhan";
        {
            "exemplarCity";
            "أستراخان";
        }
        "Athens";
        {
            "exemplarCity";
            "أثينا";
        }
        "Belgrade";
        {
            "exemplarCity";
            "بلغراد";
        }
        "Berlin";
        {
            "exemplarCity";
            "برلين";
        }
        "Bratislava";
        {
            "exemplarCity";
            "براتيسلافا";
        }
        "Brussels";
        {
            "exemplarCity";
            "بروكسل";
        }
        "Bucharest";
        {
            "exemplarCity";
            "بوخارست";
        }
    }
```

```
}
"Budapest";
{
  "exemplarCity";
  "بودابست";
}
"Busingen";
{
  "exemplarCity";
  "بوسنغن";
}
"Chisinau";
{
  "exemplarCity";
  "تشيسيناو";
}
"Copenhagen";
{
  "exemplarCity";
  "كوبنهاغن";
}
"Dublin";
{
  "long";
  {
    "daylight";
    "توقيت أيرلندا الرسمي";
  }
  "exemplarCity";
  "دبلن";
}
"Gibraltar";
{
  "exemplarCity";
  "جبل طارق";
}
"Guernsey";
{
  "exemplarCity";
  "غيرنسي";
}
"Helsinki";
{
  "exemplarCity";
  "هلسنكي";
}
"Isle_of_Man";
{
  "exemplarCity";
  "جزيرة مان";
}
"Istanbul";
{
  "exemplarCity";
  "إسطنبول";
}
"Jersey";
```

```
{
  "exemplarCity";
  "جیرسی";
}
"Kaliningrad";
{
  "exemplarCity";
  "کالینجراد";
}
"Kiev";
{
  "exemplarCity";
  "کیف";
}
"Kirov";
{
  "exemplarCity";
  "کیروف";
}
"Lisbon";
{
  "exemplarCity";
  "لشبونة";
}
"Ljubljana";
{
  "exemplarCity";
  "لیوبلیانا";
}
"London";
{
  "long";
  {
    "daylight";
    "توقيت بريطانيا الصيفي";
  }
  "exemplarCity";
  "لندن";
}
"Luxembourg";
{
  "exemplarCity";
  "لوکسمبورغ";
}
"Madrid";
{
  "exemplarCity";
  "مدريد";
}
"Malta";
{
  "exemplarCity";
  "مالطة";
}
"Mariehamn";
{
  "exemplarCity";
```

```

        "ماريهامن";
    }
    "Minsk";
    {
        "exemplarCity";
        "مينسك";
    }
    "Monaco";
    {
        "exemplarCity";
        "موناكو";
    }
    "Moscow";
    {
        "exemplarCity";
        "موسكو";
    }
    "Oslo";
    {
        "exemplarCity";
        "أوسلو";
    }
    "Paris";
    {
        "exemplarCity";
        "باريس";
    }
    "Podgorica";
    {
        "exemplarCity";
        "بودغوريكا";
    }
    "Prague";
    {
        "exemplarCity";
        "براغ";
    }
    "Riga";
    {
        "exemplarCity";
        "ريغا";
    }
    "Rome";
    {
        "exemplarCity";
        "روما";
    }
    "Samara";
    {
        "exemplarCity";
        "سمراء";
    }
    "San_Marino";
    {
        "exemplarCity";
        "سان مارينو";
    }
}

```

```
"Sarajevo";
{
  "exemplarCity";
  "سراييفو";
}
"Saratov";
{
  "exemplarCity";
  "سار اتوف";
}
"Simferopol";
{
  "exemplarCity";
  "سيمفروبول";
}
"Skopje";
{
  "exemplarCity";
  "سكوبي";
}
"Sofia";
{
  "exemplarCity";
  "صوفيا";
}
"Stockholm";
{
  "exemplarCity";
  "ستوكهولم";
}
"Tallinn";
{
  "exemplarCity";
  "تالين";
}
"Tirane";
{
  "exemplarCity";
  "تيرانا";
}
"Ulyanovsk";
{
  "exemplarCity";
  "أوليانوفسك";
}
"Uzhgorod";
{
  "exemplarCity";
  "أوزجروود";
}
"Vaduz";
{
  "exemplarCity";
  "فادوز";
}
"Vatican";
{
```

```

        "exemplarCity";
        "الفاتيكان";
    }
    "Vienna";
    {
        "exemplarCity";
        "فيينا";
    }
    "Vilnius";
    {
        "exemplarCity";
        "فيلنيوس";
    }
    "Volgograd";
    {
        "exemplarCity";
        "فولجograd";
    }
    "Warsaw";
    {
        "exemplarCity";
        "وارسو";
    }
    "Zagreb";
    {
        "exemplarCity";
        "زغرب";
    }
    "Zaporozhye";
    {
        "exemplarCity";
        "زابوروزي";
    }
    "Zurich";
    {
        "exemplarCity";
        "زيورخ";
    }
}
"Africa";
{
    "Abidjan";
    {
        "exemplarCity";
        "أبيدجان";
    }
    "Accra";
    {
        "exemplarCity";
        "أكرا";
    }
    "Addis_Ababa";
    {
        "exemplarCity";
        "أديس أبابا";
    }
    "Algiers";

```

```
{
  "exemplarCity";
  "الجزائر";
}
"Asmera";
{
  "exemplarCity";
  "أسمرّة";
}
"Bamako";
{
  "exemplarCity";
  "باماكو";
}
"Bangui";
{
  "exemplarCity";
  "بانغوي";
}
"Banjul";
{
  "exemplarCity";
  "بانجول";
}
"Bissau";
{
  "exemplarCity";
  "بيساو";
}
"Blantyre";
{
  "exemplarCity";
  "بلانتيير";
}
"Brazzaville";
{
  "exemplarCity";
  "برازافيل";
}
"Bujumbura";
{
  "exemplarCity";
  "بوجومبورا";
}
"Cairo";
{
  "exemplarCity";
  "القاهرة";
}
"Casablanca";
{
  "exemplarCity";
  "الدار البيضاء";
}
"Ceuta";
{
  "exemplarCity";
```



```
        "سيتا";
    }
    "Conakry";
    {
        "exemplarCity";
        "كوناكري";
    }
    "Dakar";
    {
        "exemplarCity";
        "داكار";
    }
    "Dar_es_Salaam";
    {
        "exemplarCity";
        "دار السلام";
    }
    "Djibouti";
    {
        "exemplarCity";
        "جيبوتي";
    }
    "Douala";
    {
        "exemplarCity";
        "دوالا";
    }
    "El_Aaiun";
    {
        "exemplarCity";
        "العيون";
    }
    "Freetown";
    {
        "exemplarCity";
        "فري تاون";
    }
    "Gaborone";
    {
        "exemplarCity";
        "غابورون";
    }
    "Harare";
    {
        "exemplarCity";
        "هراري";
    }
    "Johannesburg";
    {
        "exemplarCity";
        "جوهانسبرغ";
    }
    "Juba";
    {
        "exemplarCity";
        "جوبا";
    }
}
```

```
"Kampala";
{
  "exemplarCity";
  "كامبالا";
}
"Khartoum";
{
  "exemplarCity";
  "الخرطوم";
}
"Kigali";
{
  "exemplarCity";
  "كيغالي";
}
"Kinshasa";
{
  "exemplarCity";
  "كينشاسا";
}
"Lagos";
{
  "exemplarCity";
  "لاغوس";
}
"Libreville";
{
  "exemplarCity";
  "ليبرفيل";
}
"Lome";
{
  "exemplarCity";
  "لومي";
}
"Luanda";
{
  "exemplarCity";
  "لواندا";
}
"Lubumbashi";
{
  "exemplarCity";
  "لومبباشا";
}
"Lusaka";
{
  "exemplarCity";
  "لوساكا";
}
"Malabo";
{
  "exemplarCity";
  "مالابو";
}
"Maputo";
{
```

```
        "exemplarCity";
        "مابوتو";
    }
    "Maseru";
    {
        "exemplarCity";
        "ماسيرو";
    }
    "Mbabane";
    {
        "exemplarCity";
        "مباباني";
    }
    "Mogadishu";
    {
        "exemplarCity";
        "مقديشو";
    }
    "Monrovia";
    {
        "exemplarCity";
        "مونروفيا";
    }
    "Nairobi";
    {
        "exemplarCity";
        "نيروبي";
    }
    "Ndjamena";
    {
        "exemplarCity";
        "نجامينا";
    }
    "Niamey";
    {
        "exemplarCity";
        "نيامي";
    }
    "Nouakchott";
    {
        "exemplarCity";
        "نواكشوط";
    }
    "Ouagadougou";
    {
        "exemplarCity";
        "واغادوغو";
    }
    "Porto-Novo";
    {
        "exemplarCity";
        "بورتو نوفو";
    }
    "Sao_Tome";
    {
        "exemplarCity";
        "ساو تومي";
    }
```

```
}
  "Tripoli";
  {
    "exemplarCity";
    "طرابلس";
  }
  "Tunis";
  {
    "exemplarCity";
    "تونس";
  }
  "Windhoek";
  {
    "exemplarCity";
    "ويندهوك";
  }
}
"Asia";
{
  "Aden";
  {
    "exemplarCity";
    "عدن";
  }
  "Almaty";
  {
    "exemplarCity";
    "ألماتي";
  }
  "Amman";
  {
    "exemplarCity";
    "عمان";
  }
  "Anadyr";
  {
    "exemplarCity";
    "أندير";
  }
  "Aqtau";
  {
    "exemplarCity";
    "أكتاو";
  }
  "Aqtobe";
  {
    "exemplarCity";
    "أكتوب";
  }
  "Ashgabat";
  {
    "exemplarCity";
    "عشق آباد";
  }
  "Atyrau";
  {
    "exemplarCity";
```

```
        "أتيراو";
    }
    "Baghdad";
    {
        "exemplarCity";
        "بغداد";
    }
    "Bahrain";
    {
        "exemplarCity";
        "البحرين";
    }
    "Baku";
    {
        "exemplarCity";
        "باكو";
    }
    "Bangkok";
    {
        "exemplarCity";
        "بانكوك";
    }
    "Barnaul";
    {
        "exemplarCity";
        "بارناول";
    }
    "Beirut";
    {
        "exemplarCity";
        "بيروت";
    }
    "Bishkek";
    {
        "exemplarCity";
        "بشكيك";
    }
    "Brunei";
    {
        "exemplarCity";
        "بروناي";
    }
    "Calcutta";
    {
        "exemplarCity";
        "كالكتا";
    }
    "Chita";
    {
        "exemplarCity";
        "تشيتا";
    }
    "Choibalsan";
    {
        "exemplarCity";
        "تشوبالسان";
    }
}
```

```
"Colombo";
{
  "exemplarCity";
  "كولومبو";
}
"Damascus";
{
  "exemplarCity";
  "دمشق";
}
"Dhaka";
{
  "exemplarCity";
  "دكا";
}
"Dili";
{
  "exemplarCity";
  "ديلي";
}
"Dubai";
{
  "exemplarCity";
  "دبي";
}
"Dushanbe";
{
  "exemplarCity";
  "دوشانبي";
}
"Famagusta";
{
  "exemplarCity";
  "فاما غوستا";
}
"Gaza";
{
  "exemplarCity";
  "غزة";
}
"Hebron";
{
  "exemplarCity";
  "هيبرون (مدينة الخليل)";
}
"Hong_Kong";
{
  "exemplarCity";
  "هونغ كونغ";
}
"Hovd";
{
  "exemplarCity";
  "هوفد";
}
"Irkutsk";
{
```

```

        "exemplarCity";
        "ایرکیتسک";
    }
    "Jakarta";
    {
        "exemplarCity";
        "جاکرتا";
    }
    "Jayapura";
    {
        "exemplarCity";
        "جایابورا";
    }
    "Jerusalem";
    {
        "exemplarCity";
        "القدس";
    }
    "Kabul";
    {
        "exemplarCity";
        "کابل";
    }
    "Kamchatka";
    {
        "exemplarCity";
        "کامشاتکا";
    }
    "Karachi";
    {
        "exemplarCity";
        "کراتچی";
    }
    "Katmandu";
    {
        "exemplarCity";
        "کاتماندو";
    }
    "Khandyga";
    {
        "exemplarCity";
        "خاندیجا";
    }
    "Krasnoyarsk";
    {
        "exemplarCity";
        "کراسنویارسک";
    }
    "Kuala_Lumpur";
    {
        "exemplarCity";
        "کوالا لامبور";
    }
    "Kuching";
    {
        "exemplarCity";
        "کیشینگ";
    }

```

```
}
"Kuwait";
{
  "exemplarCity";
  "الكويت";
}
"Macau";
{
  "exemplarCity";
  "مكاو";
}
"Magadan";
{
  "exemplarCity";
  "مجادن";
}
"Makassar";
{
  "exemplarCity";
  "ماكassar";
}
"Manila";
{
  "exemplarCity";
  "مانيل";
}
"Muscat";
{
  "exemplarCity";
  "مسقط";
}
"Nicosia";
{
  "exemplarCity";
  "نيقوسيا";
}
"Novokuznetsk";
{
  "exemplarCity";
  "نوفوكوزنتسك";
}
"Novosibirsk";
{
  "exemplarCity";
  "نوفوسبيرسك";
}
"Omsk";
{
  "exemplarCity";
  "أومسك";
}
"Oral";
{
  "exemplarCity";
  "أورال";
}
"Phnom_Penh";
```



```
{
  "exemplarCity";
  "بنوم بنه";
}
"Pontianak";
{
  "exemplarCity";
  "بونتیانک";
}
"Pyongyang";
{
  "exemplarCity";
  "بیونغ یانغ";
}
"Qatar";
{
  "exemplarCity";
  "قطر";
}
"Qyzylorda";
{
  "exemplarCity";
  "کیزیلوردا";
}
"Rangoon";
{
  "exemplarCity";
  "رانغون";
}
"Riyadh";
{
  "exemplarCity";
  "الریاض";
}
"Saigon";
{
  "exemplarCity";
  "مدینة هو تشی منه";
}
"Sakhalin";
{
  "exemplarCity";
  "سکالین";
}
"Samarkand";
{
  "exemplarCity";
  "سمرقند";
}
"Seoul";
{
  "exemplarCity";
  "سول";
}
"Shanghai";
{
  "exemplarCity";
```

```
        "شنغهاي";
    }
    "Singapore";
    {
        "exemplarCity";
        "سنغافورة";
    }
    "Srednekolymsk";
    {
        "exemplarCity";
        "سريدنكوليمسك";
    }
    "Taipei";
    {
        "exemplarCity";
        "تايبيه";
    }
    "Tashkent";
    {
        "exemplarCity";
        "تشقند";
    }
    "Tbilisi";
    {
        "exemplarCity";
        "تبليسي";
    }
    "Tehran";
    {
        "exemplarCity";
        "طهران";
    }
    "Thimphu";
    {
        "exemplarCity";
        "تيمفو";
    }
    "Tokyo";
    {
        "exemplarCity";
        "طوكيو";
    }
    "Tomsk";
    {
        "exemplarCity";
        "تومسك";
    }
    "Ulaanbaatar";
    {
        "exemplarCity";
        "آلانباتار";
    }
    "Urumqi";
    {
        "exemplarCity";
        "أرومكي";
    }
}
```

```
"Ust-Nera";
{
  "exemplarCity";
  "أوست نيرا";
}
"Vientiane";
{
  "exemplarCity";
  "فيانتيان";
}
"Vladivostok";
{
  "exemplarCity";
  "فلاديفوستك";
}
"Yakutsk";
{
  "exemplarCity";
  "ياكوتسك";
}
"Yekaterinburg";
{
  "exemplarCity";
  "يكا ترنبيرج";
}
"Yerevan";
{
  "exemplarCity";
  "يريفان";
}
}
"Indian";
{
  "Antananarivo";
  {
    "exemplarCity";
    "أنتاناناريفو";
  }
  "Chagos";
  {
    "exemplarCity";
    "تشاغوس";
  }
  "Christmas";
  {
    "exemplarCity";
    "كريسماس";
  }
  "Cocos";
  {
    "exemplarCity";
    "كوكوس";
  }
  "Comoro";
  {
    "exemplarCity";
    "جزر القمر";
  }
}
```

```
}
"Kerguelen";
{
  "exemplarCity";
  "كيرغويلين";
}
"Mahe";
{
  "exemplarCity";
  "ماهي";
}
"Maldives";
{
  "exemplarCity";
  "المالديف";
}
"Mauritius";
{
  "exemplarCity";
  "موريشيوس";
}
"Mayotte";
{
  "exemplarCity";
  "مايوت";
}
"Reunion";
{
  "exemplarCity";
  "ريونيون";
}
}
"Australia";
{
  "Adelaide";
  {
    "exemplarCity";
    "أديليد";
  }
  "Brisbane";
  {
    "exemplarCity";
    "برسبان";
  }
  "Broken_Hill";
  {
    "exemplarCity";
    "بروكن هيل";
  }
  "Currie";
  {
    "exemplarCity";
    "كوري";
  }
  "Darwin";
  {
    "exemplarCity";
```

```

        "دارون";
    }
    "Eucla";
    {
        "exemplarCity";
        "أوكلا";
    }
    "Hobart";
    {
        "exemplarCity";
        "هوبارت";
    }
    "Lindeman";
    {
        "exemplarCity";
        "ليندمان";
    }
    "Lord_Howe";
    {
        "exemplarCity";
        "لورد هاو";
    }
    "Melbourne";
    {
        "exemplarCity";
        "ميلبورن";
    }
    "Perth";
    {
        "exemplarCity";
        "برثا";
    }
    "Sydney";
    {
        "exemplarCity";
        "سيدني";
    }
}
"Pacific";
{
    "Apia";
    {
        "exemplarCity";
        "أبيا";
    }
    "Auckland";
    {
        "exemplarCity";
        "أوكلاند";
    }
    "Bougainville";
    {
        "exemplarCity";
        "بوغانفيل";
    }
    "Chatham";
    {

```

```
        "exemplarCity";
        "تشاٹام";
    }
    "Easter";
    {
        "exemplarCity";
        "استر";
    }
    "Efate";
    {
        "exemplarCity";
        "إيفات";
    }
    "Enderbury";
    {
        "exemplarCity";
        "اندربيرج";
    }
    "Fakaofu";
    {
        "exemplarCity";
        "فاكاوفو";
    }
    "Fiji";
    {
        "exemplarCity";
        "فيجي";
    }
    "Funafuti";
    {
        "exemplarCity";
        "فونافوتي";
    }
    "Galapagos";
    {
        "exemplarCity";
        "جلاباجوس";
    }
    "Gambier";
    {
        "exemplarCity";
        "جامبير";
    }
    "Guadalcanal";
    {
        "exemplarCity";
        "غواد الكانال";
    }
    "Guam";
    {
        "exemplarCity";
        "غوام";
    }
    "Honolulu";
    {
        "exemplarCity";
        "هونولولو";
    }
```

```
}
"Johnston";
{
  "exemplarCity";
  "جونستون";
}
"Kiritimati";
{
  "exemplarCity";
  "كيريتي ماتي";
}
"Kosrae";
{
  "exemplarCity";
  "كوسرا";
}
"Kwajalein";
{
  "exemplarCity";
  "كواجالين";
}
"Majuro";
{
  "exemplarCity";
  "ماجورو";
}
"Marquesas";
{
  "exemplarCity";
  "ماركيساس";
}
"Midway";
{
  "exemplarCity";
  "ميدواي";
}
"Nauru";
{
  "exemplarCity";
  "ناورو";
}
"Niue";
{
  "exemplarCity";
  "نيوي";
}
"Norfolk";
{
  "exemplarCity";
  "نورفولك";
}
"Noumea";
{
  "exemplarCity";
  "نوميا";
}
"Pago_Pago";
```

```
{
    "exemplarCity";
    "باغو باغو";
}
"Palau";
{
    "exemplarCity";
    "بالاو";
}
"Pitcairn";
{
    "exemplarCity";
    "بیتکیرن";
}
"Ponape";
{
    "exemplarCity";
    "باناب";
}
"Port_Moresby";
{
    "exemplarCity";
    "پور مورسبی";
}
"Rarotonga";
{
    "exemplarCity";
    "راروتونگا";
}
"Saipan";
{
    "exemplarCity";
    "سایبان";
}
"Tahiti";
{
    "exemplarCity";
    "تاهیتی";
}
"Tarawa";
{
    "exemplarCity";
    "تاراوا";
}
"Tongatapu";
{
    "exemplarCity";
    "تونغاتابو";
}
"Truk";
{
    "exemplarCity";
    "ترك";
}
"Wake";
{
    "exemplarCity";
```



```

        "واك";
    }
    "Wallis";
    {
        "exemplarCity";
        "واليس";
    }
}
"Arctic";
{
    "Longyearbyen";
    {
        "exemplarCity";
        "لونجيربين";
    }
}
"Antarctica";
{
    "Casey";
    {
        "exemplarCity";
        "كاساي";
    }
    "Davis";
    {
        "exemplarCity";
        "دافيز";
    }
    "DumontDUrville";
    {
        "exemplarCity";
        "دي مونت دو روفيل";
    }
    "Macquarie";
    {
        "exemplarCity";
        "ماكواراي";
    }
    "Mawson";
    {
        "exemplarCity";
        "ماوسون";
    }
    "McMurdo";
    {
        "exemplarCity";
        "ماك مور دو";
    }
    "Palmer";
    {
        "exemplarCity";
        "بالمير";
    }
    "Rothera";
    {
        "exemplarCity";
        "روثيرا";
    }
}

```

```

    }
    "Syowa";
    {
      "exemplarCity";
      "سايووا";
    }
    "Troll";
    {
      "exemplarCity";
      "ترول";
    }
    "Vostok";
    {
      "exemplarCity";
      "فوستوك";
    }
  }
  "Etc";
  {
    "UTC";
    {
      "long";
      {
        "standard";
        "التوقيت العالمي المنسق";
      }
      "short";
      {
        "standard";
        "UTC";
      }
    }
    "Unknown";
    {
      "exemplarCity";
      "مدينة غير معروفة";
    }
  }
}
"metazone";
{
  "Afghanistan";
  {
    "long";
    {
      "standard";
      "توقيت أفغانستان";
    }
  }
  "Africa_Central";
  {
    "long";
    {
      "standard";
      "توقيت وسط أفريقيا";
    }
  }
}

```

```
"Africa_Eastern";
{
  "long";
  {
    "standard";
    "توقيت شرق أفريقيا";
  }
}
"Africa_Southern";
{
  "long";
  {
    "standard";
    "توقيت جنوب أفريقيا";
  }
}
"Africa_Western";
{
  "long";
  {
    "generic";
    "توقيت غرب أفريقيا",
    "standard";
    "توقيت غرب أفريقيا الرسمي",
    "daylight";
    "توقيت غرب أفريقيا الصيفي";
  }
}
"Alaska";
{
  "long";
  {
    "generic";
    "توقيت ألاسكا",
    "standard";
    "التوقيت الرسمي لألاسكا",
    "daylight";
    "توقيت ألاسكا الصيفي";
  }
}
"Amazon";
{
  "long";
  {
    "generic";
    "توقيت الأمازون",
    "standard";
    "توقيت الأمازون الرسمي",
    "daylight";
    "توقيت الأمازون الصيفي";
  }
}
"America_Central";
{
  "long";
  {
    "generic";
```

```

        "التوقيت المركزي لأمريكا الشمالية",
        "standard";
        "التوقيت الرسمي المركزي لأمريكا الشمالية",
        "daylight";
        "التوقيت الصيفي المركزي لأمريكا الشمالية";
    }
}
"America_Eastern";
{
    "long";
    {
        "generic";
        "التوقيت الشرقي لأمريكا الشمالية",
        "standard";
        "التوقيت الرسمي الشرقي لأمريكا الشمالية",
        "daylight";
        "التوقيت الصيفي الشرقي لأمريكا الشمالية";
    }
}
"America_Mountain";
{
    "long";
    {
        "generic";
        "التوقيت الجبلي لأمريكا الشمالية",
        "standard";
        "التوقيت الجبلي الرسمي لأمريكا الشمالية",
        "daylight";
        "التوقيت الجبلي الصيفي لأمريكا الشمالية";
    }
}
"America_Pacific";
{
    "long";
    {
        "generic";
        "توقيت المحيط الهادي",
        "standard";
        "توقيت المحيط الهادي الرسمي",
        "daylight";
        "توقيت المحيط الهادي الصيفي";
    }
}
"Anadyr";
{
    "long";
    {
        "generic";
        "توقيت أنادير",
        "standard";
        "توقيت أنادير الرسمي",
        "daylight";
        "التوقيت الصيفي لأنادير";
    }
}
"Apia";
{

```

```

        "long";
        {
            "generic";
            "توقيت آبيا",
            "standard";
            "التوقيت الرسمي لآبيا",
            "daylight";
            "التوقيت الصيفي لآبيا";
        }
    }
    "Arabic";
    {
        "long";
        {
            "generic";
            "التوقيت العربي",
            "standard";
            "التوقيت العربي الرسمي",
            "daylight";
            "التوقيت العربي الصيفي";
        }
    }
    "Argentina";
    {
        "long";
        {
            "generic";
            "توقيت الأرجنتين",
            "standard";
            "توقيت الأرجنتين الرسمي",
            "daylight";
            "توقيت الأرجنتين الصيفي";
        }
    }
    "Argentina_Western";
    {
        "long";
        {
            "generic";
            "توقيت غرب الأرجنتين",
            "standard";
            "توقيت غرب الأرجنتين الرسمي",
            "daylight";
            "توقيت غرب الأرجنتين الصيفي";
        }
    }
    "Armenia";
    {
        "long";
        {
            "generic";
            "توقيت أرمينيا",
            "standard";
            "توقيت أرمينيا الرسمي",
            "daylight";
            "توقيت أرمينيا الصيفي";
        }
    }

```

```

    }
    "Atlantic";
    {
        "long";
        {
            "generic";
            "توقيت الأطلسي",
            "standard";
            "التوقيت الرسمي الأطلسي",
            "daylight";
            "التوقيت الصيفي الأطلسي";
        }
    }
    "Australia_Central";
    {
        "long";
        {
            "generic";
            "توقيت وسط أستراليا",
            "standard";
            "توقيت وسط أستراليا الرسمي",
            "daylight";
            "توقيت وسط أستراليا الصيفي";
        }
    }
    "Australia_CentralWestern";
    {
        "long";
        {
            "generic";
            "توقيت غرب وسط أستراليا",
            "standard";
            "توقيت غرب وسط أستراليا الرسمي",
            "daylight";
            "توقيت غرب وسط أستراليا الصيفي";
        }
    }
    "Australia_Eastern";
    {
        "long";
        {
            "generic";
            "توقيت شرق أستراليا",
            "standard";
            "توقيت شرق أستراليا الرسمي",
            "daylight";
            "توقيت شرق أستراليا الصيفي";
        }
    }
    "Australia_Western";
    {
        "long";
        {
            "generic";
            "توقيت غرب أستراليا",
            "standard";
            "توقيت غرب أستراليا الرسمي",

```

```

        "daylight";
        "توقيت غرب أستراليا الصيفي";
    }
}
"Azerbaijan";
{
    "long";
    {
        "generic";
        "توقيت أذربيجان",
        "standard";
        "توقيت أذربيجان الرسمي",
        "daylight";
        "توقيت أذربيجان الصيفي";
    }
}
"Azores";
{
    "long";
    {
        "generic";
        "توقيت أزورس",
        "standard";
        "توقيت أزورس الرسمي",
        "daylight";
        "توقيت أزورس الصيفي";
    }
}
"Bangladesh";
{
    "long";
    {
        "generic";
        "توقيت بنغلاديش",
        "standard";
        "توقيت بنغلاديش الرسمي",
        "daylight";
        "توقيت بنغلاديش الصيفي";
    }
}
"Bhutan";
{
    "long";
    {
        "standard";
        "توقيت بوتان";
    }
}
"Bolivia";
{
    "long";
    {
        "standard";
        "توقيت بوليفيا";
    }
}
"Brasilia";

```

```

    {
      "long";
      {
        "generic";
        "توقيت برازيليا",
        "standard";
        "توقيت برازيليا الرسمي",
        "daylight";
        "توقيت برازيليا الصيفي";
      }
    }
    "Brunei";
    {
      "long";
      {
        "standard";
        "توقيت بروناي";
      }
    }
    "Cape_Verde";
    {
      "long";
      {
        "generic";
        "توقيت الرأس الأخضر",
        "standard";
        "توقيت الرأس الأخضر الرسمي",
        "daylight";
        "توقيت الرأس الأخضر الصيفي";
      }
    }
    "Chamorro";
    {
      "long";
      {
        "standard";
        "توقيت تشامورو";
      }
    }
    "Chatham";
    {
      "long";
      {
        "generic";
        "توقيت تشاتام",
        "standard";
        "توقيت تشاتام الرسمي",
        "daylight";
        "توقيت تشاتام الصيفي";
      }
    }
    "Chile";
    {
      "long";
      {
        "generic";
        "توقيت شيلي",

```



```

        "standard";
        "توقيت شيلي الرسمي",
        "daylight";
        "توقيت شيلي الصيفي";
    }
}
"China";
{
    "long";
    {
        "generic";
        "توقيت الصين",
        "standard";
        "توقيت الصين الرسمي",
        "daylight";
        "توقيت الصين الصيفي";
    }
}
"Choibalsan";
{
    "long";
    {
        "generic";
        "توقيت شويبالسان",
        "standard";
        "توقيت شويبالسان الرسمي",
        "daylight";
        "التوقيت الصيفي لشويبالسان";
    }
}
"Christmas";
{
    "long";
    {
        "standard";
        "توقيت جزر الكريسماس";
    }
}
"Cocos";
{
    "long";
    {
        "standard";
        "توقيت جزر كوكوس";
    }
}
"Colombia";
{
    "long";
    {
        "generic";
        "توقيت كولومبيا",
        "standard";
        "توقيت كولومبيا الرسمي",
        "daylight";
        "توقيت كولومبيا الصيفي";
    }
}

```

```

    }
    "Cook";
    {
        "long";
        {
            "generic";
            "توقيت جزر كووك",
            "standard";
            "توقيت جزر كووك الرسمي",
            "daylight";
            "توقيت جزر كووك الصيفي";
        }
    }
    "Cuba";
    {
        "long";
        {
            "generic";
            "توقيت كوبا",
            "standard";
            "توقيت كوبا الرسمي",
            "daylight";
            "توقيت كوبا الصيفي";
        }
    }
    "Davis";
    {
        "long";
        {
            "standard";
            "توقيت دافيز";
        }
    }
    "DumontDUrville";
    {
        "long";
        {
            "standard";
            "توقيت دي مونت دو روفيل";
        }
    }
    "East_Timor";
    {
        "long";
        {
            "standard";
            "توقيت تيمور الشرقية";
        }
    }
    "Easter";
    {
        "long";
        {
            "generic";
            "توقيت جزيرة استر",
            "standard";
            "توقيت جزيرة استر الرسمي",

```

```

        "daylight";
        "توقيت جزيرة استر الصيفي";
    }
}
"Ecuador";
{
    "long";
    {
        "standard";
        "توقيت الإكوادور";
    }
}
"Europe_Central";
{
    "long";
    {
        "generic";
        "توقيت وسط أوروبا",
        "standard";
        "توقيت وسط أوروبا الرسمي",
        "daylight";
        "توقيت وسط أوروبا الصيفي";
    }
}
"Europe_Eastern";
{
    "long";
    {
        "generic";
        "توقيت شرق أوروبا",
        "standard";
        "توقيت شرق أوروبا الرسمي",
        "daylight";
        "توقيت شرق أوروبا الصيفي";
    }
}
"Europe_Further_Eastern";
{
    "long";
    {
        "standard";
        "التوقيت الأوروبي (أكثر شرقًا)";
    }
}
"Europe_Western";
{
    "long";
    {
        "generic";
        "توقيت غرب أوروبا",
        "standard";
        "توقيت غرب أوروبا الرسمي",
        "daylight";
        "توقيت غرب أوروبا الصيفي";
    }
}
"Falkland";

```

```

        {
            "long";
            {
                "generic";
                "توقيت جزر فوكلاند",
                "standard";
                "توقيت جزر فوكلاند الرسمي",
                "daylight";
                "توقيت جزر فوكلاند الصيفي";
            }
        }
        "Fiji";
        {
            "long";
            {
                "generic";
                "توقيت فيجي",
                "standard";
                "توقيت فيجي الرسمي",
                "daylight";
                "توقيت فيجي الصيفي";
            }
        }
        "French_Guiana";
        {
            "long";
            {
                "standard";
                "توقيت غايانا الفرنسية";
            }
        }
        "French_Southern";
        {
            "long";
            {
                "standard";
                "توقيت المقاطعات الفرنسية الجنوبية";
            }
        }
        "Galapagos";
        {
            "long";
            {
                "standard";
                "توقيت غلاباغوس";
            }
        }
        "Gambier";
        {
            "long";
            {
                "standard";
                "توقيت جامبير";
            }
        }
        "Georgia";

```

```

    {
      "long";
      {
        "generic";
        "توقيت جورجيا",
        "standard";
        "توقيت جورجيا الرسمي",
        "daylight";
        "توقيت جورجيا الصيفي";
      }
    }
    "Gilbert_Islands";
    {
      "long";
      {
        "standard";
        "توقيت جزر جيلبرت";
      }
    }
    "GMT";
    {
      "long";
      {
        "standard";
        "توقيت غرينتش";
      }
    }
    "Greenland_Eastern";
    {
      "long";
      {
        "generic";
        "توقيت شرق غرينلاند",
        "standard";
        "توقيت شرق غرينلاند الرسمي",
        "daylight";
        "توقيت شرق غرينلاند الصيفي";
      }
    }
    "Greenland_Western";
    {
      "long";
      {
        "generic";
        "توقيت غرب غرينلاند",
        "standard";
        "توقيت غرب غرينلاند الرسمي",
        "daylight";
        "توقيت غرب غرينلاند الصيفي";
      }
    }
    "Guam";
    {
      "long";
      {
        "standard";
        "توقيت غوام";
      }
    }
  }

```

```

    }
  }
  "Gulf";
  {
    "long";
    {
      "standard";
      "توقيت الخليج";
    }
  }
  "Guyana";
  {
    "long";
    {
      "standard";
      "توقيت غيانا";
    }
  }
  "Hawaii_Aleutian";
  {
    "long";
    {
      "generic";
      "توقيت هاواي ألوتيان",
      "standard";
      "توقيت هاواي ألوتيان الرسمي",
      "daylight";
      "توقيت هاواي ألوتيان الصيفي";
    }
  }
  "Hong_Kong";
  {
    "long";
    {
      "generic";
      "توقيت هونغ كونغ",
      "standard";
      "توقيت هونغ كونغ الرسمي",
      "daylight";
      "توقيت هونغ كونغ الصيفي";
    }
  }
  "Hovd";
  {
    "long";
    {
      "generic";
      "توقيت هوفد",
      "standard";
      "توقيت هوفد الرسمي",
      "daylight";
      "توقيت هوفد الصيفي";
    }
  }
  "India";
  {
    "long";

```

```
{
    "standard";
    "توقيت الهند";
}
}
"Indian_Ocean";
{
    "long";
    {
        "standard";
        "توقيت المحيط الهندي";
    }
}
"Indochina";
{
    "long";
    {
        "standard";
        "توقيت الهند الصينية";
    }
}
"Indonesia_Central";
{
    "long";
    {
        "standard";
        "توقيت وسط إندونيسيا";
    }
}
"Indonesia_Eastern";
{
    "long";
    {
        "standard";
        "توقيت شرق إندونيسيا";
    }
}
"Indonesia_Western";
{
    "long";
    {
        "standard";
        "توقيت غرب إندونيسيا";
    }
}
"Iran";
{
    "long";
    {
        "generic";
        "توقيت إيران",
        "standard";
        "توقيت إيران الرسمي",
        "daylight";
        "توقيت إيران الصيفي";
    }
}
}
```

```
"Irkutsk";
{
  "long";
  {
    "generic";
    "توقيت إركوتسك",
    "standard";
    "توقيت إركوتسك الرسمي",
    "daylight";
    "توقيت إركوتسك الصيفي";
  }
}
"Israel";
{
  "long";
  {
    "generic";
    "توقيت إسرائيل",
    "standard";
    "توقيت إسرائيل الرسمي",
    "daylight";
    "توقيت إسرائيل الصيفي";
  }
}
"Japan";
{
  "long";
  {
    "generic";
    "توقيت اليابان",
    "standard";
    "توقيت اليابان الرسمي",
    "daylight";
    "توقيت اليابان الصيفي";
  }
}
"Kamchatka";
{
  "long";
  {
    "generic";
    "توقيت كامشاتكا",
    "standard";
    "توقيت بيتروبافلوفسك-كامشاتسكي",
    "daylight";
    "توقيت بيتروبافلوفسك-كامشاتسكي الصيفي";
  }
}
"Kazakhstan_Eastern";
{
  "long";
  {
    "standard";
    "توقيت شرق كازاخستان";
  }
}
"Kazakhstan_Western";
```



```

    {
      "long";
      {
        "standard";
        "توقيت غرب كازاخستان";
      }
    }
    "Korea";
    {
      "long";
      {
        "generic";
        "توقيت كوريا",
        "standard";
        "توقيت كوريا الرسمي",
        "daylight";
        "توقيت كوريا الصيفي";
      }
    }
    "Kosrae";
    {
      "long";
      {
        "standard";
        "توقيت كوسرا";
      }
    }
    "Krasnoyarsk";
    {
      "long";
      {
        "generic";
        "توقيت كراسنويارسك",
        "standard";
        "توقيت كراسنويارسك الرسمي",
        "daylight";
        "التوقيت الصيفي لكراسنويارسك";
      }
    }
    "Kyrgystan";
    {
      "long";
      {
        "standard";
        "توقيت قيرغيزستان";
      }
    }
    "Line_Islands";
    {
      "long";
      {
        "standard";
        "توقيت جزر لاين";
      }
    }
    "Lord_Howe";
    {

```

```

        "long";
        {
            "generic";
            "توقيت لورد هاو",
            "standard";
            "توقيت لورد هاو الرسمي",
            "daylight";
            "التوقيت الصيفي للورد هاو";
        }
    }
    "Macquarie";
    {
        "long";
        {
            "standard";
            "توقيت ماكوارى";
        }
    }
    "Magadan";
    {
        "long";
        {
            "generic";
            "توقيت ماغادان",
            "standard";
            "توقيت ماغادان الرسمي",
            "daylight";
            "توقيت ماغادان الصيفي";
        }
    }
    "Malaysia";
    {
        "long";
        {
            "standard";
            "توقيت ماليزيا";
        }
    }
    "Maldives";
    {
        "long";
        {
            "standard";
            "توقيت المالديف";
        }
    }
    "Marquesas";
    {
        "long";
        {
            "standard";
            "توقيت ماركيساس";
        }
    }
    "Marshall_Islands";
    {
        "long";

```

```

        {
            "standard";
            "توقيت جزر مارشال";
        }
    }
    "Mauritius";
    {
        "long";
        {
            "generic";
            "توقيت موريشيوس",
            "standard";
            "توقيت موريشيوس الرسمي",
            "daylight";
            "توقيت موريشيوس الصيفي";
        }
    }
    "Mawson";
    {
        "long";
        {
            "standard";
            "توقيت ماوسون";
        }
    }
    "Mexico_Northwest";
    {
        "long";
        {
            "generic";
            "توقيت شمال غرب المكسيك",
            "standard";
            "التوقيت الرسمي لشمال غرب المكسيك",
            "daylight";
            "التوقيت الصيفي لشمال غرب المكسيك";
        }
    }
    "Mexico_Pacific";
    {
        "long";
        {
            "generic";
            "توقيت المحيط الهادي للمكسيك",
            "standard";
            "توقيت المحيط الهادي الرسمي للمكسيك",
            "daylight";
            "توقيت المحيط الهادي الصيفي للمكسيك";
        }
    }
    "Mongolia";
    {
        "long";
        {
            "generic";
            "توقيت أولان باتور",
            "standard";
            "توقيت أولان باتور الرسمي",

```

```

        "daylight";
        "توقيت أولان باتور الصيفي";
    }
}
"Moscow";
{
    "long";
    {
        "generic";
        "توقيت موسكو",
        "standard";
        "توقيت موسكو الرسمي",
        "daylight";
        "توقيت موسكو الصيفي";
    }
}
"Myanmar";
{
    "long";
    {
        "standard";
        "توقيت ميانمار";
    }
}
"Nauru";
{
    "long";
    {
        "standard";
        "توقيت ناورو";
    }
}
"Nepal";
{
    "long";
    {
        "standard";
        "توقيت نيبال";
    }
}
}
"New_Caledonia";
{
    "long";
    {
        "generic";
        "توقيت كاليدونيا الجديدة",
        "standard";
        "توقيت كاليدونيا الجديدة الرسمي",
        "daylight";
        "توقيت كاليدونيا الجديدة الصيفي";
    }
}
}
"New_Zealand";
{
    "long";
    {
        "generic";

```

```

        "توقيت نيوزيلندا",
        "standard";
        "توقيت نيوزيلندا الرسمي",
        "daylight";
        "توقيت نيوزيلندا الصيفي";
    }
}
"Newfoundland";
{
    "long";
    {
        "generic";
        "توقيت نيوفاوندلاند",
        "standard";
        "توقيت نيوفاوندلاند الرسمي",
        "daylight";
        "توقيت نيوفاوندلاند الصيفي";
    }
}
"Niue";
{
    "long";
    {
        "standard";
        "توقيت نيوي";
    }
}
"Norfolk";
{
    "long";
    {
        "standard";
        "توقيت جزيرة نورفولك";
    }
}
"Noronha";
{
    "long";
    {
        "generic";
        "توقيت فيرناندو دي نورونها",
        "standard";
        "توقيت فرناندو دي نورونها الرسمي",
        "daylight";
        "توقيت فرناندو دي نورونها الصيفي";
    }
}
"North_Mariana";
{
    "long";
    {
        "standard";
        "توقيت جزر ماريانا الشمالية";
    }
}
"Novosibirsk";
{

```

```

        "long";
        {
            "generic";
            "توقيت نوفوسيبيرسك",
            "standard";
            "توقيت نوفوسيبيرسك الرسمي",
            "daylight";
            "توقيت نوفوسيبيرسك الصيفي";
        }
    }
    "Omsk";
    {
        "long";
        {
            "generic";
            "توقيت أومسك",
            "standard";
            "توقيت أومسك الرسمي",
            "daylight";
            "توقيت أومسك الصيفي";
        }
    }
    "Pakistan";
    {
        "long";
        {
            "generic";
            "توقيت باكستان",
            "standard";
            "توقيت باكستان الرسمي",
            "daylight";
            "توقيت باكستان الصيفي";
        }
    }
    "Palau";
    {
        "long";
        {
            "standard";
            "توقيت بالاو";
        }
    }
    "Papua_New_Guinea";
    {
        "long";
        {
            "standard";
            "توقيت بابوا غينيا الجديدة";
        }
    }
    "Paraguay";
    {
        "long";
        {
            "generic";
            "توقيت باراغواي",
            "standard";

```

```

        "توقيت باراغواي الرسمي",
        "daylight";
        "توقيت باراغواي الصيفي";
    }
}
"Peru";
{
    "long";
    {
        "generic";
        "توقيت بيرو",
        "standard";
        "توقيت بيرو الرسمي",
        "daylight";
        "توقيت بيرو الصيفي";
    }
}
"Philippines";
{
    "long";
    {
        "generic";
        "توقيت الفلبين",
        "standard";
        "توقيت الفلبين الرسمي",
        "daylight";
        "توقيت الفلبين الصيفي";
    }
}
"Phoenix_Islands";
{
    "long";
    {
        "standard";
        "توقيت جزر فينكس";
    }
}
"Pierre_Miquelon";
{
    "long";
    {
        "generic";
        "توقيت سانت بيير وميكلون",
        "standard";
        "توقيت سانت بيير وميكلون الرسمي",
        "daylight";
        "توقيت سانت بيير وميكلون الصيفي";
    }
}
"Pitcairn";
{
    "long";
    {
        "standard";
        "توقيت بيتكيرن";
    }
}
}

```

```
"Ponape";
{
  "long";
  {
    "standard";
    "توقيت بونابي";
  }
}
"Pyongyang";
{
  "long";
  {
    "standard";
    "توقيت بيونغ يانغ";
  }
}
"Reunion";
{
  "long";
  {
    "standard";
    "توقيت ريونيون";
  }
}
"Rothera";
{
  "long";
  {
    "standard";
    "توقيت روثيرا";
  }
}
"Sakhalin";
{
  "long";
  {
    "generic";
    "توقيت ساخالين",
    "standard";
    "توقيت ساخالين الرسمي",
    "daylight";
    "توقيت ساخالين الصيفي";
  }
}
"Samara";
{
  "long";
  {
    "generic";
    "توقيت سامارا",
    "standard";
    "توقيت سمارة",
    "daylight";
    "توقيت سمارة الصيفي";
  }
}
"Samoa";
```



```
{
  "long";
  {
    "generic";
    "توقيت ساموا",
    "standard";
    "توقيت ساموا الرسمي",
    "daylight";
    "توقيت ساموا الصيفي";
  }
}
"Seychelles";
{
  "long";
  {
    "standard";
    "توقيت سيشل";
  }
}
"Singapore";
{
  "long";
  {
    "standard";
    "توقيت سنغافورة";
  }
}
"Solomon";
{
  "long";
  {
    "standard";
    "توقيت جزر سليمان";
  }
}
"South_Georgia";
{
  "long";
  {
    "standard";
    "توقيت جنوب جورجيا";
  }
}
"Suriname";
{
  "long";
  {
    "standard";
    "توقيت سورينام";
  }
}
"Syowa";
{
  "long";
  {
    "standard";
    "توقيت سايووا";
  }
}
```

```
    }  
  }  
  "Tahiti";  
  {  
    "long";  
    {  
      "standard";  
      "توقيت تاهيتي";  
    }  
  }  
  "Taipei";  
  {  
    "long";  
    {  
      "generic";  
      "توقيت تايبيه",  
      "standard";  
      "توقيت تايبيه الرسمي",  
      "daylight";  
      "توقيت تايبيه الصيفي";  
    }  
  }  
  "Tajikistan";  
  {  
    "long";  
    {  
      "standard";  
      "توقيت طاجكستان";  
    }  
  }  
  "Tokelau";  
  {  
    "long";  
    {  
      "standard";  
      "توقيت توكيلاو";  
    }  
  }  
  "Tonga";  
  {  
    "long";  
    {  
      "generic";  
      "توقيت تونغا",  
      "standard";  
      "توقيت تونغا الرسمي",  
      "daylight";  
      "توقيت تونغا الصيفي";  
    }  
  }  
  "Truk";  
  {  
    "long";  
    {  
      "standard";  
      "توقيت شوك";  
    }  
  }
```

```
}
"Turkmenistan";
{
  "long";
  {
    "generic";
    "توقيت تركمانستان",
    "standard";
    "توقيت تركمانستان الرسمي",
    "daylight";
    "توقيت تركمانستان الصيفي";
  }
}
"Tuvalu";
{
  "long";
  {
    "standard";
    "توقيت توفالو";
  }
}
"Uruguay";
{
  "long";
  {
    "generic";
    "توقيت أوروغواي",
    "standard";
    "توقيت أوروغواي الرسمي",
    "daylight";
    "توقيت أوروغواي الصيفي";
  }
}
"Uzbekistan";
{
  "long";
  {
    "generic";
    "توقيت أوزبكستان",
    "standard";
    "توقيت أوزبكستان الرسمي",
    "daylight";
    "توقيت أوزبكستان الصيفي";
  }
}
"Vanuatu";
{
  "long";
  {
    "generic";
    "توقيت فانواتو",
    "standard";
    "توقيت فانواتو الرسمي",
    "daylight";
    "توقيت فانواتو الصيفي";
  }
}
}
```

```

    "Venezuela";
    {
        "long";
        {
            "standard";
            "توقيت فنزويلا";
        }
    }
    "Vladivostok";
    {
        "long";
        {
            "generic";
            "توقيت فلاديفوستوك",
            "standard";
            "توقيت فلاديفوستوك الرسمي",
            "daylight";
            "توقيت فلاديفوستوك الصيفي";
        }
    }
    "Volgograd";
    {
        "long";
        {
            "generic";
            "توقيت فولغوغراد",
            "standard";
            "توقيت فولغوغراد الرسمي",
            "daylight";
            "توقيت فولغوغراد الصيفي";
        }
    }
    "Vostok";
    {
        "long";
        {
            "standard";
            "توقيت فوستوك";
        }
    }
    "Wake";
    {
        "long";
        {
            "standard";
            "توقيت جزيرة ويك";
        }
    }
    "Wallis";
    {
        "long";
        {
            "standard";
            "توقيت واليس و فوتونا";
        }
    }
    "Yakutsk";

```

```

        {
            "long";
            {
                "generic";
                "توقيت ياكوتسك",
                "standard";
                "توقيت ياكوتسك الرسمي",
                "daylight";
                "توقيت ياكوتسك الصيفي";
            }
        }
        "Yekaterinburg";
        {
            "long";
            {
                "generic";
                "توقيت يكاترينبورغ",
                "standard";
                "توقيت يكاترينبورغ الرسمي",
                "daylight";
                "توقيت يكاترينبورغ الصيفي";
            }
        }
    }
}

```

[Functional-component]

CA-GREGORIAN.JSON

```

{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "dates": {
        "calendars": {
          "gregorian": {
            "months": {
              "format": {
                "abbreviated": {
                  "1": "يناير",
                  "2": "فبراير",
                  "3": "مارس",
                  "4": "أبريل",
                  "5": "مايو",
                  "6": "يونيو",

```

```

        "7": "يوليو",
        "8": "أغسطس",
        "9": "سبتمبر",
        "10": "أكتوبر",
        "11": "نوفمبر",
        "12": "ديسمبر"
    },
    "narrow": {
        "1": "ي",
        "2": "ف",
        "3": "م",
        "4": "أ",
        "5": "و",
        "6": "ن",
        "7": "ل",
        "8": "غ",
        "9": "س",
        "10": "ك",
        "11": "ب",
        "12": "د"
    },
    "wide": {
        "1": "يناير",
        "2": "فبراير",
        "3": "مارس",
        "4": "أبريل",
        "5": "مايو",
        "6": "يونيو",
        "7": "يوليو",
        "8": "أغسطس",
        "9": "سبتمبر",
        "10": "أكتوبر",
        "11": "نوفمبر",
        "12": "ديسمبر"
    }
},
"stand-alone": {
    "abbreviated": {
        "1": "يناير",
        "2": "فبراير",
        "3": "مارس",
        "4": "أبريل",
        "5": "مايو",
        "6": "يونيو",
        "7": "يوليو",
        "8": "أغسطس",
        "9": "سبتمبر",
        "10": "أكتوبر",
        "11": "نوفمبر",
        "12": "ديسمبر"
    },
    "narrow": {
        "1": "ي",
        "2": "ف",
        "3": "م",
        "4": "أ",
        "5": "و",

```

```

        "6": "ن",
        "7": "ل",
        "8": "غ",
        "9": "س",
        "10": "ك",
        "11": "ب",
        "12": "د"
    },
    "wide": {
        "1": "يناير",
        "2": "فبراير",
        "3": "مارس",
        "4": "أبريل",
        "5": "مايو",
        "6": "يونيو",
        "7": "يوليو",
        "8": "أغسطس",
        "9": "سبتمبر",
        "10": "أكتوبر",
        "11": "نوفمبر",
        "12": "ديسمبر"
    }
},
"days": {
    "format": {
        "abbreviated": {
            "sun": "الأحد",
            "mon": "الاثنين",
            "tue": "الثلاثاء",
            "wed": "الأربعاء",
            "thu": "الخميس",
            "fri": "الجمعة",
            "sat": "السبت"
        },
        "narrow": {
            "sun": "ح",
            "mon": "ن",
            "tue": "ث",
            "wed": "ر",
            "thu": "خ",
            "fri": "ج",
            "sat": "س"
        },
        "short": {
            "sun": "أحد",
            "mon": "إثنين",
            "tue": "ثلاثاء",
            "wed": "أربعاء",
            "thu": "خميس",
            "fri": "جمعة",
            "sat": "سبت"
        },
        "wide": {
            "sun": "الأحد",
            "mon": "الاثنين",
            "tue": "الثلاثاء",

```

```

        "wed": "الأربعاء",
        "thu": "الخميس",
        "fri": "الجمعة",
        "sat": "السبت"
    },
    },
    "stand-alone": {
        "abbreviated": {
            "sun": "الأحد",
            "mon": "الاثنين",
            "tue": "الثلاثاء",
            "wed": "الأربعاء",
            "thu": "الخميس",
            "fri": "الجمعة",
            "sat": "السبت"
        },
        "narrow": {
            "sun": "ح",
            "mon": "ن",
            "tue": "ث",
            "wed": "ر",
            "thu": "خ",
            "fri": "ج",
            "sat": "س"
        },
        "short": {
            "sun": "أحد",
            "mon": "إثنين",
            "tue": "ثلاثاء",
            "wed": "أربعاء",
            "thu": "خميس",
            "fri": "جمعة",
            "sat": "سبت"
        },
        "wide": {
            "sun": "الأحد",
            "mon": "الاثنين",
            "tue": "الثلاثاء",
            "wed": "الأربعاء",
            "thu": "الخميس",
            "fri": "الجمعة",
            "sat": "السبت"
        }
    },
    },
    "quarters": {
        "format": {
            "abbreviated": {
                "1": "الربع الأول",
                "2": "الربع الثاني",
                "3": "الربع الثالث",
                "4": "الربع الرابع"
            },
            "narrow": {
                "1": "١",
                "2": "٢",
                "3": "٣",

```



```

        "4": "٤"
      },
      "wide": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      }
    },
    "stand-alone": {
      "abbreviated": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      },
      "narrow": {
        "1": "١",
        "2": "٢",
        "3": "٣",
        "4": "٤"
      },
      "wide": {
        "1": "الربع الأول",
        "2": "الربع الثاني",
        "3": "الربع الثالث",
        "4": "الربع الرابع"
      }
    }
  },
  "dayPeriods": {
    "format": {
      "abbreviated": {
        "am": "ص",
        "pm": "م",
        "morning1": "فجرًا",
        "morning2": "ص",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
      },
      "narrow": {
        "am": "ص",
        "pm": "م",
        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
      },
      "wide": {
        "am": "ص",
        "pm": "م",

```

```

        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    }
},
"stand-alone": {
    "abbreviated": {
        "am": "ص",
        "pm": "م",
        "morning1": "فجرًا",
        "morning2": "ص",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    },
    "narrow": {
        "am": "ص",
        "pm": "م",
        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    },
    "wide": {
        "am": "صباحًا",
        "pm": "مساءً",
        "morning1": "فجرًا",
        "morning2": "صباحًا",
        "afternoon1": "ظهرًا",
        "afternoon2": "بعد الظهر",
        "evening1": "مساءً",
        "night1": "منتصف الليل",
        "night2": "ليلاً"
    }
}
},
"eras": {
    "eraNames": {
        "0": "قبل الميلاد",
        "0-alt-variant": "قبل الحقبة الحالية",
        "1": "ميلادي",
        "1-alt-variant": "بعد الميلاد"
    },
    "eraAbbr": {
        "0": "ق.م",
        "0-alt-variant": "ق. م.",
        "1": "م",
        "1-alt-variant": "ب.م"
    }
}

```

```

    },
    "eraNarrow": {
      "0": "ق.م",
      "0-alt-variant": "ق. م",
      "1": "م",
      "1-alt-variant": "م.ب"
    }
  },
  "dateFormats": {
    "full": "EEEE, d MMMM y",
    "long": "d MMMM y",
    "medium": "dd/MM/y",
    "short": "d/M/y"
  },
  "timeFormats": {
    "full": "h:mm:ss a zzzz",
    "long": "h:mm:ss a z",
    "medium": "h:mm:ss a",
    "short": "h:mm a"
  },
  "dateTimeFormats": {
    "full": "{1} {0}",
    "long": "{1} {0}",
    "medium": "{1} {0}",
    "short": "{1} {0}",
    "availableFormats": {
      "Bh": "h B",
      "Bhm": "h:mm B",
      "Bhms": "h:mm:ss B",
      "d": "d",
      "E": "ccc",
      "EBhm": "E h:mm B",
      "EBhms": "E h:mm:ss B",
      "Ed": "E, d",
      "Ehm": "E h:mm a",
      "EHm": "E HH:mm",
      "Ehms": "E h:mm:ss a",
      "EHms": "E HH:mm:ss",
      "Gy": "y G",
      "GyMMM": "MMM y G",
      "GyMMMd": "d MMM y G",
      "GyMMMEd": "E, d MMM y G",
      "h": "h a",
      "H": "HH",
      "hm": "h:mm a",
      "Hm": "HH:mm",
      "hms": "h:mm:ss a",
      "Hms": "HH:mm:ss",
      "hmsv": "h:mm:ss a v",
      "Hmsv": "HH:mm:ss v",
      "hmv": "h:mm a v",
      "Hmv": "HH:mm v",
      "M": "L",
      "Md": "d/M",
      "MEd": "E, d/M",
      "MMdd": "dd/MM",
      "MMM": "LLL",

```

```

"MMMd": "d MMM",
"MMMED": "E، d MMM",
"MMMMd": "d MMMM",
"MMMMEd": "E، d MMMM",
"MMMMW-count-zero": "الأسبوع W من MMM",
"MMMMW-count-one": "الأسبوع W من MMM",
"MMMMW-count-two": "الأسبوع W من MMM",
"MMMMW-count-few": "الأسبوع W من MMM",
"MMMMW-count-many": "الأسبوع W من MMM",
"MMMMW-count-other": "الأسبوع W من MMM",
"ms": "mm:ss",
"Y": "Y",
"yM": "M/y",
"yMd": "d/M/y",
"yMED": "E، d/M/y",
"yMM": "MM/y",
"yMMM": "MMM y",
"yMMMd": "d MMM y",
"yMMMED": "E، d MMM y",
"yMMMM": "MMMM y",
"yQQQ": "QQQ y",
"yQQQQ": "QQQQ y",
"yw-count-zero": "Y من سنة w الأسبوع",
"yw-count-one": "Y من سنة w الأسبوع",
"yw-count-two": "Y من سنة w الأسبوع",
"yw-count-few": "Y من سنة w الأسبوع",
"yw-count-many": "Y من سنة w الأسبوع",
"yw-count-other": "Y من سنة w الأسبوع",
},
"appendItems": {
  "Day": "{0} ({2}: {1})",
  "Day-Of-Week": "{0} {1}",
  "Era": "{1} {0}",
  "Hour": "{0} ({2}: {1})",
  "Minute": "{0} ({2}: {1})",
  "Month": "{0} ({2}: {1})",
  "Quarter": "{0} ({2}: {1})",
  "Second": "{0} ({2}: {1})",
  "Timezone": "{0} {1}",
  "Week": "{0} ({2}: {1})",
  "Year": "{1} {0}"
},
"intervalFormats": {
  "intervalFormatFallback": "{0} - {1}",
  "d": {
    "d": "d-d"
  },
  "h": {
    "a": "h a - h a",
    "h": "h-h a"
  },
  "H": {
    "H": "HH-HH"
  },
  "hm": {
    "a": "h:mm a - h:mm a",
    "h": "h:mm-h:mm a",

```

```

        "m": "h:mm-h:mm a"
    },
    "Hm": {
        "H": "HH:mm-HH:mm",
        "m": "HH:mm-HH:mm"
    },
    "hmv": {
        "a": "h:mm a - h:mm a v",
        "h": "h:mm-h:mm a v",
        "m": "h:mm-h:mm a v"
    },
    "Hmv": {
        "H": "HH:mm-HH:mm v",
        "m": "HH:mm-HH:mm v"
    },
    "hv": {
        "a": "h a - h a v",
        "h": "h-h a v"
    },
    "Hv": {
        "H": "HH-HH v"
    },
    "M": {
        "M": "M-M"
    },
    "Md": {
        "d": "M/d - M/d",
        "M": "M/d - M/d"
    },
    "MEd": {
        "d": "E, d/M - E, d/M",
        "M": "E, d/M - E, d/M"
    },
    "MMM": {
        "M": "MMM-MMM"
    },
    "MMMd": {
        "d": "d-d MMM",
        "M": "d MMM - d MMM"
    },
    "MMMEd": {
        "d": "E, d - E, d MMM",
        "M": "E, d MMM - E, d MMM"
    },
    "MMMM": {
        "M": "LLLL-LLLL"
    },
    "Y": {
        "Y": "Y-Y"
    },
    "YM": {
        "M": "M/Y - M/Y",
        "Y": "M/Y - M/Y"
    },
    "yMd": {
        "d": "d/M/Y - d/M/Y",
        "M": "d/M/Y - d/M/Y",
    }

```

```
"y": "d/M/y - d/M/y"
},
"yMEd": {
  "d": "E. dd/MM/y - E. dd/MM/y",
  "M": "E. d/M/y - E. d/M/y",
  "y": "E. d/M/y - E. d/M/y"
},
"yMMM": {
  "M": "MMM - MMM. y",
  "y": "MMM. y - MMM. y"
},
"yMMMd": {
  "d": "d-d MMM. y",
  "M": "d MMM - d MMM. y",
  "y": "d MMM. y - d MMM. y"
},
"yMMMEd": {
  "d": "E. d - E. d MMM. y",
  "M": "E. d MMM - E. d MMM. y",
  "y": "E. d MMM. y - E. d MMM. y"
},
"yMMMM": {
  "M": "MMMM - MMMM. y",
  "y": "MMMM. y - MMMM. y"
}
}
}
}
}
}
}
```

CA-GREGORIAN.JSX

```
{
    "main";
    {
        "ar";
        {
            "identity";
            {
                "version";
                {
                    "_number";
                    "$Revision: 13686 $",
                    "_cldrVersion";
                    "32";
                }
                "language";
                "ar";
            }
        }
        "dates";
        {
            "calendars";
```

```

    {
      "gregorian";
      {
        "months";
        {
          "format";
          {
            "abbreviated";
            {
              "1";
              "يناير",
              "2";
              "فبراير",
              "3";
              "مارس",
              "4";
              "أبريل",
              "5";
              "مايو",
              "6";
              "يونيو",
              "7";
              "يوليو",
              "8";
              "أغسطس",
              "9";
              "سبتمبر",
              "10";
              "أكتوبر",
              "11";
              "نوفمبر",
              "12";
              "ديسمبر";
            }
            "narrow";
            {
              "1";
              "ي",
              "2";
              "ف",
              "3";
              "م",
              "4";
              "أ",
              "5";
              "و",
              "6";
              "ن",
              "7";
              "ل",
              "8";
              "غ",
              "9";
              "س",
              "10";
              "ك",
              "11";
            }
          }
        }
      }
    }
  
```

```

        "ب",
        "12";
        "د";
    }
    "wide";
    {
        "1";
        "يناير",
        "2";
        "فبراير",
        "3";
        "مارس",
        "4";
        "أبريل",
        "5";
        "مايو",
        "6";
        "يونيو",
        "7";
        "يوليو",
        "8";
        "أغسطس",
        "9";
        "سبتمبر",
        "10";
        "أكتوبر",
        "11";
        "نوفمبر",
        "12";
        "ديسمبر";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "يناير",
        "2";
        "فبراير",
        "3";
        "مارس",
        "4";
        "أبريل",
        "5";
        "مايو",
        "6";
        "يونيو",
        "7";
        "يوليو",
        "8";
        "أغسطس",
        "9";
        "سبتمبر",
        "10";
        "أكتوبر",
        "11";
    }
}

```



```
        "نوفمبر",  
        "12";  
        "ديسمبر";  
    }  
    "narrow";  
    {  
        "1";  
        "ي", "2";  
        "ف", "3";  
        "م", "4";  
        "أ", "5";  
        "و", "6";  
        "ن", "7";  
        "ل", "8";  
        "غ", "9";  
        "س", "10";  
        "ك", "11";  
        "ب", "12";  
        "د";  
    }  
    "wide";  
    {  
        "1";  
        "يناير",  
        "2";  
        "فبراير",  
        "3";  
        "مارس",  
        "4";  
        "أبريل",  
        "5";  
        "مايو",  
        "6";  
        "يونيو",  
        "7";  
        "يوليو",  
        "8";  
        "أغسطس",  
        "9";  
        "سبتمبر",  
        "10";  
        "أكتوبر",  
        "11";  
        "نوفمبر",  
        "12";  
        "ديسمبر";  
    }
```

```

    }
  }
}
"days";
{
  "format";
  {
    "abbreviated";
    {
      "sun";
      "الأحد",
      "mon";
      "الاثنين",
      "tue";
      "الثلاثاء",
      "wed";
      "الأربعاء",
      "thu";
      "الخميس",
      "fri";
      "الجمعة",
      "sat";
      "السبت";
    }
  }
  "narrow";
  {
    "sun";
    "ح",
    "mon";
    "ن",
    "tue";
    "ث",
    "wed";
    "ر",
    "thu";
    "خ",
    "fri";
    "ج",
    "sat";
    "س";
  }
  "short";
  {
    "sun";
    "أحد",
    "mon";
    "إثنين",
    "tue";
    "ثلاثاء",
    "wed";
    "أربعاء",
    "thu";
    "خميس",
    "fri";
    "جمعة",
    "sat";
    "سبت";
  }
}

```

```
}
"wide";
{
  "sun";
  "الأحد",
  "mon";
  "الاثنين",
  "tue";
  "الثلاثاء",
  "wed";
  "الأربعاء",
  "thu";
  "الخميس",
  "fri";
  "الجمعة",
  "sat";
  "السبت";
}
}
"stand-alone";
{
  "abbreviated";
  {
    "sun";
    "الأحد",
    "mon";
    "الاثنين",
    "tue";
    "الثلاثاء",
    "wed";
    "الأربعاء",
    "thu";
    "الخميس",
    "fri";
    "الجمعة",
    "sat";
    "السبت";
  }
  "narrow";
  {
    "sun";
    "ح",
    "mon";
    "ن",
    "tue";
    "ث",
    "wed";
    "ر",
    "thu";
    "خ",
    "fri";
    "ج",
    "sat";
    "س";
  }
  "short";
  {
```

```

        "sun";
        "أحد",
        "mon";
        "إثنين",
        "tue";
        "ثلاثاء",
        "wed";
        "أربعاء",
        "thu";
        "خميس",
        "fri";
        "جمعة",
        "sat";
        "سبت";
    }
    "wide";
    {
        "sun";
        "الأحد",
        "mon";
        "الاثنين",
        "tue";
        "الثلاثاء",
        "wed";
        "الأربعاء",
        "thu";
        "الخميس",
        "fri";
        "الجمعة",
        "sat";
        "السبت";
    }
}
"quarters";
{
    "format";
    {
        "abbreviated";
        {
            "1";
            "الربع الأول",
            "2";
            "الربع الثاني",
            "3";
            "الربع الثالث",
            "4";
            "الربع الرابع";
        }
    }
    "narrow";
    {
        "1";
        "١",
        "2";
        "٢",
        "3";
        "٣",
    }
}

```

```

        "4";
        "٤";
    }
    "wide";
    {
        "1";
        "الربع الأول",
        "2";
        "الربع الثاني",
        "3";
        "الربع الثالث",
        "4";
        "الربع الرابع";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "1";
        "الربع الأول",
        "2";
        "الربع الثاني",
        "3";
        "الربع الثالث",
        "4";
        "الربع الرابع";
    }
    "narrow";
    {
        "1";
        "١",
        "2";
        "٢",
        "3";
        "٣",
        "4";
        "٤";
    }
    "wide";
    {
        "1";
        "الربع الأول",
        "2";
        "الربع الثاني",
        "3";
        "الربع الثالث",
        "4";
        "الربع الرابع";
    }
}
}
"dayPeriods";
{
    "format";
    {
        "abbreviated";

```

```

{
  "am";
  "ص",
  "pm";
  "م",
  "morning1";
  "فجرًا",
  "morning2";
  "ص",
  "afternoon1";
  "ظهرًا",
  "afternoon2";
  "بعد الظهر",
  "evening1";
  "مساءً",
  "night1";
  "منتصف الليل",
  "night2";
  "ليلاً";
}
"narrow";
{
  "am";
  "ص",
  "pm";
  "م",
  "morning1";
  "فجرًا",
  "morning2";
  "صباحًا",
  "afternoon1";
  "ظهرًا",
  "afternoon2";
  "بعد الظهر",
  "evening1";
  "مساءً",
  "night1";
  "منتصف الليل",
  "night2";
  "ليلاً";
}
"wide";
{
  "am";
  "ص",
  "pm";
  "م",
  "morning1";
  "فجرًا",
  "morning2";
  "صباحًا",
  "afternoon1";
  "ظهرًا",
  "afternoon2";
  "بعد الظهر",
  "evening1";
  "مساءً",

```

```

        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
}
"stand-alone";
{
    "abbreviated";
    {
        "am";
        "ص",
        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "ص",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
    "narrow";
    {
        "am";
        "ص",
        "pm";
        "م",
        "morning1";
        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
    "wide";
    {
        "am";
        "صباحًا",
        "pm";
        "مساءً",
        "morning1";

```

```

        "فجرًا",
        "morning2";
        "صباحًا",
        "afternoon1";
        "ظهرًا",
        "afternoon2";
        "بعد الظهر",
        "evening1";
        "مساءً",
        "night1";
        "منتصف الليل",
        "night2";
        "ليلاً";
    }
}
}
"eras";
{
    "eraNames";
    {
        "0";
        "قبل الميلاد",
        "0-alt-variant";
        "قبل الحقبة الحالية",
        "1";
        "ميلادي",
        "1-alt-variant";
        "بعد الميلاد";
    }
    "eraAbbr";
    {
        "0";
        "ق.م",
        "0-alt-variant";
        "ق.م.",
        "1";
        "م",
        "1-alt-variant";
        "م.ب";
    }
    "eraNarrow";
    {
        "0";
        "ق.م",
        "0-alt-variant";
        "ق.م.",
        "1";
        "م",
        "1-alt-variant";
        "م.ب";
    }
}
"dateFormats";
{
    "full";
    "EEEE, d MMMM y",
    "long";
}

```



```

        "d MMMM y",
        "medium";
        "dd/MM/y",
        "short";
        "d/M/y";
    }
    "timeFormats";
    {
        "full";
        "h:mm:ss a zzzz",
        "long";
        "h:mm:ss a z",
        "medium";
        "h:mm:ss a",
        "short";
        "h:mm a";
    }
    "dateTimeFormats";
    {
        "full";
        "{1} {0}",
        "long";
        "{1} {0}",
        "medium";
        "{1} {0}",
        "short";
        "{1} {0}",
        "availableFormats";
        {
            "Bh";
            "h B",
            "Bhm";
            "h:mm B",
            "Bhms";
            "h:mm:ss B",
            "d";
            "d",
            "E";
            "ccc",
            "EBhm";
            "E h:mm B",
            "EBhms";
            "E h:mm:ss B",
            "Ed";
            "E, d",
            "Ehm";
            "E h:mm a",
            "EHm";
            "E HH:mm",
            "Ehms";
            "E h:mm:ss a",
            "EHms";
            "E HH:mm:ss",
            "Gy";
            "y G",
            "GyMMM";
            "MMM y G",

```

```

        "GyMMMd";
        "d MMM y G",
        "GyMMMEd";
        "E, d MMM y G",
        "h";
        "h a",
        "H";
        "HH",
        "hm";
        "h:mm a",
        "Hm";
        "HH:mm",
        "hms";
        "h:mm:ss a",
        "Hms";
        "HH:mm:ss",
        "hmsv";
        "h:mm:ss a v",
        "Hmsv";
        "HH:mm:ss v",
        "hmv";
        "h:mm a v",
        "Hmv";
        "HH:mm v",
        "M";
        "L",
        "Md";
        "d/M",
        "MEd";
        "E, d/M",
        "MMdd";
        "dd/MM",
        "MMM";
        "LLL",
        "MMMd";
        "d MMM",
        "MMMEd";
        "E, d MMM",
        "MMMMd";
        "d MMMM",
        "MMMMEd";
        "E, d MMMM",
        "MMMMW-count-zero";
        "MMM من الأسبوع",
        "MMMMW-count-one";
        "MMM من الأسبوع",
        "MMMMW-count-two";
        "MMM من الأسبوع",
        "MMMMW-count-few";
        "MMM من الأسبوع",
        "MMMMW-count-many";
        "MMM من الأسبوع",
        "MMMMW-count-other";
        "MMM من الأسبوع",
        "ms";
        "mm:ss",
        "y";

```

```

        "Y",
        "yM";
    "M/y",
        "yMd";
    "d/M/y",
        "yMEd";
    "E, d/M/y",
        "yMM";
    "MM/y",
        "yMMM";
    "MMM y",
        "yMMMd";
    "d MMM y",
        "yMMMEd";
    "E, d MMM y",
        "yMMMM";
    "MMMM y",
        "yQQQ";
    "QQQ y",
        "yQQQQ";
    "QQQQ y",
        "yw-count-zero";
    "Y من سنة w الأسبوع",
        "yw-count-one";
    "Y من سنة w الأسبوع",
        "yw-count-two";
    "Y من سنة w الأسبوع",
        "yw-count-few";
    "Y من سنة w الأسبوع",
        "yw-count-many";
    "Y من سنة w الأسبوع",
        "yw-count-other";
    "Y من سنة w الأسبوع";
}
"appendItems";
{
    "Day";
    "{0} ({2}: {1})",
        "Day-Of-Week";
    "{0} {1}",
        "Era";
    "{1} {0}",
        "Hour";
    "{0} ({2}: {1})",
        "Minute";
    "{0} ({2}: {1})",
        "Month";
    "{0} ({2}: {1})",
        "Quarter";
    "{0} ({2}: {1})",
        "Second";
    "{0} ({2}: {1})",
        "Timezone";
    "{0} {1}",
        "Week";
    "{0} ({2}: {1})",
        "Year";
}

```

```

        "{1} {0}";
    }
    "intervalFormats";
    {
        "intervalFormatFallback";
        "{0} - {1}",
        "d";
        {
            "d";
            "d-d";
        }
        "h";
        {
            "a";
            "h a - h a",
            "h";
            "h-h a";
        }
        "H";
        {
            "H";
            "HH-HH";
        }
        "hm";
        {
            "a";
            "h:mm a - h:mm a",
            "h";
            "h:mm-h:mm a",
            "m";
            "h:mm-h:mm a";
        }
        "Hm";
        {
            "H";
            "HH:mm-HH:mm",
            "m";
            "HH:mm-HH:mm";
        }
        "hmv";
        {
            "a";
            "h:mm a - h:mm a v",
            "h";
            "h:mm-h:mm a v",
            "m";
            "h:mm-h:mm a v";
        }
        "Hmv";
        {
            "H";
            "HH:mm-HH:mm v",
            "m";
            "HH:mm-HH:mm v";
        }
        "hv";
        {

```

```

        "a";
        "h a - h a v",
        "h";
        "h-h a v";
    }
    "Hv";
    {
        "H";
        "HH-HH v";
    }
    "M";
    {
        "M";
        "M-M";
    }
    "Md";
    {
        "d";
        "M/d - M/d",
        "M";
        "M/d - M/d";
    }
    "MEd";
    {
        "d";
        "E, d/M - E, d/M",
        "M";
        "E, d/M - E, d/M";
    }
    "MMM";
    {
        "M";
        "MMM-MMM";
    }
    "MMMd";
    {
        "d";
        "d-d MMM",
        "M";
        "d MMM - d MMM";
    }
    "MMMEd";
    {
        "d";
        "E, d - E, d MMM",
        "M";
        "E, d MMM - E, d MMM";
    }
    "MMMM";
    {
        "M";
        "LLLL-LLLL";
    }
    "Y";
    {
        "Y";
        "Y-Y";
    }

```

```

    }
    "yM";
    {
        "M";
        "M/y - M/y",
        "Y";
        "M/y - M/y";
    }
    "yMd";
    {
        "d";
        "d/M/y - d/M/y",
        "M";
        "d/M/y - d/M/y",
        "Y";
        "d/M/y - d/M/y";
    }
    "yMEd";
    {
        "d";
        "E, dd/MM/y - E, dd/MM/y",
        "M";
        "E, d/M/y - E, d/M/y",
        "Y";
        "E, d/M/y - E, d/M/y";
    }
    "yMMM";
    {
        "M";
        "MMM - MMM, y",
        "Y";
        "MMM, y - MMM, y";
    }
    "yMMMd";
    {
        "d";
        "d-d MMM, y",
        "M";
        "d MMM - d MMM, y",
        "Y";
        "d MMM, y - d MMM, y";
    }
    "yMMMEd";
    {
        "d";
        "E, d - E, d MMM, y",
        "M";
        "E, d MMM - E, d MMM, y",
        "Y";
        "E, d MMM, y - E, d MMM, y";
    }
    "yMMMM";
    {
        "M";
        "MMMM - MMMM, y",
        "Y";
        "MMMM, y - MMMM, y";
    }

```

CURRENCIES.JSON

```
{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "numbers": {
        "currencies": {
          "ADP": {
            "displayName": "بيستا أندوري",
            "symbol": "ADP"
          },
          "AED": {
            "displayName": "درهم إماراتي",
            "displayName-count-zero": "درهم إماراتي",
            "displayName-count-one": "درهم إماراتي",
            "displayName-count-two": "درهم إماراتي",
            "displayName-count-few": "درهم إماراتي",
            "displayName-count-many": "درهم إماراتي",
            "displayName-count-other": "درهم إماراتي",
            "symbol": ".د.إ."
          },
          "AFA": {
            "displayName": "أفغاني - 2002-1927",
            "symbol": "AFA"
          },
          "AFN": {
            "displayName": "أفغاني",
            "displayName-count-zero": "أفغاني أفغانستاني",
            "displayName-count-one": "أفغاني أفغانستاني",
            "displayName-count-two": "أفغاني أفغانستاني",
            "displayName-count-few": "أفغاني أفغانستاني",
            "displayName-count-many": "أفغاني أفغانستاني",
            "displayName-count-other": "أفغاني أفغانستاني",
            "symbol": "AFN"
          },
          "ALK": {
            "displayName": "ALK",
            "symbol": "ALK"
          }
        }
      }
    }
  }
}
```

```

"ALL": {
  "displayName": "ليك ألباني",
  "displayName-count-zero": "ليك ألباني",
  "displayName-count-one": "ليك ألباني",
  "displayName-count-two": "ليك ألباني",
  "displayName-count-few": "ليك ألباني",
  "displayName-count-many": "ليك ألباني",
  "displayName-count-other": "ليك ألباني",
  "symbol": "ALL"
},
"AMD": {
  "displayName": "درام أرميني",
  "displayName-count-zero": "درام أرميني",
  "displayName-count-one": "درام أرميني",
  "displayName-count-two": "درام أرميني",
  "displayName-count-few": "درام أرميني",
  "displayName-count-many": "درام أرميني",
  "displayName-count-other": "درام أرميني",
  "symbol": "AMD"
},
"ANG": {
  "displayName": "غيلدر أنتيلي هولندي",
  "displayName-count-zero": "غيلدر أنتيلي هولندي",
  "displayName-count-one": "غيلدر أنتيلي هولندي",
  "displayName-count-two": "غيلدر أنتيلي هولندي",
  "displayName-count-few": "غيلدر أنتيلي هولندي",
  "displayName-count-many": "غيلدر أنتيلي هولندي",
  "displayName-count-other": "غيلدر أنتيلي هولندي",
  "symbol": "ANG"
},
"AOA": {
  "displayName": "كوانزا أنغولي",
  "displayName-count-zero": "كوانزا أنغولي",
  "displayName-count-one": "كوانزا أنغولي",
  "displayName-count-two": "كوانزا أنغولي",
  "displayName-count-few": "كوانزا أنغولي",
  "displayName-count-many": "كوانزا أنغولي",
  "displayName-count-other": "كوانزا أنغولي",
  "symbol": "AOA",
  "symbol-alt-narrow": "Kz"
},
"AOK": {
  "displayName": "كوانزا أنجولي - 1990-1977",
  "symbol": "AOK"
},
"AON": {
  "displayName": "كوانزا أنجولي جديدة - 2000-1990",
  "symbol": "AON"
},
"AOR": {
  "displayName": "كوانزا أنجولي معدلة - 1999 - 1995",
  "symbol": "AOR"
},
"ARA": {
  "displayName": "استرال أرجنتيني",
  "symbol": "ARA"
},

```



```

"ARL": {
  "displayName": "ARL",
  "symbol": "ARL"
},
"ARM": {
  "displayName": "ARM",
  "symbol": "ARM"
},
"ARP": {
  "displayName": "1985-1983 - بيزو أرجنتيني",
  "symbol": "ARP"
},
"ARS": {
  "displayName": "بيزو أرجنتيني",
  "displayName-count-zero": "بيزو أرجنتيني",
  "displayName-count-one": "بيزو أرجنتيني",
  "displayName-count-two": "بيزو أرجنتيني",
  "displayName-count-few": "بيزو أرجنتيني",
  "displayName-count-many": "بيزو أرجنتيني",
  "displayName-count-other": "بيزو أرجنتيني",
  "symbol": "ARS",
  "symbol-alt-narrow": "AR$"
},
"ATS": {
  "displayName": "شلن نمساوي",
  "symbol": "ATS"
},
"AUD": {
  "displayName": "دولار أسترالي",
  "displayName-count-zero": "دولار أسترالي",
  "displayName-count-one": "دولار أسترالي",
  "displayName-count-two": "دولار أسترالي",
  "displayName-count-few": "دولار أسترالي",
  "displayName-count-many": "دولار أسترالي",
  "displayName-count-other": "دولار أسترالي",
  "symbol": "AU$",
  "symbol-alt-narrow": "AU$"
},
"AWG": {
  "displayName": "فلورن أروبي",
  "displayName-count-zero": "فلورن أروبي",
  "displayName-count-one": "فلورن أروبي",
  "displayName-count-two": "فلورن أروبي",
  "displayName-count-few": "فلورن أروبي",
  "displayName-count-many": "فلورن أروبي",
  "displayName-count-other": "فلورن أروبي",
  "symbol": "AWG"
},
"AZM": {
  "displayName": "مانات أذربيجاني",
  "symbol": "AZM"
},
"AZN": {
  "displayName": "مانات أذربيجان",
  "displayName-count-zero": "مانت أذربيجاني",
  "displayName-count-one": "مانت أذربيجاني",
  "displayName-count-two": "مانت أذربيجاني",

```

```

    "displayName-count-few": "مانت أذربيجاني",
    "displayName-count-many": "مانت أذربيجاني",
    "displayName-count-other": "مانت أذربيجاني",
    "symbol": "AZN"
  },
  "BAD": {
    "displayName": "دينار البوسنة والهرسك",
    "symbol": "BAD"
  },
  "BAM": {
    "displayName": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-zero": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-one": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-two": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-few": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-many": "مارك البوسنة والهرسك قابل للتحويل",
    "displayName-count-other": "مارك البوسنة والهرسك قابل للتحويل",
    "symbol": "BAM",
    "symbol-alt-narrow": "KM"
  },
  "BAN": {
    "displayName": "BAN",
    "symbol": "BAN"
  },
  "BBD": {
    "displayName": "دولار بربادوسي",
    "displayName-count-zero": "دولار بربادوسي",
    "displayName-count-one": "دولار بربادوسي",
    "displayName-count-two": "دولار بربادوسي",
    "displayName-count-few": "دولار بربادوسي",
    "displayName-count-many": "دولار بربادوسي",
    "displayName-count-other": "دولار بربادوسي",
    "symbol": "BBD",
    "symbol-alt-narrow": "BB$"
  },
  "BDT": {
    "displayName": "تাকা بنغلاديشي",
    "displayName-count-zero": "تাকা بنغلاديشي",
    "displayName-count-one": "تাকা بنغلاديشي",
    "displayName-count-two": "تাকা بنغلاديشي",
    "displayName-count-few": "تাকা بنغلاديشي",
    "displayName-count-many": "تাকা بنغلاديشي",
    "displayName-count-other": "تাকা بنغلاديشي",
    "symbol": "BDT",
    "symbol-alt-narrow": "ট"
  },
  "BEC": {
    "displayName": "فرنك بلجيكي قابل للتحويل",
    "symbol": "BEC"
  },
  "BEF": {
    "displayName": "فرنك بلجيكي",
    "symbol": "BEF"
  },
  "BEL": {
    "displayName": "فرنك بلجيكي مالي",

```

```

    "symbol": "BEL"
  },
  "BGL": {
    "displayName": "BGL",
    "symbol": "BGL"
  },
  "BGM": {
    "displayName": "BGM",
    "symbol": "BGM"
  },
  "BGN": {
    "displayName": "ليف بلغاري",
    "displayName-count-zero": "ليف بلغاري",
    "displayName-count-one": "ليف بلغاري",
    "displayName-count-two": "ليف بلغاري",
    "displayName-count-few": "ليف بلغاري",
    "displayName-count-many": "ليف بلغاري",
    "displayName-count-other": "ليف بلغاري",
    "symbol": "BGN"
  },
  "BGO": {
    "displayName": "BGO",
    "symbol": "BGO"
  },
  "BHD": {
    "displayName": "دينار بحريني",
    "displayName-count-zero": "دينار بحريني",
    "displayName-count-one": "دينار بحريني",
    "displayName-count-two": "دينار بحريني",
    "displayName-count-few": "دينار بحريني",
    "displayName-count-many": "دينار بحريني",
    "displayName-count-other": "دينار بحريني",
    "symbol": "د.ب."
  },
  "BIF": {
    "displayName": "فرنك بروندي",
    "displayName-count-zero": "فرنك بروندي",
    "displayName-count-one": "فرنك بروندي",
    "displayName-count-two": "فرنك بروندي",
    "displayName-count-few": "فرنك بروندي",
    "displayName-count-many": "فرنك بروندي",
    "displayName-count-other": "فرنك بروندي",
    "symbol": "BIF"
  },
  "BMD": {
    "displayName": "دولار برمودي",
    "displayName-count-zero": "دولار برمودي",
    "displayName-count-one": "دولار برمودي",
    "displayName-count-two": "دولار برمودي",
    "displayName-count-few": "دولار برمودي",
    "displayName-count-many": "دولار برمودي",
    "displayName-count-other": "دولار برمودي",
    "symbol": "BMD",
    "symbol-alt-narrow": "BM$"
  },
  "BND": {
    "displayName": "دولار بروناي",

```

```

    "displayName-count-zero": "دولار برونای",
    "displayName-count-one": "دولار برونای",
    "displayName-count-two": "دولار برونای",
    "displayName-count-few": "دولار برونای",
    "displayName-count-many": "دولار برونای",
    "displayName-count-other": "دولار برونای",
    "symbol": "BND",
    "symbol-alt-narrow": "BN$"
  },
  "BOB": {
    "displayName": "بولیفيانو بولیفي",
    "displayName-count-zero": "بولیفيانو بولیفي",
    "displayName-count-one": "بولیفيانو بولیفي",
    "displayName-count-two": "بولیفيانو بولیفي",
    "displayName-count-few": "بولیفيانو بولیفي",
    "displayName-count-many": "بولیفيانو بولیفي",
    "displayName-count-other": "بولیفيانو بولیفي",
    "symbol": "BOB",
    "symbol-alt-narrow": "Bs"
  },
  "BOL": {
    "displayName": "BOL",
    "symbol": "BOL"
  },
  "BOP": {
    "displayName": "بیزو بولیفي",
    "symbol": "BOP"
  },
  "BOV": {
    "displayName": "مفدول بولیفي",
    "symbol": "BOV"
  },
  "BRB": {
    "displayName": "نوفو کروزایرو برازیلی - 1986-1967",
    "symbol": "BRB"
  },
  "BRC": {
    "displayName": "کروزادو برازیلی",
    "symbol": "BRC"
  },
  "BRE": {
    "displayName": "کروزایرو برازیلی - 1993-1990",
    "symbol": "BRE"
  },
  "BRL": {
    "displayName": "ریال برازیلی",
    "displayName-count-zero": "ریال برازیلی",
    "displayName-count-one": "ریال برازیلی",
    "displayName-count-two": "ریال برازیلی",
    "displayName-count-few": "ریال برازیلی",
    "displayName-count-many": "ریال برازیلی",
    "displayName-count-other": "ریال برازیلی",
    "symbol": "R$",
    "symbol-alt-narrow": "R$"
  },
  "BRN": {
    "displayName": "BRN",

```

```

    "symbol": "BRN"
  },
  "BRR": {
    "displayName": "BRR",
    "symbol": "BRR"
  },
  "BRZ": {
    "displayName": "BRZ",
    "symbol": "BRZ"
  },
  "BSD": {
    "displayName": "دولار باهامي",
    "displayName-count-zero": "دولار باهامي",
    "displayName-count-one": "دولار باهامي",
    "displayName-count-two": "دولار باهامي",
    "displayName-count-few": "دولار باهامي",
    "displayName-count-many": "دولار باهامي",
    "displayName-count-other": "دولار باهامي",
    "symbol": "BSD",
    "symbol-alt-narrow": "BS$"
  },
  "BTN": {
    "displayName": "نولتوم بوتاني",
    "displayName-count-zero": "نولتوم بوتاني",
    "displayName-count-one": "نولتوم بوتاني",
    "displayName-count-two": "نولتوم بوتاني",
    "displayName-count-few": "نولتوم بوتاني",
    "displayName-count-many": "نولتوم بوتاني",
    "displayName-count-other": "نولتوم بوتاني",
    "symbol": "BTN"
  },
  "BUK": {
    "displayName": "كيات بورمي",
    "symbol": "BUK"
  },
  "BWP": {
    "displayName": "بولا بتسواني",
    "displayName-count-zero": "بولا بتسواني",
    "displayName-count-one": "بولا بتسواني",
    "displayName-count-two": "بولا بتسواني",
    "displayName-count-few": "بولا بتسواني",
    "displayName-count-many": "بولا بتسواني",
    "displayName-count-other": "بولا بتسواني",
    "symbol": "BWP",
    "symbol-alt-narrow": "P"
  },
  "BYB": {
    "displayName": "روبل بيلاروسي جديد - 1999-1994",
    "symbol": "BYB"
  },
  "BYN": {
    "displayName": "روبل بيلاروسي",
    "displayName-count-zero": "روبل بيلاروسي",
    "displayName-count-one": "روبل بيلاروسي",
    "displayName-count-two": "روبل بيلاروسي",
    "displayName-count-few": "روبل بيلاروسي",
    "displayName-count-many": "روبل بيلاروسي",

```

```

    "displayName-count-other": "روبل بيلاروسي",
    "symbol": "BYN",
    "symbol-alt-narrow": "p.",
  },
  "BYR": {
    "displayName": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-zero": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-one": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-two": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-few": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-many": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "displayName-count-other": ") ۲۰۱۶-۲۰۰۰ ( روبل بيلاروسي",
    "symbol": "BYR"
  },
  "BZD": {
    "displayName": "دولار بليزي",
    "displayName-count-zero": "دولار بليزي",
    "displayName-count-one": "دولار بليزي",
    "displayName-count-two": "دولاران بليزيان",
    "displayName-count-few": "دولار بليزي",
    "displayName-count-many": "دولار بليزي",
    "displayName-count-other": "دولار بليزي",
    "symbol": "BZD",
    "symbol-alt-narrow": "BZ$"
  },
  "CAD": {
    "displayName": "دولار كندي",
    "displayName-count-zero": "دولار كندي",
    "displayName-count-one": "دولار كندي",
    "displayName-count-two": "دولار كندي",
    "displayName-count-few": "دولار كندي",
    "displayName-count-many": "دولار كندي",
    "displayName-count-other": "دولار كندي",
    "symbol": "CA$",
    "symbol-alt-narrow": "CA$"
  },
  "CDF": {
    "displayName": "فرنك كونغولي",
    "displayName-count-zero": "فرنك كونغولي",
    "displayName-count-one": "فرنك كونغولي",
    "displayName-count-two": "فرنك كونغولي",
    "displayName-count-few": "فرنك كونغولي",
    "displayName-count-many": "فرنك كونغولي",
    "displayName-count-other": "فرنك كونغولي",
    "symbol": "CDF"
  },
  "CHE": {
    "displayName": "CHE",
    "symbol": "CHE"
  },
  "CHF": {
    "displayName": "فرنك سويسري",
    "displayName-count-zero": "فرنك سويسري",
    "displayName-count-one": "فرنك سويسري",
    "displayName-count-two": "فرنك سويسري",
    "displayName-count-few": "فرنك سويسري",
    "displayName-count-many": "فرنك سويسري",

```

```

        "displayName-count-other": "فرنك سويسري",
        "symbol": "CHF"
    },
    "CHW": {
        "displayName": "CHW",
        "symbol": "CHW"
    },
    "CLE": {
        "displayName": "CLE",
        "symbol": "CLE"
    },
    "CLF": {
        "displayName": "CLF",
        "symbol": "CLF"
    },
    "CLP": {
        "displayName": "بيزو تشيلي",
        "displayName-count-zero": "بيزو تشيلي",
        "displayName-count-one": "بيزو تشيلي",
        "displayName-count-two": "بيزو تشيلي",
        "displayName-count-few": "بيزو تشيلي",
        "displayName-count-many": "بيزو تشيلي",
        "displayName-count-other": "بيزو تشيلي",
        "symbol": "CLP",
        "symbol-alt-narrow": "CL$"
    },
    "CNH": {
        "displayName": "يوان صيني (في الخارج)",
        "displayName-count-zero": "يوان صيني (في الخارج)",
        "displayName-count-one": "يوان صيني (في الخارج)",
        "displayName-count-two": "يوان صيني (في الخارج)",
        "displayName-count-few": "يوان صيني (في الخارج)",
        "displayName-count-many": "يوان صيني (في الخارج)",
        "displayName-count-other": "يوان صيني (في الخارج)",
        "symbol": "CNH"
    },
    "CNX": {
        "displayName": "CNX",
        "symbol": "CNX"
    },
    "CNY": {
        "displayName": "يوان صيني",
        "displayName-count-zero": "يوان صيني",
        "displayName-count-one": "يوان صيني",
        "displayName-count-two": "يوان صيني",
        "displayName-count-few": "يوان صيني",
        "displayName-count-many": "يوان صيني",
        "displayName-count-other": "يوان صيني",
        "symbol": "CN¥",
        "symbol-alt-narrow": "CN¥"
    },
    "COP": {
        "displayName": "بيزو كولومبي",
        "displayName-count-zero": "بيزو كولومبي",
        "displayName-count-one": "بيزو كولومبي",
        "displayName-count-two": "بيزو كولومبي",
        "displayName-count-few": "بيزو كولومبي",

```

```

        "displayName-count-many": "بيزو كولومبي",
        "displayName-count-other": "بيزو كولومبي",
        "symbol": "COP",
        "symbol-alt-narrow": "CO$"
    },
    "COU": {
        "displayName": "COU",
        "symbol": "COU"
    },
    "CRC": {
        "displayName": "كولن كوستاريكي",
        "displayName-count-zero": "كولن كوستاريكي",
        "displayName-count-one": "كولن كوستاريكي",
        "displayName-count-two": "كولن كوستاريكي",
        "displayName-count-few": "كولن كوستاريكي",
        "displayName-count-many": "كولن كوستاريكي",
        "displayName-count-other": "كولن كوستاريكي",
        "symbol": "CRC",
        "symbol-alt-narrow": "₡"
    },
    "CSD": {
        "displayName": "دينار صربي قديم",
        "symbol": "CSD"
    },
    "CSK": {
        "displayName": "كرونة تشيكوسلوفاكيا",
        "symbol": "CSK"
    },
    "CUC": {
        "displayName": "بيزو كوبي قابل للتحويل",
        "displayName-count-zero": "بيزو كوبي قابل للتحويل",
        "displayName-count-one": "بيزو كوبي قابل للتحويل",
        "displayName-count-two": "بيزو كوبي قابل للتحويل",
        "displayName-count-few": "بيزو كوبي قابل للتحويل",
        "displayName-count-many": "بيزو كوبي قابل للتحويل",
        "displayName-count-other": "بيزو كوبي قابل للتحويل",
        "symbol": "CUC",
        "symbol-alt-narrow": "$"
    },
    "CUP": {
        "displayName": "بيزو كوبي",
        "displayName-count-zero": "بيزو كوبي",
        "displayName-count-one": "بيزو كوبي",
        "displayName-count-two": "بيزو كوبي",
        "displayName-count-few": "بيزو كوبي",
        "displayName-count-many": "بيزو كوبي",
        "displayName-count-other": "بيزو كوبي",
        "symbol": "CUP",
        "symbol-alt-narrow": "CU$"
    },
    "CVE": {
        "displayName": "اسكودو الرأس الخضراء",
        "displayName-count-zero": "اسكودو الرأس الخضراء",
        "displayName-count-one": "اسكودو الرأس الخضراء",
        "displayName-count-two": "اسكودو الرأس الخضراء",
        "displayName-count-few": "اسكودو الرأس الخضراء",
        "displayName-count-many": "اسكودو الرأس الخضراء",

```



```

    "displayName-count-other": "اسكودو الرأس الخضراء",
    "symbol": "CVE"
  },
  "CYP": {
    "displayName": "جنيه قبرصي",
    "symbol": "CYP"
  },
  "CZK": {
    "displayName": "كرونة تشيكية",
    "displayName-count-zero": "كرونة تشيكية",
    "displayName-count-one": "كرونة تشيكية",
    "displayName-count-two": "كرونة تشيكية",
    "displayName-count-few": "كرونة تشيكية",
    "displayName-count-many": "كرونة تشيكية",
    "displayName-count-other": "كرونة تشيكية",
    "symbol": "CZK",
    "symbol-alt-narrow": "Kč"
  },
  "DDM": {
    "displayName": "أوستمارك ألماني شرقي",
    "symbol": "DDM"
  },
  "DEM": {
    "displayName": "مارك ألماني",
    "symbol": "DEM"
  },
  "DJF": {
    "displayName": "فرنك جيبوتي",
    "displayName-count-zero": "فرنك جيبوتي",
    "displayName-count-one": "فرنك جيبوتي",
    "displayName-count-two": "فرنك جيبوتي",
    "displayName-count-few": "فرنك جيبوتي",
    "displayName-count-many": "فرنك جيبوتي",
    "displayName-count-other": "فرنك جيبوتي",
    "symbol": "DJF"
  },
  "DKK": {
    "displayName": "كرونة دنماركية",
    "displayName-count-zero": "كرونة دنماركية",
    "displayName-count-one": "كرونة دنماركية",
    "displayName-count-two": "كرونة دنماركية",
    "displayName-count-few": "كرونة دنماركية",
    "displayName-count-many": "كرونة دنماركية",
    "displayName-count-other": "كرونة دنماركية",
    "symbol": "DKK",
    "symbol-alt-narrow": "kr"
  },
  "DOP": {
    "displayName": "بيزو الدومنيكان",
    "displayName-count-zero": "بيزو الدومنيكان",
    "displayName-count-one": "بيزو الدومنيكان",
    "displayName-count-two": "بيزو الدومنيكان",
    "displayName-count-few": "بيزو الدومنيكان",
    "displayName-count-many": "بيزو الدومنيكان",
    "displayName-count-other": "بيزو الدومنيكان",
    "symbol": "DOP",
    "symbol-alt-narrow": "DO$"
  }

```

```

    },
    "DZD": {
      "displayName": "دينار جزائري",
      "displayName-count-zero": "دينار جزائري",
      "displayName-count-one": "دينار جزائري",
      "displayName-count-two": "ديناران جزائريان",
      "displayName-count-few": "دينارات جزائرية",
      "displayName-count-many": "دينارًا جزائريًا",
      "displayName-count-other": "دينار جزائري",
      "symbol": "د.ج."
    },
    "ECS": {
      "displayName": "ECS",
      "symbol": "ECS"
    },
    "ECV": {
      "displayName": "ECV",
      "symbol": "ECV"
    },
    "EEK": {
      "displayName": "كرونه استونية",
      "symbol": "EEK"
    },
    "EGP": {
      "displayName": "جنيه مصري",
      "displayName-count-zero": "جنيه مصري",
      "displayName-count-one": "جنيه مصري",
      "displayName-count-two": "جنيهان مصريان",
      "displayName-count-few": "جنيهات مصرية",
      "displayName-count-many": "جنيهاً مصريًا",
      "displayName-count-other": "جنيه مصري",
      "symbol": "ج.م.",
      "symbol-alt-narrow": "E£"
    },
    "ERN": {
      "displayName": "ناكفا أريتري",
      "displayName-count-zero": "ناكفا أريتري",
      "displayName-count-one": "ناكفا أريتري",
      "displayName-count-two": "ناكفا أريتري",
      "displayName-count-few": "ناكفا أريتري",
      "displayName-count-many": "ناكفا أريتري",
      "displayName-count-other": "ناكفا أريتري",
      "symbol": "ERN"
    },
    "ESA": {
      "displayName": "ESA",
      "symbol": "ESA"
    },
    "ESB": {
      "displayName": "ESB",
      "symbol": "ESB"
    },
    "ESP": {
      "displayName": "بيزيتا إسباني",
      "symbol": "ESP",
      "symbol-alt-narrow": "₧"
    },
  },

```

```

"ETB": {
  "displayName": "بیر اٹیوبی",
  "displayName-count-zero": "بیر اٹیوبی",
  "displayName-count-one": "بیر اٹیوبی",
  "displayName-count-two": "بیر اٹیوبی",
  "displayName-count-few": "بیر اٹیوبی",
  "displayName-count-many": "بیر اٹیوبی",
  "displayName-count-other": "بیر اٹیوبی",
  "symbol": "ETB"
},
"EUR": {
  "displayName": "یورو",
  "displayName-count-zero": "یورو",
  "displayName-count-one": "یورو",
  "displayName-count-two": "یورو",
  "displayName-count-few": "یورو",
  "displayName-count-many": "یورو",
  "displayName-count-other": "یورو",
  "symbol": "€",
  "symbol-alt-narrow": "€"
},
"FIM": {
  "displayName": "مارکا فنلندی",
  "symbol": "FIM"
},
"FJD": {
  "displayName": "دولار فیجی",
  "displayName-count-zero": "دولار فیجی",
  "displayName-count-one": "دولار فیجی",
  "displayName-count-two": "دولار فیجی",
  "displayName-count-few": "دولار فیجی",
  "displayName-count-many": "دولار فیجی",
  "displayName-count-other": "دولار فیجی",
  "symbol": "FJD",
  "symbol-alt-narrow": "FJ$"
},
"FKP": {
  "displayName": "جنيه جزر فوکلاند",
  "displayName-count-zero": "جنيه جزر فوکلاند",
  "displayName-count-one": "جنيه جزر فوکلاند",
  "displayName-count-two": "جنيه جزر فوکلاند",
  "displayName-count-few": "جنيه جزر فوکلاند",
  "displayName-count-many": "جنيه جزر فوکلاند",
  "displayName-count-other": "جنيه جزر فوکلاند",
  "symbol": "FKP",
  "symbol-alt-narrow": "£"
},
"FRF": {
  "displayName": "فرنك فرنسي",
  "symbol": "FRF"
},
"GBP": {
  "displayName": "جنيه إسترليني",
  "displayName-count-zero": "جنيه إسترليني",
  "displayName-count-one": "جنيه إسترليني",
  "displayName-count-two": "جنيه إسترليني",
  "displayName-count-few": "جنيه إسترليني",

```

```

    "displayName-count-many": "جنيه إسترليني",
    "displayName-count-other": "جنيه إسترليني",
    "symbol": "£",
    "symbol-alt-narrow": "UK£"
  },
  "GEK": {
    "displayName": "GEK",
    "symbol": "GEK"
  },
  "GEL": {
    "displayName": "ლარი جورجى",
    "displayName-count-zero": "لاري جورجى",
    "displayName-count-one": "لاري جورجى",
    "displayName-count-two": "لاري جورجى",
    "displayName-count-few": "لاري جورجى",
    "displayName-count-many": "لاري جورجى",
    "displayName-count-other": "لاري جورجى",
    "symbol": "GEL",
    "symbol-alt-narrow": "ლ",
    "symbol-alt-variant": "ლ"
  },
  "GHC": {
    "displayName": "سيدي غانى",
    "symbol": "GHC"
  },
  "GHS": {
    "displayName": "سيدي غانا",
    "displayName-count-zero": "سيدي غانا",
    "displayName-count-one": "سيدي غانا",
    "displayName-count-two": "سيدي غانا",
    "displayName-count-few": "سيدي غانا",
    "displayName-count-many": "سيدي غانا",
    "displayName-count-other": "سيدي غانا",
    "symbol": "GHS"
  },
  "GIP": {
    "displayName": "جنيه جبل طارق",
    "displayName-count-zero": "جنيه جبل طارق",
    "displayName-count-one": "جنيه جبل طارق",
    "displayName-count-two": "جنيه جبل طارق",
    "displayName-count-few": "جنيه جبل طارق",
    "displayName-count-many": "جنيه جبل طارق",
    "displayName-count-other": "جنيه جبل طارق",
    "symbol": "GIP",
    "symbol-alt-narrow": "£"
  },
  "GMD": {
    "displayName": "دلاسي غامبي",
    "displayName-count-zero": "دلاسي غامبي",
    "displayName-count-one": "دلاسي غامبي",
    "displayName-count-two": "دلاسي غامبي",
    "displayName-count-few": "دلاسي غامبي",
    "displayName-count-many": "دلاسي غامبي",
    "displayName-count-other": "دلاسي غامبي",
    "symbol": "GMD"
  },
  "GNF": {

```

```

    "displayName": "فرنك غينيا",
    "displayName-count-zero": "فرنك غينيا",
    "displayName-count-one": "فرنك غينيا",
    "displayName-count-two": "فرنك غينيا",
    "displayName-count-few": "فرنك غينيا",
    "displayName-count-many": "فرنك غينيا",
    "displayName-count-other": "فرنك غينيا",
    "symbol": "GNF",
    "symbol-alt-narrow": "FG"
  },
  "GNS": {
    "displayName": "سيللي غينيا",
    "symbol": "GNS"
  },
  "GQE": {
    "displayName": "اكويل جونيينا غينيا الاستوائية",
    "symbol": "GQE"
  },
  "GRD": {
    "displayName": "دراخما يوناني",
    "symbol": "GRD"
  },
  "GTQ": {
    "displayName": "كوتزال غواتيمالا",
    "displayName-count-zero": "كوتزال غواتيمالا",
    "displayName-count-one": "كوتزال غواتيمالا",
    "displayName-count-two": "كوتزال غواتيمالا",
    "displayName-count-few": "كوتزال غواتيمالا",
    "displayName-count-many": "كوتزال غواتيمالا",
    "displayName-count-other": "كوتزال غواتيمالا",
    "symbol": "GTQ",
    "symbol-alt-narrow": "Q"
  },
  "GWE": {
    "displayName": "اسكود برتغالي غينيا",
    "symbol": "GWE"
  },
  "GWP": {
    "displayName": "بيزو غينيا بيساو",
    "symbol": "GWP"
  },
  "GYD": {
    "displayName": "دولار غيانا",
    "displayName-count-zero": "دولار غيانا",
    "displayName-count-one": "دولار غيانا",
    "displayName-count-two": "دولار غيانا",
    "displayName-count-few": "دولار غيانا",
    "displayName-count-many": "دولار غيانا",
    "displayName-count-other": "دولار غيانا",
    "symbol": "GYD",
    "symbol-alt-narrow": "GY$"
  },
  "HKD": {
    "displayName": "دولار هونغ كونغ",
    "displayName-count-zero": "دولار هونغ كونغ",
    "displayName-count-one": "دولار هونغ كونغ",
    "displayName-count-two": "دولار هونغ كونغ",

```

```

    "displayName-count-few": "دولار هونغ كونغ",
    "displayName-count-many": "دولار هونغ كونغ",
    "displayName-count-other": "دولار هونغ كونغ",
    "symbol": "HK$",
    "symbol-alt-narrow": "HK$"
  },
  "HNL": {
    "displayName": "ليمبيرا هندوراس",
    "displayName-count-zero": "ليمبيرا هندوراس",
    "displayName-count-one": "ليمبيرا هندوراس",
    "displayName-count-two": "ليمبيرا هندوراس",
    "displayName-count-few": "ليمبيرا هندوراس",
    "displayName-count-many": "ليمبيرا هندوراس",
    "displayName-count-other": "ليمبيرا هندوراس",
    "symbol": "HNL",
    "symbol-alt-narrow": "L"
  },
  "HRD": {
    "displayName": "دينار كرواتي",
    "symbol": "HRD"
  },
  "HRK": {
    "displayName": "كونا كرواتي",
    "displayName-count-zero": "كونا كرواتي",
    "displayName-count-one": "كونا كرواتي",
    "displayName-count-two": "كونا كرواتي",
    "displayName-count-few": "كونا كرواتي",
    "displayName-count-many": "كونا كرواتي",
    "displayName-count-other": "كونا كرواتي",
    "symbol": "HRK",
    "symbol-alt-narrow": "kn"
  },
  "HTG": {
    "displayName": "جوردي هايتي",
    "displayName-count-zero": "جوردي هايتي",
    "displayName-count-one": "جوردي هايتي",
    "displayName-count-two": "جوردي هايتي",
    "displayName-count-few": "جوردي هايتي",
    "displayName-count-many": "جوردي هايتي",
    "displayName-count-other": "جوردي هايتي",
    "symbol": "HTG"
  },
  "HUF": {
    "displayName": "فورينت هنغاري",
    "displayName-count-zero": "فورينت هنغاري",
    "displayName-count-one": "فورينت هنغاري",
    "displayName-count-two": "فورينت هنغاري",
    "displayName-count-few": "فورينت هنغاري",
    "displayName-count-many": "فورينت هنغاري",
    "displayName-count-other": "فورينت هنغاري",
    "symbol": "HUF",
    "symbol-alt-narrow": "Ft"
  },
  "IDR": {
    "displayName": "روبية إندونيسية",
    "displayName-count-zero": "روبية إندونيسية",
    "displayName-count-one": "روبية إندونيسية",

```

```

    "displayName-count-two": "روبية إندونيسية",
    "displayName-count-few": "روبية إندونيسية",
    "displayName-count-many": "روبية إندونيسية",
    "displayName-count-other": "روبية إندونيسية",
    "symbol": "IDR",
    "symbol-alt-narrow": "Rp"
  },
  "IEP": {
    "displayName": "جنيه إيرلندي",
    "symbol": "IEP"
  },
  "ILP": {
    "displayName": "جنيه إسرائيلي",
    "symbol": "ILP"
  },
  "ILR": {
    "displayName": "ILR",
    "symbol": "ILR"
  },
  "ILS": {
    "displayName": "شيكل إسرائيلي جديد",
    "displayName-count-zero": "شيكل إسرائيلي جديد",
    "displayName-count-one": "شيكل إسرائيلي جديد",
    "displayName-count-two": "شيكل إسرائيلي جديد",
    "displayName-count-few": "شيكل إسرائيلي جديد",
    "displayName-count-many": "شيكل إسرائيلي جديد",
    "displayName-count-other": "شيكل إسرائيلي جديد",
    "symbol": "₪",
    "symbol-alt-narrow": "₪"
  },
  "INR": {
    "displayName": "روبية هندي",
    "displayName-count-zero": "روبية هندي",
    "displayName-count-one": "روبية هندي",
    "displayName-count-two": "روبية هندي",
    "displayName-count-few": "روبية هندي",
    "displayName-count-many": "روبية هندي",
    "displayName-count-other": "روبية هندي",
    "symbol": "₹",
    "symbol-alt-narrow": "₹"
  },
  "IQD": {
    "displayName": "دينار عراقي",
    "displayName-count-zero": "دينار عراقي",
    "displayName-count-one": "دينار عراقي",
    "displayName-count-two": "دينار عراقي",
    "displayName-count-few": "دينار عراقي",
    "displayName-count-many": "دينار عراقي",
    "displayName-count-other": "دينار عراقي",
    "symbol": "د.ع."
  },
  "IRR": {
    "displayName": "ريال إيراني",
    "displayName-count-zero": "ريال إيراني",
    "displayName-count-one": "ريال إيراني",
    "displayName-count-two": "ريال إيراني",
    "displayName-count-few": "ريال إيراني",

```

```

    "displayName-count-many": "هلا إيرانى",
    "displayName-count-other": "هلا إيرانى",
    "symbol": "₪.",
  },
  "ISJ": {
    "displayName": "ISJ",
    "symbol": "ISJ"
  },
  "ISK": {
    "displayName": "كرونة آيسلندية",
    "displayName-count-zero": "كرونة آيسلندية",
    "displayName-count-one": "كرونة آيسلندية",
    "displayName-count-two": "كرونة آيسلندية",
    "displayName-count-few": "كرونة آيسلندية",
    "displayName-count-many": "كرونة آيسلندية",
    "displayName-count-other": "كرونة آيسلندية",
    "symbol": "ISK",
    "symbol-alt-narrow": "kr"
  },
  "ITL": {
    "displayName": "ليرة إيطالية",
    "symbol": "ITL"
  },
  "JMD": {
    "displayName": "دولار جامايكى",
    "displayName-count-zero": "دولار جامايكى",
    "displayName-count-one": "دولار جامايكى",
    "displayName-count-two": "دولار جامايكى",
    "displayName-count-few": "دولار جامايكى",
    "displayName-count-many": "دولار جامايكى",
    "displayName-count-other": "دولار جامايكى",
    "symbol": "JMD",
    "symbol-alt-narrow": "JM$"
  },
  "JOD": {
    "displayName": "دينار أردنى",
    "displayName-count-zero": "دينار أردنى",
    "displayName-count-one": "دينار أردنى",
    "displayName-count-two": "دينار أردنى",
    "displayName-count-few": "دينار أردنى",
    "displayName-count-many": "دينار أردنى",
    "displayName-count-other": "دينار أردنى",
    "symbol": "د.أ."
  },
  "JPY": {
    "displayName": "ين يابانى",
    "displayName-count-zero": "ين يابانى",
    "displayName-count-one": "ين يابانى",
    "displayName-count-two": "ين يابانى",
    "displayName-count-few": "ين يابانى",
    "displayName-count-many": "ين يابانى",
    "displayName-count-other": "ين يابانى",
    "symbol": "¥",
    "symbol-alt-narrow": "¥"
  },
  "KES": {
    "displayName": "شلل كينىي",

```



```

    "displayName-count-zero": "شلن کینیی",
    "displayName-count-one": "شلن کینیی",
    "displayName-count-two": "شلن کینیی",
    "displayName-count-few": "شلن کینیی",
    "displayName-count-many": "شلن کینیی",
    "displayName-count-other": "شلن کینیی",
    "symbol": "KES"
  },
  "KGS": {
    "displayName": "سوم قیرغستانی",
    "displayName-count-zero": "سوم قیرغستانی",
    "displayName-count-one": "سوم قیرغستانی",
    "displayName-count-two": "سوم قیرغستانی",
    "displayName-count-few": "سوم قیرغستانی",
    "displayName-count-many": "سوم قیرغستانی",
    "displayName-count-other": "سوم قیرغستانی",
    "symbol": "KGS"
  },
  "KHR": {
    "displayName": "ریال کمبودی",
    "displayName-count-zero": "ریال کمبودی",
    "displayName-count-one": "ریال کمبودی",
    "displayName-count-two": "ریال کمبودی",
    "displayName-count-few": "ریال کمبودی",
    "displayName-count-many": "ریال کمبودی",
    "displayName-count-other": "ریال کمبودی",
    "symbol": "KHR",
    "symbol-alt-narrow": "៛"
  },
  "KMF": {
    "displayName": "فرنك جزر القمر",
    "displayName-count-zero": "فرنك جزر القمر",
    "displayName-count-one": "فرنك جزر القمر",
    "displayName-count-two": "فرنك جزر القمر",
    "displayName-count-few": "فرنك جزر القمر",
    "displayName-count-many": "فرنك جزر القمر",
    "displayName-count-other": "فرنك جزر القمر",
    "symbol": "KMF",
    "symbol-alt-narrow": "CF"
  },
  "KPW": {
    "displayName": "وون كوريا الشمالية",
    "displayName-count-zero": "وون كوريا الشمالية",
    "displayName-count-one": "وون كوريا الشمالية",
    "displayName-count-two": "وون كوريا الشمالية",
    "displayName-count-few": "وون كوريا الشمالية",
    "displayName-count-many": "وون كوريا الشمالية",
    "displayName-count-other": "وون كوريا الشمالية",
    "symbol": "KPW",
    "symbol-alt-narrow": "₩"
  },
  "KRH": {
    "displayName": "KRH",
    "symbol": "KRH"
  },
  "KRO": {

```

```

    "displayName": "KRO",
    "symbol": "KRO"
  },
  "KRW": {
    "displayName": "وون كوريا الجنوبية",
    "displayName-count-zero": "وون كوريا الجنوبية",
    "displayName-count-one": "وون كوريا الجنوبية",
    "displayName-count-two": "وون كوريا الجنوبية",
    "displayName-count-few": "وون كوريا الجنوبية",
    "displayName-count-many": "وون كوريا الجنوبية",
    "displayName-count-other": "وون كوريا الجنوبية",
    "symbol": "₩",
    "symbol-alt-narrow": "₩"
  },
  "KWD": {
    "displayName": "دينار كويتي",
    "displayName-count-zero": "دينار كويتي",
    "displayName-count-one": "دينار كويتي",
    "displayName-count-two": "دينار كويتي",
    "displayName-count-few": "دينار كويتي",
    "displayName-count-many": "دينار كويتي",
    "displayName-count-other": "دينار كويتي",
    "symbol": "د.ك."
  },
  "KYD": {
    "displayName": "دولار جزر كيمن",
    "displayName-count-zero": "دولار جزر كيمن",
    "displayName-count-one": "دولار جزر كيمن",
    "displayName-count-two": "دولار جزر كيمن",
    "displayName-count-few": "دولار جزر كيمن",
    "displayName-count-many": "دولار جزر كيمن",
    "displayName-count-other": "دولار جزر كيمن",
    "symbol": "KYD",
    "symbol-alt-narrow": "KY$"
  },
  "KZT": {
    "displayName": "تينغ كازاخستاني",
    "displayName-count-zero": "تينغ كازاخستاني",
    "displayName-count-one": "تينغ كازاخستاني",
    "displayName-count-two": "تينغ كازاخستاني",
    "displayName-count-few": "تينغ كازاخستاني",
    "displayName-count-many": "تينغ كازاخستاني",
    "displayName-count-other": "تينغ كازاخستاني",
    "symbol": "KZT",
    "symbol-alt-narrow": "₸"
  },
  "LAK": {
    "displayName": "كيب لاوسي",
    "displayName-count-zero": "كيب لاوسي",
    "displayName-count-one": "كيب لاوسي",
    "displayName-count-two": "كيب لاوسي",
    "displayName-count-few": "كيب لاوسي",
    "displayName-count-many": "كيب لاوسي",
    "displayName-count-other": "كيب لاوسي",
    "symbol": "LAK",
    "symbol-alt-narrow": "₭"
  },
}

```

```

"LBP": {
  "displayName": "جنيه لبناني",
  "displayName-count-zero": "جنيه لبناني",
  "displayName-count-one": "جنيه لبناني",
  "displayName-count-two": "جنيه لبناني",
  "displayName-count-few": "جنيه لبناني",
  "displayName-count-many": "جنيه لبناني",
  "displayName-count-other": "جنيه لبناني",
  "symbol": ".ل.ل",
  "symbol-alt-narrow": "Lℓ"
},
"LKR": {
  "displayName": "روبية سريلانكية",
  "displayName-count-zero": "روبية سريلانكية",
  "displayName-count-one": "روبية سريلانكية",
  "displayName-count-two": "روبية سريلانكية",
  "displayName-count-few": "روبية سريلانكية",
  "displayName-count-many": "روبية سريلانكية",
  "displayName-count-other": "روبية سريلانكية",
  "symbol": "LKR",
  "symbol-alt-narrow": "Rs"
},
"LRD": {
  "displayName": "دولار ليبيري",
  "displayName-count-zero": "دولار ليبيري",
  "displayName-count-one": "دولار ليبيري",
  "displayName-count-two": "دولاران ليبيريان",
  "displayName-count-few": "دولارات ليبيرية",
  "displayName-count-many": "دولارًا ليبيريًا",
  "displayName-count-other": "دولار ليبيري",
  "symbol": "LRD",
  "symbol-alt-narrow": "$"
},
"LSL": {
  "displayName": "لوتي ليسوتو",
  "symbol": "LSL"
},
"LTL": {
  "displayName": "ليتة ليتوانية",
  "displayName-count-zero": "ليتة ليتوانية",
  "displayName-count-one": "ليتة ليتوانية",
  "displayName-count-two": "ليتة ليتوانية",
  "displayName-count-few": "ليتة ليتوانية",
  "displayName-count-many": "ليتة ليتوانية",
  "displayName-count-other": "ليتة ليتوانية",
  "symbol": "LTL",
  "symbol-alt-narrow": "Lt"
},
"LTT": {
  "displayName": "تالوناس ليتواني",
  "symbol": "LTT"
},
"LUC": {
  "displayName": "فرنك لوكسمبرج قابل للتحويل",
  "symbol": "LUC"
},
"LUF": {

```

```

    "displayName": "فرنك لوكسمبرج",
    "symbol": "LUF"
  },
  "LUL": {
    "displayName": "فرنك لوكسمبرج المالي",
    "symbol": "LUL"
  },
  "LVL": {
    "displayName": "لاتس لاتفيا",
    "displayName-count-zero": "لاتس لاتفي",
    "displayName-count-one": "لاتس لاتفي",
    "displayName-count-two": "لاتس لاتفي",
    "displayName-count-few": "لاتس لاتفي",
    "displayName-count-many": "لاتس لاتفي",
    "displayName-count-other": "لاتس لاتفي",
    "symbol": "LVL",
    "symbol-alt-narrow": "Ls"
  },
  "LVR": {
    "displayName": "روبل لاتفيا",
    "symbol": "LVR"
  },
  "LYD": {
    "displayName": "دينار ليبي",
    "displayName-count-zero": "دينار ليبي",
    "displayName-count-one": "دينار ليبي",
    "displayName-count-two": "ديناران ليبيان",
    "displayName-count-few": "دينارات ليبية",
    "displayName-count-many": "دينارًا ليبيا",
    "displayName-count-other": "دينار ليبي",
    "symbol": "د.ل."
  },
  "MAD": {
    "displayName": "درهم مغربي",
    "displayName-count-zero": "درهم مغربي",
    "displayName-count-one": "درهم مغربي",
    "displayName-count-two": "درهمان مغربيان",
    "displayName-count-few": "دراهم مغربية",
    "displayName-count-many": "درهما مغربيا",
    "displayName-count-other": "درهم مغربي",
    "symbol": "د.م."
  },
  "MAF": {
    "displayName": "فرنك مغربي",
    "symbol": "MAF"
  },
  "MCF": {
    "displayName": "MCF",
    "symbol": "MCF"
  },
  "MDC": {
    "displayName": "MDC",
    "symbol": "MDC"
  },
  "MDL": {
    "displayName": "ليو مولدوفي",
    "displayName-count-zero": "ليو مولدوفي",

```

```

    "displayName-count-one": "ليو مولدوفي",
    "displayName-count-two": "ليو مولدوفي",
    "displayName-count-few": "ليو مولدوفي",
    "displayName-count-many": "ليو مولدوفي",
    "displayName-count-other": "ليو مولدوفي",
    "symbol": "MDL"
  },
  "MGA": {
    "displayName": "أرياري مدغشقر",
    "displayName-count-zero": "أرياري مدغشقر",
    "displayName-count-one": "أرياري مدغشقر",
    "displayName-count-two": "أرياري مدغشقر",
    "displayName-count-few": "أرياري مدغشقر",
    "displayName-count-many": "أرياري مدغشقر",
    "displayName-count-other": "أرياري مدغشقر",
    "symbol": "MGA",
    "symbol-alt-narrow": "Ar"
  },
  "MGF": {
    "displayName": "فرنك مدغشقر",
    "symbol": "MGF"
  },
  "MKD": {
    "displayName": "دينار مقدوني",
    "displayName-count-zero": "دينار مقدوني",
    "displayName-count-one": "دينار مقدوني",
    "displayName-count-two": "ديناران مقدونيان",
    "displayName-count-few": "دينارات مقدونية",
    "displayName-count-many": "دينارًا مقدونيًا",
    "displayName-count-other": "دينار مقدوني",
    "symbol": "MKD"
  },
  "MKN": {
    "displayName": "MKN",
    "symbol": "MKN"
  },
  "MLF": {
    "displayName": "فرنك مالي",
    "symbol": "MLF"
  },
  "MMK": {
    "displayName": "كيات ميانمار",
    "displayName-count-zero": "كيات ميانمار",
    "displayName-count-one": "كيات ميانمار",
    "displayName-count-two": "كيات ميانمار",
    "displayName-count-few": "كيات ميانمار",
    "displayName-count-many": "كيات ميانمار",
    "displayName-count-other": "كيات ميانمار",
    "symbol": "MMK",
    "symbol-alt-narrow": "K"
  },
  "MNT": {
    "displayName": "توغروغ منغولي",
    "displayName-count-zero": "توغروغ منغولي",
    "displayName-count-one": "توغروغ منغولي",
    "displayName-count-two": "توغروغ منغولي",
    "displayName-count-few": "توغروغ منغولي",

```

```

    "displayName-count-many": "توغروغ منغولي",
    "displayName-count-other": "توغروغ منغولي",
    "symbol": "MNT",
    "symbol-alt-narrow": "₮"
  },
  "MOP": {
    "displayName": "باتاكا ماكاوي",
    "displayName-count-zero": "باتاكا ماكاوي",
    "displayName-count-one": "باتاكا ماكاوي",
    "displayName-count-two": "باتاكا ماكاوي",
    "displayName-count-few": "باتاكا ماكاوي",
    "displayName-count-many": "باتاكا ماكاوي",
    "displayName-count-other": "باتاكا ماكاوي",
    "symbol": "MOP"
  },
  "MRO": {
    "displayName": "أوقية موريتانية",
    "displayName-count-zero": "أوقية موريتانية",
    "displayName-count-one": "أوقية موريتانية",
    "displayName-count-two": "أوقية موريتانية",
    "displayName-count-few": "أوقية موريتانية",
    "displayName-count-many": "أوقية موريتانية",
    "displayName-count-other": "أوقية موريتانية",
    "symbol": "أ.م."
  },
  "MTL": {
    "displayName": "ليرة مالطية",
    "symbol": "MTL"
  },
  "MTP": {
    "displayName": "جنيه مالطي",
    "symbol": "MTP"
  },
  "MUR": {
    "displayName": "روبية موريشيوسية",
    "displayName-count-zero": "روبية موريشيوسية",
    "displayName-count-one": "روبية موريشيوسية",
    "displayName-count-two": "روبية موريشيوسية",
    "displayName-count-few": "روبية موريشيوسية",
    "displayName-count-many": "روبية موريشيوسية",
    "displayName-count-other": "روبية موريشيوسية",
    "symbol": "MUR",
    "symbol-alt-narrow": "Rs"
  },
  "MVP": {
    "displayName": "MVP",
    "symbol": "MVP"
  },
  "MVR": {
    "displayName": "روفيه جزر المالديف",
    "displayName-count-zero": "روفيه جزر المالديف",
    "displayName-count-one": "روفيه جزر المالديف",
    "displayName-count-two": "روفيه جزر المالديف",
    "displayName-count-few": "روفيه جزر المالديف",
    "displayName-count-many": "روفيه جزر المالديف",
    "displayName-count-other": "روفيه جزر المالديف",
    "symbol": "MVR"
  }

```

```

    },
    "MWK": {
      "displayName": "كواشا مالاوي",
      "displayName-count-zero": "كواشا مالاوي",
      "displayName-count-one": "كواشا مالاوي",
      "displayName-count-two": "كواشا مالاوي",
      "displayName-count-few": "كواشا مالاوي",
      "displayName-count-many": "كواشا مالاوي",
      "displayName-count-other": "كواشا مالاوي",
      "symbol": "MWK"
    },
    "MXN": {
      "displayName": "بيزو مكسيكي",
      "displayName-count-zero": "بيزو مكسيكي",
      "displayName-count-one": "بيزو مكسيكي",
      "displayName-count-two": "بيزو مكسيكي",
      "displayName-count-few": "بيزو مكسيكي",
      "displayName-count-many": "بيزو مكسيكي",
      "displayName-count-other": "بيزو مكسيكي",
      "symbol": "MX$",
      "symbol-alt-narrow": "MX$"
    },
    "MXP": {
      "displayName": "بيزو فضي مكسيكي - 1992-1861",
      "symbol": "MXP"
    },
    "MXV": {
      "displayName": "MXV",
      "symbol": "MXV"
    },
    "MYR": {
      "displayName": "رينغيت ماليزي",
      "displayName-count-zero": "رينغيت ماليزي",
      "displayName-count-one": "رينغيت ماليزي",
      "displayName-count-two": "رينغيت ماليزي",
      "displayName-count-few": "رينغيت ماليزي",
      "displayName-count-many": "رينغيت ماليزي",
      "displayName-count-other": "رينغيت ماليزي",
      "symbol": "MYR",
      "symbol-alt-narrow": "RM"
    },
    "MZE": {
      "displayName": "اسكود موزمبيق",
      "symbol": "MZE"
    },
    "MZM": {
      "displayName": "MZM",
      "symbol": "MZM"
    },
    "MZN": {
      "displayName": "متكال موزمبيق",
      "displayName-count-zero": "متكال موزمبيق",
      "displayName-count-one": "متكال موزمبيق",
      "displayName-count-two": "متكال موزمبيق",
      "displayName-count-few": "متكال موزمبيق",
      "displayName-count-many": "متكال موزمبيق",
      "displayName-count-other": "متكال موزمبيق",

```

```

    "symbol": "MZN"
  },
  "NAD": {
    "displayName": "دولار نامیبي",
    "displayName-count-zero": "دولار نامیبي",
    "displayName-count-one": "دولار نامیبي",
    "displayName-count-two": "دولار نامیبي",
    "displayName-count-few": "دولار نامیبي",
    "displayName-count-many": "دولار نامیبي",
    "displayName-count-other": "دولار نامیبي",
    "symbol": "NAD",
    "symbol-alt-narrow": "$"
  },
  "NGN": {
    "displayName": "نايرا نیجیری",
    "displayName-count-zero": "نايرا نیجیری",
    "displayName-count-one": "نايرا نیجیری",
    "displayName-count-two": "نايرا نیجیری",
    "displayName-count-few": "نايرا نیجیری",
    "displayName-count-many": "نايرا نیجیری",
    "displayName-count-other": "نايرا نیجیری",
    "symbol": "NGN",
    "symbol-alt-narrow": "₦"
  },
  "NIC": {
    "displayName": "کوردوبه نیکاراگوا",
    "symbol": "NIC"
  },
  "NIO": {
    "displayName": "قرطبة نیکاراگوا",
    "displayName-count-zero": "قرطبة نیکاراگوا",
    "displayName-count-one": "قرطبة نیکاراگوا",
    "displayName-count-two": "قرطبة نیکاراگوا",
    "displayName-count-few": "قرطبة نیکاراگوا",
    "displayName-count-many": "قرطبة نیکاراگوا",
    "displayName-count-other": "قرطبة نیکاراگوا",
    "symbol": "NIO",
    "symbol-alt-narrow": "C$"
  },
  "NLG": {
    "displayName": "جلدر هولندی",
    "symbol": "NLG"
  },
  "NOK": {
    "displayName": "کرونة نرویجیة",
    "displayName-count-zero": "کرونة نرویجیة",
    "displayName-count-one": "کرونة نرویجیة",
    "displayName-count-two": "کرونة نرویجیة",
    "displayName-count-few": "کرونة نرویجیة",
    "displayName-count-many": "کرونة نرویجیة",
    "displayName-count-other": "کرونة نرویجیة",
    "symbol": "NOK",
    "symbol-alt-narrow": "kr"
  },
  "NPR": {
    "displayName": "روبية نیبالی",
    "displayName-count-zero": "روبية نیبالی",

```



```

    "displayName-count-one": "روبية نيپالي",
    "displayName-count-two": "روبية نيپالي",
    "displayName-count-few": "روبية نيپالي",
    "displayName-count-many": "روبية نيپالي",
    "displayName-count-other": "روبية نيپالي",
    "symbol": "NPR",
    "symbol-alt-narrow": "Rs"
  },
  "NZD": {
    "displayName": "دولار نيوزيلندي",
    "displayName-count-zero": "دولار نيوزيلندي",
    "displayName-count-one": "دولار نيوزيلندي",
    "displayName-count-two": "دولار نيوزيلندي",
    "displayName-count-few": "دولار نيوزيلندي",
    "displayName-count-many": "دولار نيوزيلندي",
    "displayName-count-other": "دولار نيوزيلندي",
    "symbol": "NZ$",
    "symbol-alt-narrow": "NZ$"
  },
  "OMR": {
    "displayName": "ريال عماني",
    "displayName-count-zero": "ريال عماني",
    "displayName-count-one": "ريال عماني",
    "displayName-count-two": "ريال عماني",
    "displayName-count-few": "ريال عماني",
    "displayName-count-many": "ريال عماني",
    "displayName-count-other": "ريال عماني",
    "symbol": "ر.ع."
  },
  "PAB": {
    "displayName": "بالبوا بنمي",
    "displayName-count-zero": "بالبوا بنمي",
    "displayName-count-one": "بالبوا بنمي",
    "displayName-count-two": "بالبوا بنمي",
    "displayName-count-few": "بالبوا بنمي",
    "displayName-count-many": "بالبوا بنمي",
    "displayName-count-other": "بالبوا بنمي",
    "symbol": "PAB"
  },
  "PEI": {
    "displayName": "PEI",
    "symbol": "PEI"
  },
  "PEN": {
    "displayName": "سول بيروفي",
    "displayName-count-zero": "سول بيروفي",
    "displayName-count-one": "سول بيروفي",
    "displayName-count-two": "سول بيروفي",
    "displayName-count-few": "سول بيروفي",
    "displayName-count-many": "سول بيروفي",
    "displayName-count-other": "سول بيروفي",
    "symbol": "PEN"
  },
  "PES": {
    "displayName": "PES",
    "symbol": "PES"
  },
},

```

```

"PGK": {
  "displayName": "كيننا بابوا غينيا الجديدة",
  "displayName-count-zero": "كيننا بابوا غينيا الجديدة",
  "displayName-count-one": "كيننا بابوا غينيا الجديدة",
  "displayName-count-two": "كيننا بابوا غينيا الجديدة",
  "displayName-count-few": "كيننا بابوا غينيا الجديدة",
  "displayName-count-many": "كيننا بابوا غينيا الجديدة",
  "displayName-count-other": "كيننا بابوا غينيا الجديدة",
  "symbol": "PGK"
},
"PHP": {
  "displayName": "بيزو فلبيني",
  "displayName-count-zero": "بيزو فلبيني",
  "displayName-count-one": "بيزو فلبيني",
  "displayName-count-two": "بيزو فلبيني",
  "displayName-count-few": "بيزو فلبيني",
  "displayName-count-many": "بيزو فلبيني",
  "displayName-count-other": "بيزو فلبيني",
  "symbol": "PHP",
  "symbol-alt-narrow": "₱"
},
"PKR": {
  "displayName": "روبية باكستاني",
  "displayName-count-zero": "روبية باكستاني",
  "displayName-count-one": "روبية باكستاني",
  "displayName-count-two": "روبية باكستاني",
  "displayName-count-few": "روبية باكستاني",
  "displayName-count-many": "روبية باكستاني",
  "displayName-count-other": "روبية باكستاني",
  "symbol": "PKR",
  "symbol-alt-narrow": "Rs"
},
"PLN": {
  "displayName": "زلوتي بولندي",
  "displayName-count-zero": "زلوتي بولندي",
  "displayName-count-one": "زلوتي بولندي",
  "displayName-count-two": "زلوتي بولندي",
  "displayName-count-few": "زلوتي بولندي",
  "displayName-count-many": "زلوتي بولندي",
  "displayName-count-other": "زلوتي بولندي",
  "symbol": "PLN",
  "symbol-alt-narrow": "zł"
},
"PLZ": {
  "displayName": "زلوتي بولندي - 1995-1950",
  "symbol": "PLZ"
},
"PTE": {
  "displayName": "اسكود برتغالي",
  "symbol": "PTE"
},
"PYG": {
  "displayName": "غواراني باراغواي",
  "displayName-count-zero": "غواراني باراغواي",
  "displayName-count-one": "غواراني باراغواي",
  "displayName-count-two": "غواراني باراغواي",
  "displayName-count-few": "غواراني باراغواي",

```

```

    "displayName-count-many": "غواراني باراغواي",
    "displayName-count-other": "غواراني باراغواي",
    "symbol": "PYG",
    "symbol-alt-narrow": "₲"
  },
  "QAR": {
    "displayName": "ڤل قطري",
    "displayName-count-zero": "ڤل قطري",
    "displayName-count-one": "ڤل قطري",
    "displayName-count-two": "ڤل قطري",
    "displayName-count-few": "ڤل قطري",
    "displayName-count-many": "ڤل قطري",
    "displayName-count-other": "ڤل قطري",
    "symbol": "ر.ق."
  },
  "RHD": {
    "displayName": "دولار روديسي",
    "symbol": "RHD"
  },
  "ROL": {
    "displayName": "ليو روماني قديم",
    "symbol": "ROL"
  },
  "RON": {
    "displayName": "ليو روماني",
    "displayName-count-zero": "ليو روماني",
    "displayName-count-one": "ليو روماني",
    "displayName-count-two": "ليو روماني",
    "displayName-count-few": "ليو روماني",
    "displayName-count-many": "ليو روماني",
    "displayName-count-other": "ليو روماني",
    "symbol": "RON",
    "symbol-alt-narrow": "lei"
  },
  "RSD": {
    "displayName": "دينار صربي",
    "displayName-count-zero": "دينار صربي",
    "displayName-count-one": "دينار صربي",
    "displayName-count-two": "ديناران صربيان",
    "displayName-count-few": "دينارات صربية",
    "displayName-count-many": "دينارًا صربيًا",
    "displayName-count-other": "دينار صربي",
    "symbol": "RSD"
  },
  "RUB": {
    "displayName": "روبل روسي",
    "displayName-count-zero": "روبل روسي",
    "displayName-count-one": "روبل روسي",
    "displayName-count-two": "روبل روسي",
    "displayName-count-few": "روبل روسي",
    "displayName-count-many": "روبل روسي",
    "displayName-count-other": "روبل روسي",
    "symbol": "RUB",
    "symbol-alt-narrow": "₽"
  },
  "RUR": {
    "displayName": "روبل روسي - 1998-1991",

```

```

    "symbol": "RUR",
    "symbol-alt-narrow": "p."
  },
  "RWF": {
    "displayName": "فرنك رواندي",
    "displayName-count-zero": "فرنك رواندي",
    "displayName-count-one": "فرنك رواندي",
    "displayName-count-two": "فرنك رواندي",
    "displayName-count-few": "فرنك رواندي",
    "displayName-count-many": "فرنك رواندي",
    "displayName-count-other": "فرنك رواندي",
    "symbol": "RWF",
    "symbol-alt-narrow": "RF"
  },
  "SAR": {
    "displayName": "ريال سعودي",
    "displayName-count-zero": "ريال سعودي",
    "displayName-count-one": "ريال سعودي",
    "displayName-count-two": "ريال سعودي",
    "displayName-count-few": "ريال سعودي",
    "displayName-count-many": "ريال سعودي",
    "displayName-count-other": "ريال سعودي",
    "symbol": "ر.س."
  },
  "SBD": {
    "displayName": "دولار جزر سليمان",
    "displayName-count-zero": "دولار جزر سليمان",
    "displayName-count-one": "دولار جزر سليمان",
    "displayName-count-two": "دولار جزر سليمان",
    "displayName-count-few": "دولار جزر سليمان",
    "displayName-count-many": "دولار جزر سليمان",
    "displayName-count-other": "دولار جزر سليمان",
    "symbol": "SBD",
    "symbol-alt-narrow": "SB$"
  },
  "SCR": {
    "displayName": "روبية سيشيلية",
    "displayName-count-zero": "روبية سيشيلية",
    "displayName-count-one": "روبية سيشيلية",
    "displayName-count-two": "روبية سيشيلية",
    "displayName-count-few": "روبية سيشيلية",
    "displayName-count-many": "روبية سيشيلية",
    "displayName-count-other": "روبية سيشيلية",
    "symbol": "SCR"
  },
  "SDD": {
    "displayName": "دينار سوداني",
    "symbol": "د.س."
  },
  "SDG": {
    "displayName": "جنيه سوداني",
    "displayName-count-zero": "جنيه سوداني",
    "displayName-count-one": "جنيه سوداني",
    "displayName-count-two": "جنيه سوداني",
    "displayName-count-few": "جنيهات سودانية",
    "displayName-count-many": "جنيهًا سودانيًا",
    "displayName-count-other": "جنيه سوداني",
  }

```

```

    "symbol": "ج.س.",
  },
  "SDP": {
    "displayName": "جنيه سوداني قديم",
    "symbol": "SDP"
  },
  "SEK": {
    "displayName": "كرونة سويدية",
    "displayName-count-zero": "كرونة سويدية",
    "displayName-count-one": "كرونة سويدية",
    "displayName-count-two": "كرونة سويدية",
    "displayName-count-few": "كرونة سويدية",
    "displayName-count-many": "كرونة سويدية",
    "displayName-count-other": "كرونة سويدية",
    "symbol": "SEK",
    "symbol-alt-narrow": "kr"
  },
  "SGD": {
    "displayName": "دولار سنغافوري",
    "displayName-count-zero": "دولار سنغافوري",
    "displayName-count-one": "دولار سنغافوري",
    "displayName-count-two": "دولار سنغافوري",
    "displayName-count-few": "دولار سنغافوري",
    "displayName-count-many": "دولار سنغافوري",
    "displayName-count-other": "دولار سنغافوري",
    "symbol": "SGD",
    "symbol-alt-narrow": "$"
  },
  "SHP": {
    "displayName": "جنيه سانت هيلين",
    "displayName-count-zero": "جنيه سانت هيلين",
    "displayName-count-one": "جنيه سانت هيلين",
    "displayName-count-two": "جنيه سانت هيلين",
    "displayName-count-few": "جنيه سانت هيلين",
    "displayName-count-many": "جنيه سانت هيلين",
    "displayName-count-other": "جنيه سانت هيلين",
    "symbol": "SHP",
    "symbol-alt-narrow": "£"
  },
  "SIT": {
    "displayName": "تولار سلوفيني",
    "symbol": "SIT"
  },
  "SKK": {
    "displayName": "كرونة سلوفاكية",
    "symbol": "SKK"
  },
  "SLL": {
    "displayName": "ليون سيراليوني",
    "displayName-count-zero": "ليون سيراليوني",
    "displayName-count-one": "ليون سيراليوني",
    "displayName-count-two": "ليون سيراليوني",
    "displayName-count-few": "ليون سيراليوني",
    "displayName-count-many": "ليون سيراليوني",
    "displayName-count-other": "ليون سيراليوني",
    "symbol": "SLL"
  },
},

```

```

"SOS": {
  "displayName": "شلن صومالي",
  "displayName-count-zero": "شلن صومالي",
  "displayName-count-one": "شلن صومالي",
  "displayName-count-two": "شلن صومالي",
  "displayName-count-few": "شلن صومالي",
  "displayName-count-many": "شلن صومالي",
  "displayName-count-other": "شلن صومالي",
  "symbol": "SOS"
},
"SRD": {
  "displayName": "دولار سورينامي",
  "displayName-count-zero": "دولار سورينامي",
  "displayName-count-one": "دولار سورينامي",
  "displayName-count-two": "دولار سورينامي",
  "displayName-count-few": "دولار سورينامي",
  "displayName-count-many": "دولار سورينامي",
  "displayName-count-other": "دولار سورينامي",
  "symbol": "SRD",
  "symbol-alt-narrow": "SR$"
},
"SRG": {
  "displayName": "جلدر سورينامي",
  "symbol": "SRG"
},
"SSP": {
  "displayName": "جنيه جنوب السودان",
  "displayName-count-zero": "جنيه جنوب السودان",
  "displayName-count-one": "جنيه جنوب السودان",
  "displayName-count-two": "جنيهات جنوب السودان",
  "displayName-count-few": "جنيهات جنوب السودان",
  "displayName-count-many": "جنيهًا جنوب السودان",
  "displayName-count-other": "جنيه جنوب السودان",
  "symbol": "SSP",
  "symbol-alt-narrow": "£"
},
"STD": {
  "displayName": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-zero": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-one": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-two": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-few": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-many": "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-other": "دوبرا ساو تومي وبرينسيبي",
  "symbol": "STD",
  "symbol-alt-narrow": "Db"
},
"STN": {
  "displayName": "STN",
  "symbol": "STN"
},
"SUR": {
  "displayName": "روبل سوفيتي",
  "symbol": "SUR"
},
"SVC": {
  "displayName": "كولون سلفادوري",

```

```

    "symbol": "SVC"
  },
  "SYP": {
    "displayName": "ليرة سورية",
    "displayName-count-zero": "ليرة سورية",
    "displayName-count-one": "ليرة سورية",
    "displayName-count-two": "ليرة سورية",
    "displayName-count-few": "ليرة سورية",
    "displayName-count-many": "ليرة سورية",
    "displayName-count-other": "ليرة سورية",
    "symbol": "ل.س.",
    "symbol-alt-narrow": "£"
  },
  "SZL": {
    "displayName": "ليلانجيني سوازيلندي",
    "displayName-count-zero": "ليلانجيني سوازيلندي",
    "displayName-count-one": "ليلانجيني سوازيلندي",
    "displayName-count-two": "ليلانجيني سوازيلندي",
    "displayName-count-few": "ليلانجيني سوازيلندي",
    "displayName-count-many": "ليلانجيني سوازيلندي",
    "displayName-count-other": "ليلانجيني سوازيلندي",
    "symbol": "SZL"
  },
  "THB": {
    "displayName": "باخت تايلاندي",
    "displayName-count-zero": "باخت تايلاندي",
    "displayName-count-one": "باخت تايلاندي",
    "displayName-count-two": "باخت تايلاندي",
    "displayName-count-few": "باخت تايلاندي",
    "displayName-count-many": "باخت تايلاندي",
    "displayName-count-other": "باخت تايلاندي",
    "symbol": "฿",
    "symbol-alt-narrow": "฿"
  },
  "TJR": {
    "displayName": "روبل طاجيكستاني",
    "symbol": "TJR"
  },
  "TJS": {
    "displayName": "سوموني طاجيكستاني",
    "displayName-count-zero": "سوموني طاجيكستاني",
    "displayName-count-one": "سوموني طاجيكستاني",
    "displayName-count-two": "سوموني طاجيكستاني",
    "displayName-count-few": "سوموني طاجيكستاني",
    "displayName-count-many": "سوموني طاجيكستاني",
    "displayName-count-other": "سوموني طاجيكستاني",
    "symbol": "TJS"
  },
  "TMM": {
    "displayName": "مانات تركمنستاني",
    "symbol": "TMM"
  },
  "TMT": {
    "displayName": "مانات تركمانستان",
    "displayName-count-zero": "مانات تركمانستان",
    "displayName-count-one": "مانات تركمانستان",
    "displayName-count-two": "مانات تركمانستان",

```

```

        "displayName-count-few": "مانات تركمانستان",
        "displayName-count-many": "مانات تركمانستان",
        "displayName-count-other": "مانات تركمانستان",
        "symbol": "TMT"
    },
    "TND": {
        "displayName": "دينار تونسي",
        "displayName-count-zero": "دينار تونسي",
        "displayName-count-one": "دينار تونسي",
        "displayName-count-two": "ديناران تونسيان",
        "displayName-count-few": "دينارات تونسية",
        "displayName-count-many": "دينارًا تونسيًا",
        "displayName-count-other": "دينار تونسي",
        "symbol": "د.ت."
    },
    "TOP": {
        "displayName": "بانغا تونغا",
        "displayName-count-zero": "بانغا تونغا",
        "displayName-count-one": "بانغا تونغا",
        "displayName-count-two": "بانغا تونغا",
        "displayName-count-few": "بانغا تونغا",
        "displayName-count-many": "بانغا تونغا",
        "displayName-count-other": "بانغا تونغا",
        "symbol": "TOP",
        "symbol-alt-narrow": "T$"
    },
    "TPE": {
        "displayName": "اسكود تيموري",
        "symbol": "TPE"
    },
    "TRL": {
        "displayName": "ليرة تركي",
        "symbol": "TRL"
    },
    "TRY": {
        "displayName": "ليرة تركية",
        "displayName-count-zero": "ليرة تركية",
        "displayName-count-one": "ليرة تركية",
        "displayName-count-two": "ليرة تركية",
        "displayName-count-few": "ليرة تركية",
        "displayName-count-many": "ليرة تركية",
        "displayName-count-other": "ليرة تركية",
        "symbol": "TRY",
        "symbol-alt-narrow": "₺",
        "symbol-alt-variant": "TL"
    },
    "TTD": {
        "displayName": "دولار ترينداد وتوباغو",
        "displayName-count-zero": "دولار ترينداد وتوباغو",
        "displayName-count-one": "دولار ترينداد وتوباغو",
        "displayName-count-two": "دولار ترينداد وتوباغو",
        "displayName-count-few": "دولار ترينداد وتوباغو",
        "displayName-count-many": "دولار ترينداد وتوباغو",
        "displayName-count-other": "دولار ترينداد وتوباغو",
        "symbol": "TTD",
        "symbol-alt-narrow": "TT$"
    },
    },

```



```

"TWD": {
  "displayName": "دولار تايواني",
  "displayName-count-zero": "دولار تايواني",
  "displayName-count-one": "دولار تايواني",
  "displayName-count-two": "دولار تايواني",
  "displayName-count-few": "دولار تايواني",
  "displayName-count-many": "دولار تايواني",
  "displayName-count-other": "دولار تايواني",
  "symbol": "NT$",
  "symbol-alt-narrow": "NT$"
},
"TZS": {
  "displayName": "شلن تنزاني",
  "displayName-count-zero": "شلن تنزاني",
  "displayName-count-one": "شلن تنزاني",
  "displayName-count-two": "شلن تنزاني",
  "displayName-count-few": "شلن تنزاني",
  "displayName-count-many": "شلن تنزاني",
  "displayName-count-other": "شلن تنزاني",
  "symbol": "TZS"
},
"UAH": {
  "displayName": "هريفنيا اوكراني",
  "displayName-count-zero": "هريفنيا اوكراني",
  "displayName-count-one": "هريفنيا اوكراني",
  "displayName-count-two": "هريفنيا اوكراني",
  "displayName-count-few": "هريفنيا اوكراني",
  "displayName-count-many": "هريفنيا اوكراني",
  "displayName-count-other": "هريفنيا اوكراني",
  "symbol": "UAH",
  "symbol-alt-narrow": "₴"
},
"UAK": {
  "displayName": "UAK",
  "symbol": "UAK"
},
"UGS": {
  "displayName": "شلن اوغندي - 1987-1966",
  "symbol": "UGS"
},
"UGX": {
  "displayName": "شلن اوغندي",
  "displayName-count-zero": "شلن اوغندي",
  "displayName-count-one": "شلن اوغندي",
  "displayName-count-two": "شلن اوغندي",
  "displayName-count-few": "شلن اوغندي",
  "displayName-count-many": "شلن اوغندي",
  "displayName-count-other": "شلن اوغندي",
  "symbol": "UGX"
},
"USD": {
  "displayName": "دولار امريكي",
  "displayName-count-zero": "دولار امريكي",
  "displayName-count-one": "دولار امريكي",
  "displayName-count-two": "دولار امريكي",
  "displayName-count-few": "دولار امريكي",
  "displayName-count-many": "دولار امريكي",

```

```

        "displayName-count-other": "دولار أمريكي",
        "symbol": "US$",
        "symbol-alt-narrow": "US$"
    },
    "USN": {
        "displayName": "دولار أمريكي (اليوم التالي)",
        "symbol": "USN"
    },
    "USS": {
        "displayName": "دولار أمريكي (نفس اليوم)",
        "symbol": "USS"
    },
    "UYI": {
        "displayName": "UYI",
        "symbol": "UYI"
    },
    "UYP": {
        "displayName": "بيزو أوروغواي - 1993-1975",
        "symbol": "UYP"
    },
    "UYU": {
        "displayName": "بيزو أوروغواي",
        "displayName-count-zero": "بيزو أوروغواي",
        "displayName-count-one": "بيزو أوروغواي",
        "displayName-count-two": "بيزو أوروغواي",
        "displayName-count-few": "بيزو أوروغواي",
        "displayName-count-many": "بيزو أوروغواي",
        "displayName-count-other": "بيزو أوروغواي",
        "symbol": "UYU",
        "symbol-alt-narrow": "UY$"
    },
    "UZS": {
        "displayName": "سوم أوزبكستاني",
        "displayName-count-zero": "سوم أوزبكستاني",
        "displayName-count-one": "سوم أوزبكستاني",
        "displayName-count-two": "سوم أوزبكستاني",
        "displayName-count-few": "سوم أوزبكستاني",
        "displayName-count-many": "سوم أوزبكستاني",
        "displayName-count-other": "سوم أوزبكستاني",
        "symbol": "UZS"
    },
    "VEB": {
        "displayName": "بوليفار فنزويلي - 2008-1871",
        "symbol": "VEB"
    },
    "VEF": {
        "displayName": "بوليفار فنزويلي",
        "displayName-count-zero": "بوليفار فنزويلي",
        "displayName-count-one": "بوليفار فنزويلي",
        "displayName-count-two": "بوليفار فنزويلي",
        "displayName-count-few": "بوليفار فنزويلي",
        "displayName-count-many": "بوليفار فنزويلي",
        "displayName-count-other": "بوليفار فنزويلي",
        "symbol": "VEF",
        "symbol-alt-narrow": "Bs"
    },
    "VND": {

```

```

        "displayName": "دونج فيتنامي",
        "displayName-count-zero": "دونج فيتنامي",
        "displayName-count-one": "دونج فيتنامي",
        "displayName-count-two": "دونج فيتنامي",
        "displayName-count-few": "دونج فيتنامي",
        "displayName-count-many": "دونج فيتنامي",
        "displayName-count-other": "دونج فيتنامي",
        "symbol": "₫",
        "symbol-alt-narrow": "₫"
    },
    "VNN": {
        "displayName": "VNN",
        "symbol": "VNN"
    },
    "VUV": {
        "displayName": "فاتو فانواتو",
        "displayName-count-zero": "فاتو فانواتو",
        "displayName-count-one": "فاتو فانواتو",
        "displayName-count-two": "فاتو فانواتو",
        "displayName-count-few": "فاتو فانواتو",
        "displayName-count-many": "فاتو فانواتو",
        "displayName-count-other": "فاتو فانواتو",
        "symbol": "VUV"
    },
    "WST": {
        "displayName": "تالا ساموا",
        "displayName-count-zero": "تالا ساموا",
        "displayName-count-one": "تالا ساموا",
        "displayName-count-two": "تالا ساموا",
        "displayName-count-few": "تالا ساموا",
        "displayName-count-many": "تالا ساموا",
        "displayName-count-other": "تالا ساموا",
        "symbol": "WST"
    },
    "XAF": {
        "displayName": "فرنك وسط أفريقي",
        "displayName-count-zero": "فرنك وسط أفريقي",
        "displayName-count-one": "فرنك وسط أفريقي",
        "displayName-count-two": "فرنك وسط أفريقي",
        "displayName-count-few": "فرنك وسط أفريقي",
        "displayName-count-many": "فرنك وسط أفريقي",
        "displayName-count-other": "فرنك وسط أفريقي",
        "symbol": "FCFA"
    },
    "XAG": {
        "displayName": "فضة",
        "symbol": "XAG"
    },
    "XAU": {
        "displayName": "ذهب",
        "symbol": "XAU"
    },
    "XBA": {
        "displayName": "الوحدة الأوروبية المركبة",
        "symbol": "XBA"
    },
    "XBB": {

```

```

    "displayName": "الوحدة المالية الأوروبية",
    "symbol": "XBB"
  },
  "XBC": {
    "displayName": "الوحدة الحسابية الأوروبية",
    "symbol": "XBC"
  },
  "XBD": {
    "displayName": "وحدة الحساب الأوروبية (XBD)",
    "symbol": "XBD"
  },
  "XCD": {
    "displayName": "دولار شرق الكاريبي",
    "displayName-count-zero": "دولار شرق الكاريبي",
    "displayName-count-one": "دولار شرق الكاريبي",
    "displayName-count-two": "دولار شرق الكاريبي",
    "displayName-count-few": "دولار شرق الكاريبي",
    "displayName-count-many": "دولار شرق الكاريبي",
    "displayName-count-other": "دولار شرق الكاريبي",
    "symbol": "EC$",
    "symbol-alt-narrow": "$"
  },
  "XDR": {
    "displayName": "حقوق السحب الخاصة",
    "symbol": "XDR"
  },
  "XEU": {
    "displayName": "وحدة النقد الأوروبية",
    "symbol": "XEU"
  },
  "XFO": {
    "displayName": "فرنك فرنسي ذهبي",
    "symbol": "XFO"
  },
  "XFU": {
    "displayName": "فرنك فرنسي (UIC)",
    "symbol": "XFU"
  },
  "XOF": {
    "displayName": "فرنك غرب أفريقي",
    "displayName-count-zero": "فرنك غرب أفريقي",
    "displayName-count-one": "فرنك غرب أفريقي",
    "displayName-count-two": "فرنك غرب أفريقي",
    "displayName-count-few": "فرنك غرب أفريقي",
    "displayName-count-many": "فرنك غرب أفريقي",
    "displayName-count-other": "فرنك غرب أفريقي",
    "symbol": "CFA"
  },
  "XPD": {
    "displayName": "بالاديوم",
    "symbol": "XPD"
  },
  "XPF": {
    "displayName": "فرنك سي إف بي",
    "displayName-count-zero": "فرنك سي إف بي",
    "displayName-count-one": "فرنك سي إف بي",
    "displayName-count-two": "فرنك سي إف بي",

```

```

        "displayName-count-few": "فرنك سي إف بي",
        "displayName-count-many": "فرنك سي إف بي",
        "displayName-count-other": "فرنك سي إف بي",
        "symbol": "CFPF"
    },
    "XPT": {
        "displayName": "البلاطين",
        "symbol": "XPT"
    },
    "XRE": {
        "displayName": "XRE",
        "symbol": "XRE"
    },
    "XSU": {
        "displayName": "XSU",
        "symbol": "XSU"
    },
    "XTS": {
        "displayName": "كود اختبار العملة",
        "symbol": "XTS"
    },
    "XUA": {
        "displayName": "XUA",
        "symbol": "XUA"
    },
    "XXX": {
        "displayName": "عملة غير معروفة",
        "displayName-count-zero": "(عملة غير معروفة)",
        "displayName-count-one": "(عملة غير معروفة)",
        "displayName-count-two": "(عملة غير معروفة)",
        "displayName-count-few": "(عملة غير معروفة)",
        "displayName-count-many": "(عملة غير معروفة)",
        "displayName-count-other": "(عملة غير معروفة)",
        "symbol": "****"
    },
    "YDD": {
        "displayName": "دينار يمني",
        "symbol": "YDD"
    },
    "YER": {
        "displayName": "ر.ي.",
        "displayName-count-zero": "ر.ي.",
        "displayName-count-one": "ر.ي.",
        "displayName-count-two": "ر.ي.",
        "displayName-count-few": "ر.ي.",
        "displayName-count-many": "ر.ي.",
        "displayName-count-other": "ر.ي.",
        "symbol": "ر.ي."
    },
    "YUD": {
        "displayName": "دينار يوغسلافي",
        "symbol": "YUD"
    },
    "YUM": {
        "displayName": "YUM",
        "symbol": "YUM"
    },
    },

```

```

"YUN": {
  "displayName": "دينار يوغسلافي قابل للتحويل",
  "symbol": "YUN"
},
"YUR": {
  "displayName": "YUR",
  "symbol": "YUR"
},
"ZAL": {
  "displayName": "راند جنوب أفريقيا -مالي",
  "symbol": "ZAL"
},
"ZAR": {
  "displayName": "راند جنوب أفريقيا",
  "displayName-count-zero": "راند جنوب أفريقيا",
  "displayName-count-one": "راند جنوب أفريقيا",
  "displayName-count-two": "راند جنوب أفريقيا",
  "displayName-count-few": "راند جنوب أفريقيا",
  "displayName-count-many": "راند جنوب أفريقيا",
  "displayName-count-other": "راند جنوب أفريقيا",
  "symbol": "ZAR",
  "symbol-alt-narrow": "R"
},
"ZMK": {
  "displayName": "2012-1968 - كواشا زامبي",
  "displayName-count-zero": "2012-1968 - كواشا زامبي",
  "displayName-count-one": "2012-1968 - كواشا زامبي",
  "displayName-count-two": "2012-1968 - كواشا زامبي",
  "displayName-count-few": "2012-1968 - كواشا زامبي",
  "displayName-count-many": "2012-1968 - كواشا زامبي",
  "displayName-count-other": "2012-1968 - كواشا زامبي",
  "symbol": "ZMK"
},
"ZMW": {
  "displayName": "كواشا زامبي",
  "displayName-count-zero": "كواشا زامبي",
  "displayName-count-one": "كواشا زامبي",
  "displayName-count-two": "كواشا زامبي",
  "displayName-count-few": "كواشا زامبي",
  "displayName-count-many": "كواشا زامبي",
  "displayName-count-other": "كواشا زامبي",
  "symbol": "ZMW",
  "symbol-alt-narrow": "ZK"
},
"ZRN": {
  "displayName": "زائير زائيري جديد",
  "symbol": "ZRN"
},
"ZRZ": {
  "displayName": "زائير زائيري",
  "symbol": "ZRZ"
},
"ZWD": {
  "displayName": "دولار زمبابوي",
  "symbol": "ZWD"
},
"ZWL": {

```

```
"displayName": "2009 زمبابوي دولار",  
  "symbol": "ZWL"  
},  
  "ZWR": {  
    "displayName": "ZWR",  
    "symbol": "ZWR"  
  }  
}  
}  
}  
}
```

CURRENCIES.JSX

```
{
  "main";
  {
    "ar";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13686 $",
          "_cldrVersion";
          "32";
        }
        "language";
        "ar";
      }
    }
    "numbers";
    {
      "currencies";
      {
        "ADP";
        {
          "displayName";
          "بيستا أندوري",
          "symbol";
          "ADP";
        }
      }
      "AED";
      {
        "displayName";
        "درهم إماراتي",
        "displayName-count-zero";
        "درهم إماراتي",
        "displayName-count-one";
        "درهم إماراتي",
        "displayName-count-two";
        "درهم إماراتي",
        "displayName-count-few";
        "درهم إماراتي",
        "displayName-count-many";
      }
    }
  }
}
```

```

        "درهم إماراتي",
        "displayName-count-other";
        "درهم إماراتي",
        "symbol";
        "د.إ.";
    }
    "AFA";
    {
        "displayName";
        "2002-1927 - أفغاني",
        "symbol";
        "AFA";
    }
    "AFN";
    {
        "displayName";
        "أفغاني",
        "displayName-count-zero";
        "أفغاني أفغانستان",
        "displayName-count-one";
        "أفغاني أفغانستان",
        "displayName-count-two";
        "أفغاني أفغانستان",
        "displayName-count-few";
        "أفغاني أفغانستان",
        "displayName-count-many";
        "أفغاني أفغانستان",
        "displayName-count-other";
        "أفغاني أفغانستان",
        "symbol";
        "AFN";
    }
    "ALK";
    {
        "displayName";
        "ALK",
        "symbol";
        "ALK";
    }
    "ALL";
    {
        "displayName";
        "ليك ألباني",
        "displayName-count-zero";
        "ليك ألباني",
        "displayName-count-one";
        "ليك ألباني",
        "displayName-count-two";
        "ليك ألباني",
        "displayName-count-few";
        "ليك ألباني",
        "displayName-count-many";
        "ليك ألباني",
        "displayName-count-other";
        "ليك ألباني",
        "symbol";
        "ALL";
    }

```



```

    }
    "AMD";
    {
        "displayName";
        "درام أرميني",
        "displayName-count-zero";
        "درام أرميني",
        "displayName-count-one";
        "درام أرميني",
        "displayName-count-two";
        "درام أرميني",
        "displayName-count-few";
        "درام أرميني",
        "displayName-count-many";
        "درام أرميني",
        "displayName-count-other";
        "درام أرميني",
        "symbol";
        "AMD";
    }
    "ANG";
    {
        "displayName";
        "غيلدر أنتيلي هولندي",
        "displayName-count-zero";
        "غيلدر أنتيلي هولندي",
        "displayName-count-one";
        "غيلدر أنتيلي هولندي",
        "displayName-count-two";
        "غيلدر أنتيلي هولندي",
        "displayName-count-few";
        "غيلدر أنتيلي هولندي",
        "displayName-count-many";
        "غيلدر أنتيلي هولندي",
        "displayName-count-other";
        "غيلدر أنتيلي هولندي",
        "symbol";
        "ANG";
    }
    "AOA";
    {
        "displayName";
        "كوانزا أنغولي",
        "displayName-count-zero";
        "كوانزا أنغولي",
        "displayName-count-one";
        "كوانزا أنغولي",
        "displayName-count-two";
        "كوانزا أنغولي",
        "displayName-count-few";
        "كوانزا أنغولي",
        "displayName-count-many";
        "كوانزا أنغولي",
        "displayName-count-other";
        "كوانزا أنغولي",
        "symbol";
        "AOA";
    }

```

```

        "symbol-alt-narrow";
        "Kz";
    }
    "AOK";
    {
        "displayName";
        "1990-1977 - أنجولي",
        "symbol";
        "AOK";
    }
    "AON";
    {
        "displayName";
        "2000-1990 - أنجولي جديدة",
        "symbol";
        "AON";
    }
    "AOR";
    {
        "displayName";
        "1999 - 1995 - أنجولي معدلة",
        "symbol";
        "AOR";
    }
    "ARA";
    {
        "displayName";
        "استرال أرجنتيني",
        "symbol";
        "ARA";
    }
    "ARL";
    {
        "displayName";
        "ARL",
        "symbol";
        "ARL";
    }
    "ARM";
    {
        "displayName";
        "ARM",
        "symbol";
        "ARM";
    }
    "ARP";
    {
        "displayName";
        "1985-1983 - بيزو أرجنتيني",
        "symbol";
        "ARP";
    }
    "ARS";
    {
        "displayName";
        "بيزو أرجنتيني",
        "displayName-count-zero";
    }

```

```

        "بيزو أرجنتيني",
        "displayName-count-one";
        "بيزو أرجنتيني",
        "displayName-count-two";
        "بيزو أرجنتيني",
        "displayName-count-few";
        "بيزو أرجنتيني",
        "displayName-count-many";
        "بيزو أرجنتيني",
        "displayName-count-other";
        "بيزو أرجنتيني",
        "symbol";
        "ARS",
        "symbol-alt-narrow";
        "AR$";
    }
    "ATS";
    {
        "displayName";
        "شلن نمساوي",
        "symbol";
        "ATS";
    }
    "AUD";
    {
        "displayName";
        "دولار أسترالي",
        "displayName-count-zero";
        "دولار أسترالي",
        "displayName-count-one";
        "دولار أسترالي",
        "displayName-count-two";
        "دولار أسترالي",
        "displayName-count-few";
        "دولار أسترالي",
        "displayName-count-many";
        "دولار أسترالي",
        "displayName-count-other";
        "دولار أسترالي",
        "symbol";
        "AU$";
        "symbol-alt-narrow";
        "AU$";
    }
    "AWG";
    {
        "displayName";
        "فلورن أروبي",
        "displayName-count-zero";
        "فلورن أروبي",
        "displayName-count-one";
        "فلورن أروبي",
        "displayName-count-two";
        "فلورن أروبي",
        "displayName-count-few";
        "فلورن أروبي",
        "displayName-count-many";
    }

```

```

        "فلورن أروبي",
        "displayName-count-other";
        "فلورن أروبي",
        "symbol";
        "AWG";
    }
    "AZM";
    {
        "displayName";
        "مانات أذربيجاني",
        "symbol";
        "AZM";
    }
    "AZN";
    {
        "displayName";
        "مانات أذربيجان",
        "displayName-count-zero";
        "مانت أذربيجاني",
        "displayName-count-one";
        "مانت أذربيجاني",
        "displayName-count-two";
        "مانت أذربيجاني",
        "displayName-count-few";
        "مانت أذربيجاني",
        "displayName-count-many";
        "مانت أذربيجاني",
        "displayName-count-other";
        "مانت أذربيجاني",
        "symbol";
        "AZN";
    }
    "BAD";
    {
        "displayName";
        "دينار البوسنة والهرسك",
        "symbol";
        "BAD";
    }
    "BAM";
    {
        "displayName";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-zero";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-one";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-two";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-few";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-many";
        "مارك البوسنة والهرسك قابل للتحويل",
        "displayName-count-other";
        "مارك البوسنة والهرسك قابل للتحويل",
        "symbol";
        "BAM";
    }

```

```

        "symbol-alt-narrow";
        "KM";
    }
    "BAN";
    {
        "displayName";
        "BAN",
        "symbol";
        "BAN";
    }
    "BBD";
    {
        "displayName";
        "دولار بربادوسي",
        "displayName-count-zero";
        "دولار بربادوسي",
        "displayName-count-one";
        "دولار بربادوسي",
        "displayName-count-two";
        "دولار بربادوسي",
        "displayName-count-few";
        "دولار بربادوسي",
        "displayName-count-many";
        "دولار بربادوسي",
        "displayName-count-other";
        "دولار بربادوسي",
        "symbol";
        "BBD",
        "symbol-alt-narrow";
        "BB$";
    }
    "BDT";
    {
        "displayName";
        "তাকা বাংলাদেশি",
        "displayName-count-zero";
        "তাকা বাংলাদেশি",
        "displayName-count-one";
        "তাকা বাংলাদেশি",
        "displayName-count-two";
        "তাকা বাংলাদেশি",
        "displayName-count-few";
        "তাকা বাংলাদেশি",
        "displayName-count-many";
        "তাকা বাংলাদেশি",
        "displayName-count-other";
        "তাকা বাংলাদেশি",
        "symbol";
        "BDT",
        "symbol-alt-narrow";
        "৳";
    }
    "BEC";
    {
        "displayName";
        "فرنك بلجيكي قابل للتحويل",

```

```

        "symbol";
        "BEC";
    }
    "BEF";
    {
        "displayName";
        "فرنك بلجيكي",
        "symbol";
        "BEF";
    }
    "BEL";
    {
        "displayName";
        "فرنك بلجيكي مالي",
        "symbol";
        "BEL";
    }
    "BGL";
    {
        "displayName";
        "BGL",
        "symbol";
        "BGL";
    }
    "BGM";
    {
        "displayName";
        "BGM",
        "symbol";
        "BGM";
    }
    "BGN";
    {
        "displayName";
        "ليف بلغاري",
        "displayName-count-zero";
        "ليف بلغاري",
        "displayName-count-one";
        "ليف بلغاري",
        "displayName-count-two";
        "ليف بلغاري",
        "displayName-count-few";
        "ليف بلغاري",
        "displayName-count-many";
        "ليف بلغاري",
        "displayName-count-other";
        "ليف بلغاري",
        "symbol";
        "BGN";
    }
    "BGO";
    {
        "displayName";
        "BGO",
        "symbol";
        "BGO";
    }
}

```

```

"BHD";
{
  "displayName";
  "دينار بحريني",
  "displayName-count-zero";
  "دينار بحريني",
  "displayName-count-one";
  "دينار بحريني",
  "displayName-count-two";
  "دينار بحريني",
  "displayName-count-few";
  "دينار بحريني",
  "displayName-count-many";
  "دينار بحريني",
  "displayName-count-other";
  "دينار بحريني",
  "symbol";
  "د.ب.";
}
"BIF";
{
  "displayName";
  "فرنك بروندي",
  "displayName-count-zero";
  "فرنك بروندي",
  "displayName-count-one";
  "فرنك بروندي",
  "displayName-count-two";
  "فرنك بروندي",
  "displayName-count-few";
  "فرنك بروندي",
  "displayName-count-many";
  "فرنك بروندي",
  "displayName-count-other";
  "فرنك بروندي",
  "symbol";
  "BIF";
}
"BMD";
{
  "displayName";
  "دولار برمودي",
  "displayName-count-zero";
  "دولار برمودي",
  "displayName-count-one";
  "دولار برمودي",
  "displayName-count-two";
  "دولار برمودي",
  "displayName-count-few";
  "دولار برمودي",
  "displayName-count-many";
  "دولار برمودي",
  "displayName-count-other";
  "دولار برمودي",
  "symbol";
  "BMD",
  "symbol-alt-narrow";
}

```

```

        "BM$";
    }
    "BND";
    {
        "displayName";
        "دولار بروناي",
        "displayName-count-zero";
        "دولار بروناي",
        "displayName-count-one";
        "دولار بروناي",
        "displayName-count-two";
        "دولار بروناي",
        "displayName-count-few";
        "دولار بروناي",
        "displayName-count-many";
        "دولار بروناي",
        "displayName-count-other";
        "دولار بروناي",
        "symbol";
        "BND",
        "symbol-alt-narrow";
        "BN$";
    }
    "BOB";
    {
        "displayName";
        "بولىفيانو بولىفي",
        "displayName-count-zero";
        "بولىفيانو بولىفي",
        "displayName-count-one";
        "بولىفيانو بولىفي",
        "displayName-count-two";
        "بولىفيانو بولىفي",
        "displayName-count-few";
        "بولىفيانو بولىفي",
        "displayName-count-many";
        "بولىفيانو بولىفي",
        "displayName-count-other";
        "بولىفيانو بولىفي",
        "symbol";
        "BOB",
        "symbol-alt-narrow";
        "Bs";
    }
    "BOL";
    {
        "displayName";
        "BOL",
        "symbol";
        "BOL";
    }
    "BOP";
    {
        "displayName";
        "بىزو بولىفي",
        "symbol";
        "BOP";
    }

```



```

    }
    "BOV";
    {
        "displayName";
        "مفدول بوليفي",
        "symbol";
        "BOV";
    }
    "BRB";
    {
        "displayName";
        "نوفو كروزايرو برازيلى - 1986-1967",
        "symbol";
        "BRB";
    }
    "BRC";
    {
        "displayName";
        "كروزادو برازيلى",
        "symbol";
        "BRC";
    }
    "BRE";
    {
        "displayName";
        "كروزايرو برازيلى - 1993-1990",
        "symbol";
        "BRE";
    }
    "BRL";
    {
        "displayName";
        "ريال برازيلى",
        "displayName-count-zero";
        "ريال برازيلى",
        "displayName-count-one";
        "ريال برازيلى",
        "displayName-count-two";
        "ريال برازيلى",
        "displayName-count-few";
        "ريال برازيلى",
        "displayName-count-many";
        "ريال برازيلى",
        "displayName-count-other";
        "ريال برازيلى",
        "symbol";
        "R$",
        "symbol-alt-narrow";
        "R$";
    }
    "BRN";
    {
        "displayName";
        "BRN",
        "symbol";
        "BRN";
    }
}

```

```

"BRR";
{
  "displayName";
  "BRR",
  "symbol";
  "BRR";
}
"BRZ";
{
  "displayName";
  "BRZ",
  "symbol";
  "BRZ";
}
"BSD";
{
  "displayName";
  "دولار باهامي",
  "displayName-count-zero";
  "دولار باهامي",
  "displayName-count-one";
  "دولار باهامي",
  "displayName-count-two";
  "دولار باهامي",
  "displayName-count-few";
  "دولار باهامي",
  "displayName-count-many";
  "دولار باهامي",
  "displayName-count-other";
  "دولار باهامي",
  "symbol";
  "BSD",
  "symbol-alt-narrow";
  "BS$";
}
"BTN";
{
  "displayName";
  "نولتوم بوتاني",
  "displayName-count-zero";
  "نولتوم بوتاني",
  "displayName-count-one";
  "نولتوم بوتاني",
  "displayName-count-two";
  "نولتوم بوتاني",
  "displayName-count-few";
  "نولتوم بوتاني",
  "displayName-count-many";
  "نولتوم بوتاني",
  "displayName-count-other";
  "نولتوم بوتاني",
  "symbol";
  "BTN";
}
"BUK";
{
  "displayName";

```

```

        "کیات بورمی",
        "symbol";
        "BUK";
    }
    "BWP";
    {
        "displayName";
        "بولا بتسوani",
        "displayName-count-zero";
        "بولا بتسوani",
        "displayName-count-one";
        "بولا بتسوani",
        "displayName-count-two";
        "بولا بتسوani",
        "displayName-count-few";
        "بولا بتسوani",
        "displayName-count-many";
        "بولا بتسوani",
        "displayName-count-other";
        "بولا بتسوani",
        "symbol";
        "BWP",
        "symbol-alt-narrow";
        "p";
    }
    "BYB";
    {
        "displayName";
        "روبل بیلاروسی جدید - 1999-1994",
        "symbol";
        "BYB";
    }
    "BYN";
    {
        "displayName";
        "روبل بیلاروسی",
        "displayName-count-zero";
        "روبل بیلاروسی",
        "displayName-count-one";
        "روبل بیلاروسی",
        "displayName-count-two";
        "روبل بیلاروسی",
        "displayName-count-few";
        "روبل بیلاروسی",
        "displayName-count-many";
        "روبل بیلاروسی",
        "displayName-count-other";
        "روبل بیلاروسی",
        "symbol";
        "BYN",
        "symbol-alt-narrow";
        "p.";
    }
    "BYR";
    {
        "displayName";
        "روبل بیلاروسی (۲۰۱۶-۲۰۰۰)",

```

```

        "displayName-count-zero";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "displayName-count-one";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "displayName-count-two";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "displayName-count-few";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "displayName-count-many";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "displayName-count-other";
        "۲۰۱۶-۲۰۰۰ ( روبل بیلاروسي)",
        "symbol";
        "BYR";
    }
    "BZD";
    {
        "displayName";
        "دولار بلیزي",
        "displayName-count-zero";
        "دولار بلیزي",
        "displayName-count-one";
        "دولار بلیزي",
        "displayName-count-two";
        "دولاران بلیزیا",
        "displayName-count-few";
        "دولار بلیزي",
        "displayName-count-many";
        "دولار بلیزي",
        "displayName-count-other";
        "دولار بلیزي",
        "symbol";
        "BZD",
        "symbol-alt-narrow";
        "BZ$";
    }
    "CAD";
    {
        "displayName";
        "دولار کندي",
        "displayName-count-zero";
        "دولار کندي",
        "displayName-count-one";
        "دولار کندي",
        "displayName-count-two";
        "دولار کندي",
        "displayName-count-few";
        "دولار کندي",
        "displayName-count-many";
        "دولار کندي",
        "displayName-count-other";
        "دولار کندي",
        "symbol";
        "CA$",
        "symbol-alt-narrow";
        "CA$";
    }
}

```

```

"CDF";
{
  "displayName";
  "فرنك كونغولي",
  "displayName-count-zero";
  "فرنك كونغولي",
  "displayName-count-one";
  "فرنك كونغولي",
  "displayName-count-two";
  "فرنك كونغولي",
  "displayName-count-few";
  "فرنك كونغولي",
  "displayName-count-many";
  "فرنك كونغولي",
  "displayName-count-other";
  "فرنك كونغولي",
  "symbol";
  "CDF";
}
"CHE";
{
  "displayName";
  "CHE",
  "symbol";
  "CHE";
}
"CHF";
{
  "displayName";
  "فرنك سويسري",
  "displayName-count-zero";
  "فرنك سويسري",
  "displayName-count-one";
  "فرنك سويسري",
  "displayName-count-two";
  "فرنك سويسري",
  "displayName-count-few";
  "فرنك سويسري",
  "displayName-count-many";
  "فرنك سويسري",
  "displayName-count-other";
  "فرنك سويسري",
  "symbol";
  "CHF";
}
"CHW";
{
  "displayName";
  "CHW",
  "symbol";
  "CHW";
}
"CLE";
{
  "displayName";
  "CLE",
  "symbol";
}

```

```

        "CLE";
    }
    "CLF";
    {
        "displayName";
        "CLF",
        "symbol";
        "CLF";
    }
    "CLP";
    {
        "displayName";
        "بيزو تشيلي",
        "displayName-count-zero";
        "بيزو تشيلي",
        "displayName-count-one";
        "بيزو تشيلي",
        "displayName-count-two";
        "بيزو تشيلي",
        "displayName-count-few";
        "بيزو تشيلي",
        "displayName-count-many";
        "بيزو تشيلي",
        "displayName-count-other";
        "بيزو تشيلي",
        "symbol";
        "CLP",
        "symbol-alt-narrow";
        "CL$";
    }
    "CNH";
    {
        "displayName";
        "يوان صيني (في الخارج)",
        "displayName-count-zero";
        "يوان صيني (في الخارج)",
        "displayName-count-one";
        "يوان صيني (في الخارج)",
        "displayName-count-two";
        "يوان صيني (في الخارج)",
        "displayName-count-few";
        "يوان صيني (في الخارج)",
        "displayName-count-many";
        "يوان صيني (في الخارج)",
        "displayName-count-other";
        "يوان صيني (في الخارج)",
        "symbol";
        "CNH";
    }
    "CNX";
    {
        "displayName";
        "CNX",
        "symbol";
        "CNX";
    }
    "CNY";

```

```

    {
      "displayName";
      "يوان صيني",
      "displayName-count-zero";
      "يوان صيني",
      "displayName-count-one";
      "يوان صيني",
      "displayName-count-two";
      "يوان صيني",
      "displayName-count-few";
      "يوان صيني",
      "displayName-count-many";
      "يوان صيني",
      "displayName-count-other";
      "يوان صيني",
      "symbol";
      "CN¥",
      "symbol-alt-narrow";
      "CN¥";
    }
    "COP";
    {
      "displayName";
      "بیزو كولومبي",
      "displayName-count-zero";
      "بیزو كولومبي",
      "displayName-count-one";
      "بیزو كولومبي",
      "displayName-count-two";
      "بیزو كولومبي",
      "displayName-count-few";
      "بیزو كولومبي",
      "displayName-count-many";
      "بیزو كولومبي",
      "displayName-count-other";
      "بیزو كولومبي",
      "symbol";
      "COP",
      "symbol-alt-narrow";
      "CO$";
    }
    "COU";
    {
      "displayName";
      "COU",
      "symbol";
      "COU";
    }
    "CRC";
    {
      "displayName";
      "كولن كوستاريكي",
      "displayName-count-zero";
      "كولن كوستاريكي",
      "displayName-count-one";
      "كولن كوستاريكي",
      "displayName-count-two";
    }

```

```

        "كولن كوستاريكي",
        "displayName-count-few";
        "كولن كوستاريكي",
        "displayName-count-many";
        "كولن كوستاريكي",
        "displayName-count-other";
        "كولن كوستاريكي",
        "symbol";
        "CRC",
        "symbol-alt-narrow";
        "¢";
    }
    "CSD";
    {
        "displayName";
        "دينار صربي قديم",
        "symbol";
        "CSD";
    }
    "CSK";
    {
        "displayName";
        "كرونة تشيكوسلوفاكيا",
        "symbol";
        "CSK";
    }
    "CUC";
    {
        "displayName";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-zero";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-one";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-two";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-few";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-many";
        "بيزو كوبي قابل للتحويل",
        "displayName-count-other";
        "بيزو كوبي قابل للتحويل",
        "symbol";
        "CUC",
        "symbol-alt-narrow";
        "$";
    }
    "CUP";
    {
        "displayName";
        "بيزو كوبي",
        "displayName-count-zero";
        "بيزو كوبي",
        "displayName-count-one";
        "بيزو كوبي",
        "displayName-count-two";
        "بيزو كوبي",
    }

```



```

        "displayName-count-few";
        "بيزو كوبي",
        "displayName-count-many";
        "بيزو كوبي",
        "displayName-count-other";
        "بيزو كوبي",
        "symbol";
        "CUP",
        "symbol-alt-narrow";
        "CU$";
    }
    "CVE";
    {
        "displayName";
        "اسكودو الرأس الخضراء",
        "displayName-count-zero";
        "اسكودو الرأس الخضراء",
        "displayName-count-one";
        "اسكودو الرأس الخضراء",
        "displayName-count-two";
        "اسكودو الرأس الخضراء",
        "displayName-count-few";
        "اسكودو الرأس الخضراء",
        "displayName-count-many";
        "اسكودو الرأس الخضراء",
        "displayName-count-other";
        "اسكودو الرأس الخضراء",
        "symbol";
        "CVE";
    }
    "CYP";
    {
        "displayName";
        "جنيه قبرصي",
        "symbol";
        "CYP";
    }
    "CZK";
    {
        "displayName";
        "كرونة تشيكية",
        "displayName-count-zero";
        "كرونة تشيكية",
        "displayName-count-one";
        "كرونة تشيكية",
        "displayName-count-two";
        "كرونة تشيكية",
        "displayName-count-few";
        "كرونة تشيكية",
        "displayName-count-many";
        "كرونة تشيكية",
        "displayName-count-other";
        "كرونة تشيكية",
        "symbol";
        "CZK",
        "symbol-alt-narrow";
        "Kč";
    }

```

```

    }
    "DDM";
    {
        "displayName";
        "أوستمارك ألماني شرقي",
        "symbol";
        "DDM";
    }
    "DEM";
    {
        "displayName";
        "مارك ألماني",
        "symbol";
        "DEM";
    }
    "DJF";
    {
        "displayName";
        "فرنك جيبوتي",
        "displayName-count-zero";
        "فرنك جيبوتي",
        "displayName-count-one";
        "فرنك جيبوتي",
        "displayName-count-two";
        "فرنك جيبوتي",
        "displayName-count-few";
        "فرنك جيبوتي",
        "displayName-count-many";
        "فرنك جيبوتي",
        "displayName-count-other";
        "فرنك جيبوتي",
        "symbol";
        "DJF";
    }
    "DKK";
    {
        "displayName";
        "كرونة دنماركية",
        "displayName-count-zero";
        "كرونة دنماركية",
        "displayName-count-one";
        "كرونة دنماركية",
        "displayName-count-two";
        "كرونة دنماركية",
        "displayName-count-few";
        "كرونة دنماركية",
        "displayName-count-many";
        "كرونة دنماركية",
        "displayName-count-other";
        "كرونة دنماركية",
        "symbol";
        "DKK",
        "symbol-alt-narrow";
        "kr";
    }
    "DOP";
    {

```

```

        "displayName";
        "بیزو الدومنیکان",
        "displayName-count-zero";
        "بیزو الدومنیکان",
        "displayName-count-one";
        "بیزو الدومنیکان",
        "displayName-count-two";
        "بیزو الدومنیکان",
        "displayName-count-few";
        "بیزو الدومنیکان",
        "displayName-count-many";
        "بیزو الدومنیکان",
        "displayName-count-other";
        "بیزو الدومنیکان",
        "symbol";
        "DOP",
        "symbol-alt-narrow";
        "DO$";
    }
    "DZD";
    {
        "displayName";
        "دینار جزائری",
        "displayName-count-zero";
        "دینار جزائری",
        "displayName-count-one";
        "دینار جزائری",
        "displayName-count-two";
        "دیناران جزائریان",
        "displayName-count-few";
        "دینارات جزائریة",
        "displayName-count-many";
        "دینارًا جزائریًا",
        "displayName-count-other";
        "دینار جزائری",
        "symbol";
        "د.ج.";
    }
    "ECS";
    {
        "displayName";
        "ECS",
        "symbol";
        "ECS";
    }
    "ECV";
    {
        "displayName";
        "ECV",
        "symbol";
        "ECV";
    }
    "EEK";
    {
        "displayName";
        "کرونة استونیة",
        "symbol";
    }

```

```

        "EEK";
    }
    "EGP";
    {
        "displayName";
        "جنيه مصري",
        "displayName-count-zero";
        "جنيه مصري",
        "displayName-count-one";
        "جنيه مصري",
        "displayName-count-two";
        "جنيهان مصريان",
        "displayName-count-few";
        "جنيهات مصرية",
        "displayName-count-many";
        "جنيهاً مصرياً",
        "displayName-count-other";
        "جنيه مصري",
        "symbol";
        "ج.م.٠",
        "symbol-alt-narrow";
        "£E";
    }
    "ERN";
    {
        "displayName";
        "ناكفا أريتري",
        "displayName-count-zero";
        "ناكفا أريتري",
        "displayName-count-one";
        "ناكفا أريتري",
        "displayName-count-two";
        "ناكفا أريتري",
        "displayName-count-few";
        "ناكفا أريتري",
        "displayName-count-many";
        "ناكفا أريتري",
        "displayName-count-other";
        "ناكفا أريتري",
        "symbol";
        "ERN";
    }
    "ESA";
    {
        "displayName";
        "ESA",
        "symbol";
        "ESA";
    }
    "ESB";
    {
        "displayName";
        "ESB",
        "symbol";
        "ESB";
    }
    "ESP";

```

```

    {
      "displayName";
      "بیزیتا إسبانی",
      "symbol";
      "ESP",
      "symbol-alt-narrow";
      "₪";
    }
    "ETB";
    {
      "displayName";
      "بیر أثیوبی",
      "displayName-count-zero";
      "بیر أثیوبی",
      "displayName-count-one";
      "بیر أثیوبی",
      "displayName-count-two";
      "بیر أثیوبی",
      "displayName-count-few";
      "بیر أثیوبی",
      "displayName-count-many";
      "بیر أثیوبی",
      "displayName-count-other";
      "بیر أثیوبی",
      "symbol";
      "ETB";
    }
    "EUR";
    {
      "displayName";
      "یورو",
      "displayName-count-zero";
      "یورو",
      "displayName-count-one";
      "یورو",
      "displayName-count-two";
      "یورو",
      "displayName-count-few";
      "یورو",
      "displayName-count-many";
      "یورو",
      "displayName-count-other";
      "یورو",
      "symbol";
      "€",
      "symbol-alt-narrow";
      "€";
    }
    "FIM";
    {
      "displayName";
      "مارکا فنلندی",
      "symbol";
      "FIM";
    }
    "FJD";
    {

```

```

        "displayName";
        "دولار فيجي",
        "displayName-count-zero";
        "دولار فيجي",
        "displayName-count-one";
        "دولار فيجي",
        "displayName-count-two";
        "دولار فيجي",
        "displayName-count-few";
        "دولار فيجي",
        "displayName-count-many";
        "دولار فيجي",
        "displayName-count-other";
        "دولار فيجي",
        "symbol";
        "FJD",
        "symbol-alt-narrow";
        "FJ$";
    }
    "FKP";
    {
        "displayName";
        "جنيه جزر فوكلاند",
        "displayName-count-zero";
        "جنيه جزر فوكلاند",
        "displayName-count-one";
        "جنيه جزر فوكلاند",
        "displayName-count-two";
        "جنيه جزر فوكلاند",
        "displayName-count-few";
        "جنيه جزر فوكلاند",
        "displayName-count-many";
        "جنيه جزر فوكلاند",
        "displayName-count-other";
        "جنيه جزر فوكلاند",
        "symbol";
        "FKP",
        "symbol-alt-narrow";
        "£";
    }
    "FRF";
    {
        "displayName";
        "فرنك فرنسي",
        "symbol";
        "FRF";
    }
    "GBP";
    {
        "displayName";
        "جنيه إسترليني",
        "displayName-count-zero";
        "جنيه إسترليني",
        "displayName-count-one";
        "جنيه إسترليني",
        "displayName-count-two";
        "جنيه إسترليني",
    }

```

```

        "displayName-count-few";
        "جنيه إسترليني",
        "displayName-count-many";
        "جنيه إسترليني",
        "displayName-count-other";
        "جنيه إسترليني",
        "symbol";
        "£",
        "symbol-alt-narrow";
        "UK£";
    }
    "GEK";
    {
        "displayName";
        "GEK",
        "symbol";
        "GEK";
    }
    "GEL";
    {
        "displayName";
        "ლარი جورجი",
        "displayName-count-zero";
        "ლარი جورجი",
        "displayName-count-one";
        "ლარი جورجი",
        "displayName-count-two";
        "ლარი جورجი",
        "displayName-count-few";
        "ლარი جورجი",
        "displayName-count-many";
        "ლარი جورجი",
        "displayName-count-other";
        "ლარი جورجი",
        "symbol";
        "GEL",
        "symbol-alt-narrow";
        "ლ",
        "symbol-alt-variant";
        "ლ";
    }
    "GHC";
    {
        "displayName";
        "სიდი განი",
        "symbol";
        "GHC";
    }
    "GHS";
    {
        "displayName";
        "სიდი განა",
        "displayName-count-zero";
        "სიდი განა",
        "displayName-count-one";
        "სიდი განა",
        "displayName-count-two";
    }

```

```

        "سيدي غانا",
        "displayName-count-few";
        "سيدي غانا",
        "displayName-count-many";
        "سيدي غانا",
        "displayName-count-other";
        "سيدي غانا",
        "symbol";
        "GHS";
    }
    "GIP";
    {
        "displayName";
        "جنيه جبل طارق",
        "displayName-count-zero";
        "جنيه جبل طارق",
        "displayName-count-one";
        "جنيه جبل طارق",
        "displayName-count-two";
        "جنيه جبل طارق",
        "displayName-count-few";
        "جنيه جبل طارق",
        "displayName-count-many";
        "جنيه جبل طارق",
        "displayName-count-other";
        "جنيه جبل طارق",
        "symbol";
        "GIP",
        "symbol-alt-narrow";
        "£";
    }
    "GMD";
    {
        "displayName";
        "دلاسي غامبي",
        "displayName-count-zero";
        "دلاسي غامبي",
        "displayName-count-one";
        "دلاسي غامبي",
        "displayName-count-two";
        "دلاسي غامبي",
        "displayName-count-few";
        "دلاسي غامبي",
        "displayName-count-many";
        "دلاسي غامبي",
        "displayName-count-other";
        "دلاسي غامبي",
        "symbol";
        "GMD";
    }
    "GNF";
    {
        "displayName";
        "فرنك غينيا",
        "displayName-count-zero";
        "فرنك غينيا",
        "displayName-count-one";
    }

```



```

        "فرنك غينيا",
        "displayName-count-two";
        "فرنك غينيا",
        "displayName-count-few";
        "فرنك غينيا",
        "displayName-count-many";
        "فرنك غينيا",
        "displayName-count-other";
        "فرنك غينيا",
        "symbol";
        "GNF",
        "symbol-alt-narrow";
        "FG";
    }
    "GNS";
    {
        "displayName";
        "سيلي غينيا",
        "symbol";
        "GNS";
    }
    "GQE";
    {
        "displayName";
        "اكويل جونيئا غينيا الاستوائية",
        "symbol";
        "GQE";
    }
    "GRD";
    {
        "displayName";
        "دراخما يوناني",
        "symbol";
        "GRD";
    }
    "GTQ";
    {
        "displayName";
        "كوتزال غواتيمالا",
        "displayName-count-zero";
        "كوتزال غواتيمالا",
        "displayName-count-one";
        "كوتزال غواتيمالا",
        "displayName-count-two";
        "كوتزال غواتيمالا",
        "displayName-count-few";
        "كوتزال غواتيمالا",
        "displayName-count-many";
        "كوتزال غواتيمالا",
        "displayName-count-other";
        "كوتزال غواتيمالا",
        "symbol";
        "GTQ",
        "symbol-alt-narrow";
        "Q";
    }
    "GWE";

```

```

    {
      "displayName";
      "اسکود برتغالی غینیا",
      "symbol";
      "GWE";
    }
    "GWP";
    {
      "displayName";
      "بیزو غینیا بیساو",
      "symbol";
      "GWP";
    }
    "GYD";
    {
      "displayName";
      "دولار غيانا",
      "displayName-count-zero";
      "دولار غيانا",
      "displayName-count-one";
      "دولار غيانا",
      "displayName-count-two";
      "دولار غيانا",
      "displayName-count-few";
      "دولار غيانا",
      "displayName-count-many";
      "دولار غيانا",
      "displayName-count-other";
      "دولار غيانا",
      "symbol";
      "GYD",
      "symbol-alt-narrow";
      "GY$";
    }
    "HKD";
    {
      "displayName";
      "دولار هونغ كونغ",
      "displayName-count-zero";
      "دولار هونغ كونغ",
      "displayName-count-one";
      "دولار هونغ كونغ",
      "displayName-count-two";
      "دولار هونغ كونغ",
      "displayName-count-few";
      "دولار هونغ كونغ",
      "displayName-count-many";
      "دولار هونغ كونغ",
      "displayName-count-other";
      "دولار هونغ كونغ",
      "symbol";
      "HK$",
      "symbol-alt-narrow";
      "HK$";
    }
    "HNL";
    {

```

```

        "displayName";
        "ليمبيرا هندوراس",
        "displayName-count-zero";
        "ليمبيرا هندوراس",
        "displayName-count-one";
        "ليمبيرا هندوراس",
        "displayName-count-two";
        "ليمبيرا هندوراس",
        "displayName-count-few";
        "ليمبيرا هندوراس",
        "displayName-count-many";
        "ليمبيرا هندوراس",
        "displayName-count-other";
        "ليمبيرا هندوراس",
        "symbol";
        "HNL",
        "symbol-alt-narrow";
        "L";
    }
    "HRD";
    {
        "displayName";
        "دينار كرواتي",
        "symbol";
        "HRD";
    }
    "HRK";
    {
        "displayName";
        "كونا كرواتي",
        "displayName-count-zero";
        "كونا كرواتي",
        "displayName-count-one";
        "كونا كرواتي",
        "displayName-count-two";
        "كونا كرواتي",
        "displayName-count-few";
        "كونا كرواتي",
        "displayName-count-many";
        "كونا كرواتي",
        "displayName-count-other";
        "كونا كرواتي",
        "symbol";
        "HRK",
        "symbol-alt-narrow";
        "kn";
    }
    "HTG";
    {
        "displayName";
        "جوردي هايتي",
        "displayName-count-zero";
        "جوردي هايتي",
        "displayName-count-one";
        "جوردي هايتي",
        "displayName-count-two";
        "جوردي هايتي",
    }

```

```

        "displayName-count-few";
        "جوردی هایتي",
        "displayName-count-many";
        "جوردی هایتي",
        "displayName-count-other";
        "جوردی هایتي",
        "symbol";
        "HTG";
    }
    "HUF";
    {
        "displayName";
        "فورينت هنگاري",
        "displayName-count-zero";
        "فورينت هنگاري",
        "displayName-count-one";
        "فورينت هنگاري",
        "displayName-count-two";
        "فورينت هنگاري",
        "displayName-count-few";
        "فورينت هنگاري",
        "displayName-count-many";
        "فورينت هنگاري",
        "displayName-count-other";
        "فورينت هنگاري",
        "symbol";
        "HUF",
        "symbol-alt-narrow";
        "Ft";
    }
    "IDR";
    {
        "displayName";
        "روبية إندونيسية",
        "displayName-count-zero";
        "روبية إندونيسية",
        "displayName-count-one";
        "روبية إندونيسية",
        "displayName-count-two";
        "روبية إندونيسية",
        "displayName-count-few";
        "روبية إندونيسية",
        "displayName-count-many";
        "روبية إندونيسية",
        "displayName-count-other";
        "روبية إندونيسية",
        "symbol";
        "IDR",
        "symbol-alt-narrow";
        "Rp";
    }
    "IEP";
    {
        "displayName";
        "جنيه إيرلندي",
        "symbol";
        "IEP";
    }

```

```

    }
    "ILP";
    {
        "displayName";
        "جنيه إسرائيلي",
        "symbol";
        "ILP";
    }
    "ILR";
    {
        "displayName";
        "ILR",
        "symbol";
        "ILR";
    }
    "ILS";
    {
        "displayName";
        "ש"יכל ישראלי جديد",
        "displayName-count-zero";
        "ש"יכל ישראלי جديد",
        "displayName-count-one";
        "ש"יכל ישראלי جديد",
        "displayName-count-two";
        "ש"יכל ישראלי جديد",
        "displayName-count-few";
        "ש"יכל ישראלי جديد",
        "displayName-count-many";
        "ש"יכל ישראלי جديد",
        "displayName-count-other";
        "ש"יכל ישראלי جديد",
        "symbol";
        "₪",
        "symbol-alt-narrow";
        "₪";
    }
    "INR";
    {
        "displayName";
        "روبية هندي",
        "displayName-count-zero";
        "روبية هندي",
        "displayName-count-one";
        "روبية هندي",
        "displayName-count-two";
        "روبية هندي",
        "displayName-count-few";
        "روبية هندي",
        "displayName-count-many";
        "روبية هندي",
        "displayName-count-other";
        "روبية هندي",
        "symbol";
        "₹",
        "symbol-alt-narrow";
        "₹";
    }
}

```

```

"IQD";
{
  "displayName";
  "دينار عراقي",
  "displayName-count-zero";
  "دينار عراقي",
  "displayName-count-one";
  "دينار عراقي",
  "displayName-count-two";
  "دينار عراقي",
  "displayName-count-few";
  "دينار عراقي",
  "displayName-count-many";
  "دينار عراقي",
  "displayName-count-other";
  "دينار عراقي",
  "symbol";
  "د.ع.";
}
"IRR";
{
  "displayName";
  "ریال ایرانی",
  "displayName-count-zero";
  "ریال ایرانی",
  "displayName-count-one";
  "ریال ایرانی",
  "displayName-count-two";
  "ریال ایرانی",
  "displayName-count-few";
  "ریال ایرانی",
  "displayName-count-many";
  "ریال ایرانی",
  "displayName-count-other";
  "ریال ایرانی",
  "symbol";
  "ر.ا.";
}
"ISJ";
{
  "displayName";
  "ISJ",
  "symbol";
  "ISJ";
}
"ISK";
{
  "displayName";
  "كرونه أيسلندية",
  "displayName-count-zero";
  "كرونه أيسلندية",
  "displayName-count-one";
  "كرونه أيسلندية",
  "displayName-count-two";
  "كرونه أيسلندية",
  "displayName-count-few";
  "كرونه أيسلندية";
}

```

```

        "displayName-count-many";
        "كرونة أيسلندية",
        "displayName-count-other";
        "كرونة أيسلندية",
        "symbol";
        "ISK",
        "symbol-alt-narrow";
        "kr";
    }
    "ITL";
    {
        "displayName";
        "ليرة إيطالية",
        "symbol";
        "ITL";
    }
    "JMD";
    {
        "displayName";
        "دولار جامايكي",
        "displayName-count-zero";
        "دولار جامايكي",
        "displayName-count-one";
        "دولار جامايكي",
        "displayName-count-two";
        "دولار جامايكي",
        "displayName-count-few";
        "دولار جامايكي",
        "displayName-count-many";
        "دولار جامايكي",
        "displayName-count-other";
        "دولار جامايكي",
        "symbol";
        "JMD",
        "symbol-alt-narrow";
        "JM$";
    }
    "JOD";
    {
        "displayName";
        "دينار أردني",
        "displayName-count-zero";
        "دينار أردني",
        "displayName-count-one";
        "دينار أردني",
        "displayName-count-two";
        "دينار أردني",
        "displayName-count-few";
        "دينار أردني",
        "displayName-count-many";
        "دينار أردني",
        "displayName-count-other";
        "دينار أردني",
        "symbol";
        "د.أ.";
    }
    "JPY";

```

```

    {
      "displayName";
      "ين ياباني",
      "displayName-count-zero";
      "ين ياباني",
      "displayName-count-one";
      "ين ياباني",
      "displayName-count-two";
      "ين ياباني",
      "displayName-count-few";
      "ين ياباني",
      "displayName-count-many";
      "ين ياباني",
      "displayName-count-other";
      "ين ياباني",
      "symbol";
      "JP¥",
      "symbol-alt-narrow";
      "JP¥";
    }
    "KES";
    {
      "displayName";
      "شلن كينيي",
      "displayName-count-zero";
      "شلن كينيي",
      "displayName-count-one";
      "شلن كينيي",
      "displayName-count-two";
      "شلن كينيي",
      "displayName-count-few";
      "شلن كينيي",
      "displayName-count-many";
      "شلن كينيي",
      "displayName-count-other";
      "شلن كينيي",
      "symbol";
      "KES";
    }
    "KGS";
    {
      "displayName";
      "سوم قيرغستاني",
      "displayName-count-zero";
      "سوم قيرغستاني",
      "displayName-count-one";
      "سوم قيرغستاني",
      "displayName-count-two";
      "سوم قيرغستاني",
      "displayName-count-few";
      "سوم قيرغستاني",
      "displayName-count-many";
      "سوم قيرغستاني",
      "displayName-count-other";
      "سوم قيرغستاني",
      "symbol";
      "KGS";
    }
  }

```



```

    }
    "KHR";
    {
        "displayName";
        "ریال کمبودی",
        "displayName-count-zero";
        "ریال کمبودی",
        "displayName-count-one";
        "ریال کمبودی",
        "displayName-count-two";
        "ریال کمبودی",
        "displayName-count-few";
        "ریال کمبودی",
        "displayName-count-many";
        "ریال کمبودی",
        "displayName-count-other";
        "ریال کمبودی",
        "symbol";
        "KHR",
        "symbol-alt-narrow";
        "₭";
    }
    "KMF";
    {
        "displayName";
        "فرنك جزر القمر",
        "displayName-count-zero";
        "فرنك جزر القمر",
        "displayName-count-one";
        "فرنك جزر القمر",
        "displayName-count-two";
        "فرنك جزر القمر",
        "displayName-count-few";
        "فرنك جزر القمر",
        "displayName-count-many";
        "فرنك جزر القمر",
        "displayName-count-other";
        "فرنك جزر القمر",
        "symbol";
        "KMF",
        "symbol-alt-narrow";
        "CF";
    }
    "KPW";
    {
        "displayName";
        "وون كوريا الشمالية",
        "displayName-count-zero";
        "وون كوريا الشمالية",
        "displayName-count-one";
        "وون كوريا الشمالية",
        "displayName-count-two";
        "وون كوريا الشمالية",
        "displayName-count-few";
        "وون كوريا الشمالية",
        "displayName-count-many";
    }

```

```

        "وون كوريا الشمالية",
        "displayName-count-other";
        "وون كوريا الشمالية",
        "symbol";
        "KPW",
        "symbol-alt-narrow";
        "₩";
    }
    "KRH";
    {
        "displayName";
        "KRH",
        "symbol";
        "KRH";
    }
    "KRO";
    {
        "displayName";
        "KRO",
        "symbol";
        "KRO";
    }
    "KRW";
    {
        "displayName";
        "وون كوريا الجنوبية",
        "displayName-count-zero";
        "وون كوريا الجنوبية",
        "displayName-count-one";
        "وون كوريا الجنوبية",
        "displayName-count-two";
        "وون كوريا الجنوبية",
        "displayName-count-few";
        "وون كوريا الجنوبية",
        "displayName-count-many";
        "وون كوريا الجنوبية",
        "displayName-count-other";
        "وون كوريا الجنوبية",
        "symbol";
        "₩",
        "symbol-alt-narrow";
        "₩";
    }
    "KWD";
    {
        "displayName";
        "دينار كويتي",
        "displayName-count-zero";
        "دينار كويتي",
        "displayName-count-one";
        "دينار كويتي",
        "displayName-count-two";
        "دينار كويتي",
        "displayName-count-few";
        "دينار كويتي",
        "displayName-count-many";
        "دينار كويتي",
    }

```

```

        "displayName-count-other";
        "دينار كويتي",
        "symbol";
        "د.ك.";
    }
    "KYD";
    {
        "displayName";
        "دولار جزر كيمن",
        "displayName-count-zero";
        "دولار جزر كيمن",
        "displayName-count-one";
        "دولار جزر كيمن",
        "displayName-count-two";
        "دولار جزر كيمن",
        "displayName-count-few";
        "دولار جزر كيمن",
        "displayName-count-many";
        "دولار جزر كيمن",
        "displayName-count-other";
        "دولار جزر كيمن",
        "symbol";
        "KYD",
        "symbol-alt-narrow";
        "KY$";
    }
    "KZT";
    {
        "displayName";
        "تينغ كازاخستاني",
        "displayName-count-zero";
        "تينغ كازاخستاني",
        "displayName-count-one";
        "تينغ كازاخستاني",
        "displayName-count-two";
        "تينغ كازاخستاني",
        "displayName-count-few";
        "تينغ كازاخستاني",
        "displayName-count-many";
        "تينغ كازاخستاني",
        "displayName-count-other";
        "تينغ كازاخستاني",
        "symbol";
        "KZT",
        "symbol-alt-narrow";
        "₸";
    }
    "LAK";
    {
        "displayName";
        "كيب لاوسي",
        "displayName-count-zero";
        "كيب لاوسي",
        "displayName-count-one";
        "كيب لاوسي",
        "displayName-count-two";
        "كيب لاوسي",
    }

```

```

        "displayName-count-few";
        "كيب لاوسي",
        "displayName-count-many";
        "كيب لاوسي",
        "displayName-count-other";
        "كيب لاوسي",
        "symbol";
        "LAK",
        "symbol-alt-narrow";
        "₭";
    }
    "LBP";
    {
        "displayName";
        "جنيه لبناني",
        "displayName-count-zero";
        "جنيه لبناني",
        "displayName-count-one";
        "جنيه لبناني",
        "displayName-count-two";
        "جنيه لبناني",
        "displayName-count-few";
        "جنيه لبناني",
        "displayName-count-many";
        "جنيه لبناني",
        "displayName-count-other";
        "جنيه لبناني",
        "symbol";
        "ل.ل.",
        "symbol-alt-narrow";
        "L£";
    }
    "LKR";
    {
        "displayName";
        "روبية سريلانكية",
        "displayName-count-zero";
        "روبية سريلانكية",
        "displayName-count-one";
        "روبية سريلانكية",
        "displayName-count-two";
        "روبية سريلانكية",
        "displayName-count-few";
        "روبية سريلانكية",
        "displayName-count-many";
        "روبية سريلانكية",
        "displayName-count-other";
        "روبية سريلانكية",
        "symbol";
        "LKR",
        "symbol-alt-narrow";
        "Rs";
    }
    "LRD";
    {
        "displayName";
        "دولار ليبيري",

```

```

        "displayName-count-zero";
        "دولار ليبيري",
        "displayName-count-one";
        "دولار ليبيري",
        "displayName-count-two";
        "دولاران ليبيريان",
        "displayName-count-few";
        "دولارات ليبيرية",
        "displayName-count-many";
        "دولارًا ليبيريًا",
        "displayName-count-other";
        "دولار ليبيري",
        "symbol";
        "LRD",
        "symbol-alt-narrow";
        "$";
    }
    "LSL";
    {
        "displayName";
        "لوتي ليسوتو",
        "symbol";
        "LSL";
    }
    "LTL";
    {
        "displayName";
        "ليتاليتوانية",
        "displayName-count-zero";
        "ليتاليتوانية",
        "displayName-count-one";
        "ليتاليتوانية",
        "displayName-count-two";
        "ليتاليتوانية",
        "displayName-count-few";
        "ليتاليتوانية",
        "displayName-count-many";
        "ليتاليتوانية",
        "displayName-count-other";
        "ليتاليتوانية",
        "symbol";
        "LTL",
        "symbol-alt-narrow";
        "Lt";
    }
    "LTT";
    {
        "displayName";
        "تالوناس ليتواني",
        "symbol";
        "LTT";
    }
    "LUC";
    {
        "displayName";
        "فرنك لوكسمبرج قابل للتحويل",
        "symbol";
    }

```

```

        "LUC";
    }
    "LUF";
    {
        "displayName";
        "فرنك لوكسمبرج",
        "symbol";
        "LUF";
    }
    "LUL";
    {
        "displayName";
        "فرنك لوكسمبرج المالي",
        "symbol";
        "LUL";
    }
    "LVL";
    {
        "displayName";
        "لاتس لاتفيا",
        "displayName-count-zero";
        "لاتس لاتفي",
        "displayName-count-one";
        "لاتس لاتفي",
        "displayName-count-two";
        "لاتس لاتفي",
        "displayName-count-few";
        "لاتس لاتفي",
        "displayName-count-many";
        "لاتس لاتفي",
        "displayName-count-other";
        "لاتس لاتفي",
        "symbol";
        "LVL",
        "symbol-alt-narrow";
        "Ls";
    }
    "LVR";
    {
        "displayName";
        "روبل لاتفيا",
        "symbol";
        "LVR";
    }
    "LYD";
    {
        "displayName";
        "دينار ليبي",
        "displayName-count-zero";
        "دينار ليبي",
        "displayName-count-one";
        "دينار ليبي",
        "displayName-count-two";
        "ديناران ليبيان",
        "displayName-count-few";
        "دينارات ليبية",
        "displayName-count-many";
    }

```

```

        "دينارًا ليبيا",
        "displayName-count-other";
        "دينار ليبي",
        "symbol";
        "د.ل.";
    }
    "MAD";
    {
        "displayName";
        "درهم مغربي",
        "displayName-count-zero";
        "درهم مغربي",
        "displayName-count-one";
        "درهم مغربي",
        "displayName-count-two";
        "درهمان مغربيان",
        "displayName-count-few";
        "دراهم مغربية",
        "displayName-count-many";
        "درهما مغربيا",
        "displayName-count-other";
        "درهم مغربي",
        "symbol";
        "د.م.";
    }
    "MAF";
    {
        "displayName";
        "فرنك مغربي",
        "symbol";
        "MAF";
    }
    "MCF";
    {
        "displayName";
        "MCF",
        "symbol";
        "MCF";
    }
    "MDC";
    {
        "displayName";
        "MDC",
        "symbol";
        "MDC";
    }
    "MDL";
    {
        "displayName";
        "ليو مولدوفي",
        "displayName-count-zero";
        "ليو مولدوفي",
        "displayName-count-one";
        "ليو مولدوفي",
        "displayName-count-two";
        "ليو مولدوفي",
        "displayName-count-few";
    }

```

```

        "ليو مولدوفي",
        "displayName-count-many";
        "ليو مولدوفي",
        "displayName-count-other";
        "ليو مولدوفي",
        "symbol";
        "MDL";
    }
    "MGA";
    {
        "displayName";
        "أرياري مدغشقر",
        "displayName-count-zero";
        "أرياري مدغشقر",
        "displayName-count-one";
        "أرياري مدغشقر",
        "displayName-count-two";
        "أرياري مدغشقر",
        "displayName-count-few";
        "أرياري مدغشقر",
        "displayName-count-many";
        "أرياري مدغشقر",
        "displayName-count-other";
        "أرياري مدغشقر",
        "symbol";
        "MGA",
        "symbol-alt-narrow";
        "Ar";
    }
    "MGF";
    {
        "displayName";
        "فرنك مدغشقر",
        "symbol";
        "MGF";
    }
    "MKD";
    {
        "displayName";
        "دينار مقدوني",
        "displayName-count-zero";
        "دينار مقدوني",
        "displayName-count-one";
        "دينار مقدوني",
        "displayName-count-two";
        "ديناران مقدونيان",
        "displayName-count-few";
        "دينارات مقدونية",
        "displayName-count-many";
        "دينارًا مقدونيا",
        "displayName-count-other";
        "دينار مقدوني",
        "symbol";
        "MKD";
    }
    "MKN";
    {

```



```

        "displayName";
        "MKN",
        "symbol";
        "MKN";
    }
    "MLF";
    {
        "displayName";
        "فرنك مالي",
        "symbol";
        "MLF";
    }
    "MMK";
    {
        "displayName";
        "كيات ميانمار",
        "displayName-count-zero";
        "كيات ميانمار",
        "displayName-count-one";
        "كيات ميانمار",
        "displayName-count-two";
        "كيات ميانمار",
        "displayName-count-few";
        "كيات ميانمار",
        "displayName-count-many";
        "كيات ميانمار",
        "displayName-count-other";
        "كيات ميانمار",
        "symbol";
        "MMK",
        "symbol-alt-narrow";
        "K";
    }
    "MNT";
    {
        "displayName";
        "توغروغ منغولي",
        "displayName-count-zero";
        "توغروغ منغولي",
        "displayName-count-one";
        "توغروغ منغولي",
        "displayName-count-two";
        "توغروغ منغولي",
        "displayName-count-few";
        "توغروغ منغولي",
        "displayName-count-many";
        "توغروغ منغولي",
        "displayName-count-other";
        "توغروغ منغولي",
        "symbol";
        "MNT",
        "symbol-alt-narrow";
        "₮";
    }
    "MOP";
    {
        "displayName";

```

```

        "باتاكا ماكاوي",
        "displayName-count-zero";
        "باتاكا ماكاوي",
        "displayName-count-one";
        "باتاكا ماكاوي",
        "displayName-count-two";
        "باتاكا ماكاوي",
        "displayName-count-few";
        "باتاكا ماكاوي",
        "displayName-count-many";
        "باتاكا ماكاوي",
        "displayName-count-other";
        "باتاكا ماكاوي",
        "symbol";
        "MOP";
    }
    "MRO";
    {
        "displayName";
        "أوقية موريتانية",
        "displayName-count-zero";
        "أوقية موريتانية",
        "displayName-count-one";
        "أوقية موريتانية",
        "displayName-count-two";
        "أوقية موريتانية",
        "displayName-count-few";
        "أوقية موريتانية",
        "displayName-count-many";
        "أوقية موريتانية",
        "displayName-count-other";
        "أوقية موريتانية",
        "symbol";
        "أ.م.";
    }
    "MTL";
    {
        "displayName";
        "ليرة مالطية",
        "symbol";
        "MTL";
    }
    "MTP";
    {
        "displayName";
        "جنيه مالطي",
        "symbol";
        "MTP";
    }
    "MUR";
    {
        "displayName";
        "روبية موريشيوسية",
        "displayName-count-zero";
        "روبية موريشيوسية",
        "displayName-count-one";
        "روبية موريشيوسية",

```

```

        "displayName-count-two";
        "روبية موريشيوسية",
        "displayName-count-few";
        "روبية موريشيوسية",
        "displayName-count-many";
        "روبية موريشيوسية",
        "displayName-count-other";
        "روبية موريشيوسية",
        "symbol";
        "MUR",
        "symbol-alt-narrow";
        "Rs";
    }
    "MVP";
    {
        "displayName";
        "MVP",
        "symbol";
        "MVP";
    }
    "MVR";
    {
        "displayName";
        "روفيه جزر المالديف",
        "displayName-count-zero";
        "روفيه جزر المالديف",
        "displayName-count-one";
        "روفيه جزر المالديف",
        "displayName-count-two";
        "روفيه جزر المالديف",
        "displayName-count-few";
        "روفيه جزر المالديف",
        "displayName-count-many";
        "روفيه جزر المالديف",
        "displayName-count-other";
        "روفيه جزر المالديف",
        "symbol";
        "MVR";
    }
    "MWK";
    {
        "displayName";
        "كواشا ملاوي",
        "displayName-count-zero";
        "كواشا ملاوي",
        "displayName-count-one";
        "كواشا ملاوي",
        "displayName-count-two";
        "كواشا ملاوي",
        "displayName-count-few";
        "كواشا ملاوي",
        "displayName-count-many";
        "كواشا ملاوي",
        "displayName-count-other";
        "كواشا ملاوي",
        "symbol";
        "MWK";
    }

```

```

    }
    "MXN";
    {
      "displayName";
      "بيزو مكسيكي",
      "displayName-count-zero";
      "بيزو مكسيكي",
      "displayName-count-one";
      "بيزو مكسيكي",
      "displayName-count-two";
      "بيزو مكسيكي",
      "displayName-count-few";
      "بيزو مكسيكي",
      "displayName-count-many";
      "بيزو مكسيكي",
      "displayName-count-other";
      "بيزو مكسيكي",
      "symbol";
      "MX$";
      "symbol-alt-narrow";
      "MX$";
    }
    "MXP";
    {
      "displayName";
      "بيزو فضي مكسيكي - 1861-1992",
      "symbol";
      "MXP";
    }
    "MXV";
    {
      "displayName";
      "MXV",
      "symbol";
      "MXV";
    }
    "MYR";
    {
      "displayName";
      "رينغيت ماليزي",
      "displayName-count-zero";
      "رينغيت ماليزي",
      "displayName-count-one";
      "رينغيت ماليزي",
      "displayName-count-two";
      "رينغيت ماليزي",
      "displayName-count-few";
      "رينغيت ماليزي",
      "displayName-count-many";
      "رينغيت ماليزي",
      "displayName-count-other";
      "رينغيت ماليزي",
      "symbol";
      "MYR",
      "symbol-alt-narrow";
      "RM";
    }
  }

```

```

"MZE";
{
  "displayName";
  "اسكود موزمبيقي",
  "symbol";
  "MZE";
}
"MZM";
{
  "displayName";
  "MZM",
  "symbol";
  "MZM";
}
"MZN";
{
  "displayName";
  "متكال موزمبيقي",
  "displayName-count-zero";
  "متكال موزمبيقي",
  "displayName-count-one";
  "متكال موزمبيقي",
  "displayName-count-two";
  "متكال موزمبيقي",
  "displayName-count-few";
  "متكال موزمبيقي",
  "displayName-count-many";
  "متكال موزمبيقي",
  "displayName-count-other";
  "متكال موزمبيقي",
  "symbol";
  "MZN";
}
"NAD";
{
  "displayName";
  "دولار ناميبي",
  "displayName-count-zero";
  "دولار ناميبي",
  "displayName-count-one";
  "دولار ناميبي",
  "displayName-count-two";
  "دولار ناميبي",
  "displayName-count-few";
  "دولار ناميبي",
  "displayName-count-many";
  "دولار ناميبي",
  "displayName-count-other";
  "دولار ناميبي",
  "symbol";
  "NAD",
  "symbol-alt-narrow";
  "$";
}
"NGN";
{
  "displayName";

```

```

        "نايرا نيجيري",
        "displayName-count-zero";
        "نايرا نيجيري",
        "displayName-count-one";
        "نايرا نيجيري",
        "displayName-count-two";
        "نايرا نيجيري",
        "displayName-count-few";
        "نايرا نيجيري",
        "displayName-count-many";
        "نايرا نيجيري",
        "displayName-count-other";
        "نايرا نيجيري",
        "symbol";
        "NGN",
        "symbol-alt-narrow";
        "₦";
    }
    "NIC";
    {
        "displayName";
        "كوردوبه نيكاراغوا",
        "symbol";
        "NIC";
    }
    "NIO";
    {
        "displayName";
        "قرطبة نيكاراغوا",
        "displayName-count-zero";
        "قرطبة نيكاراغوا",
        "displayName-count-one";
        "قرطبة نيكاراغوا",
        "displayName-count-two";
        "قرطبة نيكاراغوا",
        "displayName-count-few";
        "قرطبة نيكاراغوا",
        "displayName-count-many";
        "قرطبة نيكاراغوا",
        "displayName-count-other";
        "قرطبة نيكاراغوا",
        "symbol";
        "NIO",
        "symbol-alt-narrow";
        "C$";
    }
    "NLG";
    {
        "displayName";
        "جلدر هولندي",
        "symbol";
        "NLG";
    }
    "NOK";
    {
        "displayName";
        "كرونة نرويجية",

```

```

        "displayName-count-zero";
        "كرونة نرويجية",
        "displayName-count-one";
        "كرونة نرويجية",
        "displayName-count-two";
        "كرونة نرويجية",
        "displayName-count-few";
        "كرونة نرويجية",
        "displayName-count-many";
        "كرونة نرويجية",
        "displayName-count-other";
        "كرونة نرويجية",
        "symbol";
        "NOK",
        "symbol-alt-narrow";
        "kr";
    }
    "NPR";
    {
        "displayName";
        "روبية نيبالي",
        "displayName-count-zero";
        "روبية نيبالي",
        "displayName-count-one";
        "روبية نيبالي",
        "displayName-count-two";
        "روبية نيبالي",
        "displayName-count-few";
        "روبية نيبالي",
        "displayName-count-many";
        "روبية نيبالي",
        "displayName-count-other";
        "روبية نيبالي",
        "symbol";
        "NPR",
        "symbol-alt-narrow";
        "Rs";
    }
    "NZD";
    {
        "displayName";
        "دولار نيوزيلندي",
        "displayName-count-zero";
        "دولار نيوزيلندي",
        "displayName-count-one";
        "دولار نيوزيلندي",
        "displayName-count-two";
        "دولار نيوزيلندي",
        "displayName-count-few";
        "دولار نيوزيلندي",
        "displayName-count-many";
        "دولار نيوزيلندي",
        "displayName-count-other";
        "دولار نيوزيلندي",
        "symbol";
        "NZ$",
        "symbol-alt-narrow";
    }

```

```

        "NZ$";
    }
    "OMR";
    {
        "displayName";
        "ول عماني",
        "displayName-count-zero";
        "ول عماني",
        "displayName-count-one";
        "ول عماني",
        "displayName-count-two";
        "ول عماني",
        "displayName-count-few";
        "ول عماني",
        "displayName-count-many";
        "ول عماني",
        "displayName-count-other";
        "ول عماني",
        "symbol";
        "ر.ع.";
    }
    "PAB";
    {
        "displayName";
        "بالبوا بنمي",
        "displayName-count-zero";
        "بالبوا بنمي",
        "displayName-count-one";
        "بالبوا بنمي",
        "displayName-count-two";
        "بالبوا بنمي",
        "displayName-count-few";
        "بالبوا بنمي",
        "displayName-count-many";
        "بالبوا بنمي",
        "displayName-count-other";
        "بالبوا بنمي",
        "symbol";
        "PAB";
    }
    "PEI";
    {
        "displayName";
        "PEI",
        "symbol";
        "PEI";
    }
    "PEN";
    {
        "displayName";
        "سول بيروفي",
        "displayName-count-zero";
        "سول بيروفي",
        "displayName-count-one";
        "سول بيروفي",
        "displayName-count-two";
        "سول بيروفي",
    }

```



```

        "displayName-count-few";
        "سول بیروفي",
        "displayName-count-many";
        "سول بیروفي",
        "displayName-count-other";
        "سول بیروفي",
        "symbol";
        "PEN";
    }
    "PES";
    {
        "displayName";
        "PES",
        "symbol";
        "PES";
    }
    "PGK";
    {
        "displayName";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-zero";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-one";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-two";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-few";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-many";
        "کینا بابوا غینیا الجديدة",
        "displayName-count-other";
        "کینا بابوا غینیا الجديدة",
        "symbol";
        "PGK";
    }
    "PHP";
    {
        "displayName";
        "بیزو فلپینی",
        "displayName-count-zero";
        "بیزو فلپینی",
        "displayName-count-one";
        "بیزو فلپینی",
        "displayName-count-two";
        "بیزو فلپینی",
        "displayName-count-few";
        "بیزو فلپینی",
        "displayName-count-many";
        "بیزو فلپینی",
        "displayName-count-other";
        "بیزو فلپینی",
        "symbol";
        "PHP",
        "symbol-alt-narrow";
        "₱";
    }
    "PKR";

```

```

{
  "displayName";
  "روبيۃ باڪستاني",
  "displayName-count-zero";
  "روبيۃ باڪستاني",
  "displayName-count-one";
  "روبيۃ باڪستاني",
  "displayName-count-two";
  "روبيۃ باڪستاني",
  "displayName-count-few";
  "روبيۃ باڪستاني",
  "displayName-count-many";
  "روبيۃ باڪستاني",
  "displayName-count-other";
  "روبيۃ باڪستاني",
  "symbol";
  "PKR",
  "symbol-alt-narrow";
  "Rs";
}
"PLN";
{
  "displayName";
  "زلوتي بولندي",
  "displayName-count-zero";
  "زلوتي بولندي",
  "displayName-count-one";
  "زلوتي بولندي",
  "displayName-count-two";
  "زلوتي بولندي",
  "displayName-count-few";
  "زلوتي بولندي",
  "displayName-count-many";
  "زلوتي بولندي",
  "displayName-count-other";
  "زلوتي بولندي",
  "symbol";
  "PLN",
  "symbol-alt-narrow";
  "zł";
}
"PLZ";
{
  "displayName";
  "زلوتي بولندي - 1950-1995",
  "symbol";
  "PLZ";
}
"PTE";
{
  "displayName";
  "اسڪود پرتغالي",
  "symbol";
  "PTE";
}
"PYG";
{

```

```

        "displayName";
        "غواراني باراغواي",
        "displayName-count-zero";
        "غواراني باراغواي",
        "displayName-count-one";
        "غواراني باراغواي",
        "displayName-count-two";
        "غواراني باراغواي",
        "displayName-count-few";
        "غواراني باراغواي",
        "displayName-count-many";
        "غواراني باراغواي",
        "displayName-count-other";
        "غواراني باراغواي",
        "symbol";
        "PYG",
        "symbol-alt-narrow";
        "₲";
    }
    "QAR";
    {
        "displayName";
        "وقد قطري",
        "displayName-count-zero";
        "وقد قطري",
        "displayName-count-one";
        "وقد قطري",
        "displayName-count-two";
        "وقد قطري",
        "displayName-count-few";
        "وقد قطري",
        "displayName-count-many";
        "وقد قطري",
        "displayName-count-other";
        "وقد قطري",
        "symbol";
        "ر.ق.";
    }
    "RHD";
    {
        "displayName";
        "دولار روديسي",
        "symbol";
        "RHD";
    }
    "ROL";
    {
        "displayName";
        "ليو روماني قديم",
        "symbol";
        "ROL";
    }
    "RON";
    {
        "displayName";
        "ليو روماني",
        "displayName-count-zero";

```

```

        "ليو روماني",
        "displayName-count-one";
        "ليو روماني",
        "displayName-count-two";
        "ليو روماني",
        "displayName-count-few";
        "ليو روماني",
        "displayName-count-many";
        "ليو روماني",
        "displayName-count-other";
        "ليو روماني",
        "symbol";
        "RON",
        "symbol-alt-narrow";
        "lei";
    }
    "RSD";
    {
        "displayName";
        "دينار صربي",
        "displayName-count-zero";
        "دينار صربي",
        "displayName-count-one";
        "دينار صربي",
        "displayName-count-two";
        "ديناران صربيان",
        "displayName-count-few";
        "دينارات صربية",
        "displayName-count-many";
        "دينارًا صربيًا",
        "displayName-count-other";
        "دينار صربي",
        "symbol";
        "RSD";
    }
    "RUB";
    {
        "displayName";
        "روبل روسي",
        "displayName-count-zero";
        "روبل روسي",
        "displayName-count-one";
        "روبل روسي",
        "displayName-count-two";
        "روبل روسي",
        "displayName-count-few";
        "روبل روسي",
        "displayName-count-many";
        "روبل روسي",
        "displayName-count-other";
        "روبل روسي",
        "symbol";
        "RUB",
        "symbol-alt-narrow";
        "₽";
    }
    "RUR";

```

```

    {
      "displayName";
      "1998-1991 - روبل روسي",
      "symbol";
      "RUR",
      "symbol-alt-narrow";
      "p.";
    }
    "RWF";
    {
      "displayName";
      "فرنك رواندي",
      "displayName-count-zero";
      "فرنك رواندي",
      "displayName-count-one";
      "فرنك رواندي",
      "displayName-count-two";
      "فرنك رواندي",
      "displayName-count-few";
      "فرنك رواندي",
      "displayName-count-many";
      "فرنك رواندي",
      "displayName-count-other";
      "فرنك رواندي",
      "symbol";
      "RWF",
      "symbol-alt-narrow";
      "RF";
    }
    "SAR";
    {
      "displayName";
      "رول سعودي",
      "displayName-count-zero";
      "رول سعودي",
      "displayName-count-one";
      "رول سعودي",
      "displayName-count-two";
      "رول سعودي",
      "displayName-count-few";
      "رول سعودي",
      "displayName-count-many";
      "رول سعودي",
      "displayName-count-other";
      "رول سعودي",
      "symbol";
      "ر.س.";
    }
    "SBD";
    {
      "displayName";
      "دولار جزر سليمان",
      "displayName-count-zero";
      "دولار جزر سليمان",
      "displayName-count-one";
      "دولار جزر سليمان",
      "displayName-count-two";
    }
  
```

```

        "دولار جزر سليمان",
        "displayName-count-few";
        "دولار جزر سليمان",
        "displayName-count-many";
        "دولار جزر سليمان",
        "displayName-count-other";
        "دولار جزر سليمان",
        "symbol";
        "SBD",
        "symbol-alt-narrow";
        "SB$";
    }
    "SCR";
    {
        "displayName";
        "روبية سيشيلية",
        "displayName-count-zero";
        "روبية سيشيلية",
        "displayName-count-one";
        "روبية سيشيلية",
        "displayName-count-two";
        "روبية سيشيلية",
        "displayName-count-few";
        "روبية سيشيلية",
        "displayName-count-many";
        "روبية سيشيلية",
        "displayName-count-other";
        "روبية سيشيلية",
        "symbol";
        "SCR";
    }
    "SDD";
    {
        "displayName";
        "دينار سوداني",
        "symbol";
        "د.س.";
    }
    "SDG";
    {
        "displayName";
        "جنيه سوداني",
        "displayName-count-zero";
        "جنيه سوداني",
        "displayName-count-one";
        "جنيه سوداني",
        "displayName-count-two";
        "جنيه سوداني",
        "displayName-count-few";
        "جنيهات سودانية",
        "displayName-count-many";
        "جنيهاً سودانياً",
        "displayName-count-other";
        "جنيه سوداني",
        "symbol";
        "ج.س.";
    }
}

```

```

"SDP";
{
  "displayName";
  "جنيه سوداني قديم",
  "symbol";
  "SDP";
}
"SEK";
{
  "displayName";
  "كرونة سويدية",
  "displayName-count-zero";
  "كرونة سويدية",
  "displayName-count-one";
  "كرونة سويدية",
  "displayName-count-two";
  "كرونة سويدية",
  "displayName-count-few";
  "كرونة سويدية",
  "displayName-count-many";
  "كرونة سويدية",
  "displayName-count-other";
  "كرونة سويدية",
  "symbol";
  "SEK",
  "symbol-alt-narrow";
  "kr";
}
"SGD";
{
  "displayName";
  "دولار سنغافوري",
  "displayName-count-zero";
  "دولار سنغافوري",
  "displayName-count-one";
  "دولار سنغافوري",
  "displayName-count-two";
  "دولار سنغافوري",
  "displayName-count-few";
  "دولار سنغافوري",
  "displayName-count-many";
  "دولار سنغافوري",
  "displayName-count-other";
  "دولار سنغافوري",
  "symbol";
  "SGD",
  "symbol-alt-narrow";
  "$";
}
"SHP";
{
  "displayName";
  "جنيه سانت هيلين",
  "displayName-count-zero";
  "جنيه سانت هيلين",
  "displayName-count-one";
  "جنيه سانت هيلين",

```

```

        "displayName-count-two";
        "جنيه سانت هيلين",
        "displayName-count-few";
        "جنيه سانت هيلين",
        "displayName-count-many";
        "جنيه سانت هيلين",
        "displayName-count-other";
        "جنيه سانت هيلين",
        "symbol";
        "SHP",
        "symbol-alt-narrow";
        "£";
    }
    "SIT";
    {
        "displayName";
        "تولار سلوفيني",
        "symbol";
        "SIT";
    }
    "SKK";
    {
        "displayName";
        "كرونه سلوفاكية",
        "symbol";
        "SKK";
    }
    "SLL";
    {
        "displayName";
        "ليون سيراليوني",
        "displayName-count-zero";
        "ليون سيراليوني",
        "displayName-count-one";
        "ليون سيراليوني",
        "displayName-count-two";
        "ليون سيراليوني",
        "displayName-count-few";
        "ليون سيراليوني",
        "displayName-count-many";
        "ليون سيراليوني",
        "displayName-count-other";
        "ليون سيراليوني",
        "symbol";
        "SLL";
    }
    "SOS";
    {
        "displayName";
        "شلن صومالي",
        "displayName-count-zero";
        "شلن صومالي",
        "displayName-count-one";
        "شلن صومالي",
        "displayName-count-two";
        "شلن صومالي",
        "displayName-count-few";
    }

```



```

        "شلن صومالي",
        "displayName-count-many";
        "شلن صومالي",
        "displayName-count-other";
        "شلن صومالي",
        "symbol";
        "SOS";
    }
    "SRD";
    {
        "displayName";
        "دولار سورينامي",
        "displayName-count-zero";
        "دولار سورينامي",
        "displayName-count-one";
        "دولار سورينامي",
        "displayName-count-two";
        "دولار سورينامي",
        "displayName-count-few";
        "دولار سورينامي",
        "displayName-count-many";
        "دولار سورينامي",
        "displayName-count-other";
        "دولار سورينامي",
        "symbol";
        "SRD",
        "symbol-alt-narrow";
        "SR$";
    }
    "SRG";
    {
        "displayName";
        "جلدر سورينامي",
        "symbol";
        "SRG";
    }
    "SSP";
    {
        "displayName";
        "جنيه جنوب السودان",
        "displayName-count-zero";
        "جنيه جنوب السودان",
        "displayName-count-one";
        "جنيه جنوب السودان",
        "displayName-count-two";
        "جنيه جنوب السودان",
        "displayName-count-few";
        "جنيهات جنوب السودان",
        "displayName-count-many";
        "جنيها جنوب السودان",
        "displayName-count-other";
        "جنيه جنوب السودان",
        "symbol";
        "SSP",
        "symbol-alt-narrow";
        "£";
    }
}

```

```

"STD";
{
  "displayName";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-zero";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-one";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-two";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-few";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-many";
  "دوبرا ساو تومي وبرينسيبي",
  "displayName-count-other";
  "دوبرا ساو تومي وبرينسيبي",
  "symbol";
  "STD",
  "symbol-alt-narrow";
  "Db";
}
"STN";
{
  "displayName";
  "STN",
  "symbol";
  "STN";
}
"SUR";
{
  "displayName";
  "روبل سوفيتي",
  "symbol";
  "SUR";
}
"SVC";
{
  "displayName";
  "كولون سلفادوري",
  "symbol";
  "SVC";
}
"SYP";
{
  "displayName";
  "ليرة سورية",
  "displayName-count-zero";
  "ليرة سورية",
  "displayName-count-one";
  "ليرة سورية",
  "displayName-count-two";
  "ليرة سورية",
  "displayName-count-few";
  "ليرة سورية",
  "displayName-count-many";
  "ليرة سورية",
  "displayName-count-other";
}

```

```

        "ليرة سورية",
        "symbol";
        "ل.س.",
        "symbol-alt-narrow";
        "£";
    }
    "SZL";
    {
        "displayName";
        "ليلانجيني سوازيلندي",
        "displayName-count-zero";
        "ليلانجيني سوازيلندي",
        "displayName-count-one";
        "ليلانجيني سوازيلندي",
        "displayName-count-two";
        "ليلانجيني سوازيلندي",
        "displayName-count-few";
        "ليلانجيني سوازيلندي",
        "displayName-count-many";
        "ليلانجيني سوازيلندي",
        "displayName-count-other";
        "ليلانجيني سوازيلندي",
        "symbol";
        "SZL";
    }
    "THB";
    {
        "displayName";
        "باخت تايلاندي",
        "displayName-count-zero";
        "باخت تايلاندي",
        "displayName-count-one";
        "باخت تايلاندي",
        "displayName-count-two";
        "باخت تايلاندي",
        "displayName-count-few";
        "باخت تايلاندي",
        "displayName-count-many";
        "باخت تايلاندي",
        "displayName-count-other";
        "باخت تايلاندي",
        "symbol";
        "฿",
        "symbol-alt-narrow";
        "฿";
    }
    "TJR";
    {
        "displayName";
        "روبل طاجيكستاني",
        "symbol";
        "TJR";
    }
    "TJS";
    {
        "displayName";
        "سوموني طاجيكستاني",

```

```

        "displayName-count-zero";
        "سوموني طاجيكستاني",
        "displayName-count-one";
        "سوموني طاجيكستاني",
        "displayName-count-two";
        "سوموني طاجيكستاني",
        "displayName-count-few";
        "سوموني طاجيكستاني",
        "displayName-count-many";
        "سوموني طاجيكستاني",
        "displayName-count-other";
        "سوموني طاجيكستاني",
        "symbol";
        "TJS";
    }
    "TMM";
    {
        "displayName";
        "مانات تركمنستاني",
        "symbol";
        "TMM";
    }
    "TMT";
    {
        "displayName";
        "مانات تركمانستان",
        "displayName-count-zero";
        "مانات تركمانستان",
        "displayName-count-one";
        "مانات تركمانستان",
        "displayName-count-two";
        "مانات تركمانستان",
        "displayName-count-few";
        "مانات تركمانستان",
        "displayName-count-many";
        "مانات تركمانستان",
        "displayName-count-other";
        "مانات تركمانستان",
        "symbol";
        "TMT";
    }
    "TND";
    {
        "displayName";
        "دينار تونسي",
        "displayName-count-zero";
        "دينار تونسي",
        "displayName-count-one";
        "دينار تونسي",
        "displayName-count-two";
        "ديناران تونسيان",
        "displayName-count-few";
        "دينارات تونسية",
        "displayName-count-many";
        "دينارًا تونسيًا",
        "displayName-count-other";
        "دينار تونسي",
    }

```

```

        "symbol";
        "د.ت.";
    }
    "TOP";
    {
        "displayName";
        "بانغا تونغا",
        "displayName-count-zero";
        "بانغا تونغا",
        "displayName-count-one";
        "بانغا تونغا",
        "displayName-count-two";
        "بانغا تونغا",
        "displayName-count-few";
        "بانغا تونغا",
        "displayName-count-many";
        "بانغا تونغا",
        "displayName-count-other";
        "بانغا تونغا",
        "symbol";
        "TOP",
        "symbol-alt-narrow";
        "T$";
    }
    "TPE";
    {
        "displayName";
        "اسكود تيموري",
        "symbol";
        "TPE";
    }
    "TRL";
    {
        "displayName";
        "ليرة تركي",
        "symbol";
        "TRL";
    }
    "TRY";
    {
        "displayName";
        "ليرة تركية",
        "displayName-count-zero";
        "ليرة تركية",
        "displayName-count-one";
        "ليرة تركية",
        "displayName-count-two";
        "ليرة تركية",
        "displayName-count-few";
        "ليرة تركية",
        "displayName-count-many";
        "ليرة تركية",
        "displayName-count-other";
        "ليرة تركية",
        "symbol";
        "TRY",
        "symbol-alt-narrow";
    }

```

```

        "₹",
        "symbol-alt-variant";
        "TL";
    }
    "TTD";
    {
        "displayName";
        "دولار ترینداد وتوباغو",
        "displayName-count-zero";
        "دولار ترینداد وتوباغو",
        "displayName-count-one";
        "دولار ترینداد وتوباغو",
        "displayName-count-two";
        "دولار ترینداد وتوباغو",
        "displayName-count-few";
        "دولار ترینداد وتوباغو",
        "displayName-count-many";
        "دولار ترینداد وتوباغو",
        "displayName-count-other";
        "دولار ترینداد وتوباغو",
        "symbol";
        "TTD",
        "symbol-alt-narrow";
        "TT$";
    }
    "TWD";
    {
        "displayName";
        "دولار تایوانی",
        "displayName-count-zero";
        "دولار تایوانی",
        "displayName-count-one";
        "دولار تایوانی",
        "displayName-count-two";
        "دولار تایوانی",
        "displayName-count-few";
        "دولار تایوانی",
        "displayName-count-many";
        "دولار تایوانی",
        "displayName-count-other";
        "دولار تایوانی",
        "symbol";
        "NT$";
        "symbol-alt-narrow";
        "NT$";
    }
    "TZS";
    {
        "displayName";
        "شلن تنزانی",
        "displayName-count-zero";
        "شلن تنزانی",
        "displayName-count-one";
        "شلن تنزانی",
        "displayName-count-two";
        "شلن تنزانی",
        "displayName-count-few";
    }

```

```

        "شلن تنزاني",
        "displayName-count-many";
        "شلن تنزاني",
        "displayName-count-other";
        "شلن تنزاني",
        "symbol";
        "TZS";
    }
    "UAH";
    {
        "displayName";
        "هريفنيا أوكراي",
        "displayName-count-zero";
        "هريفنيا أوكراي",
        "displayName-count-one";
        "هريفنيا أوكراي",
        "displayName-count-two";
        "هريفنيا أوكراي",
        "displayName-count-few";
        "هريفنيا أوكراي",
        "displayName-count-many";
        "هريفنيا أوكراي",
        "displayName-count-other";
        "هريفنيا أوكراي",
        "symbol";
        "UAH",
        "symbol-alt-narrow";
        "₴";
    }
    "UAK";
    {
        "displayName";
        "UAK",
        "symbol";
        "UAK";
    }
    "UGS";
    {
        "displayName";
        "شلن أوغندي - 1966-1987",
        "symbol";
        "UGS";
    }
    "UGX";
    {
        "displayName";
        "شلن أوغندي",
        "displayName-count-zero";
        "شلن أوغندي",
        "displayName-count-one";
        "شلن أوغندي",
        "displayName-count-two";
        "شلن أوغندي",
        "displayName-count-few";
        "شلن أوغندي",
        "displayName-count-many";
        "شلن أوغندي",
    }

```

```

        "displayName-count-other";
        "شلن أوغندي",
        "symbol";
        "UGX";
    }
    "USD";
    {
        "displayName";
        "دولار أمريكي",
        "displayName-count-zero";
        "دولار أمريكي",
        "displayName-count-one";
        "دولار أمريكي",
        "displayName-count-two";
        "دولار أمريكي",
        "displayName-count-few";
        "دولار أمريكي",
        "displayName-count-many";
        "دولار أمريكي",
        "displayName-count-other";
        "دولار أمريكي",
        "symbol";
        "US$";
        "symbol-alt-narrow";
        "US$";
    }
    "USN";
    {
        "displayName";
        "دولار أمريكي (اليوم التالي)",
        "symbol";
        "USN";
    }
    "USS";
    {
        "displayName";
        "دولار أمريكي (نفس اليوم)",
        "symbol";
        "USS";
    }
    "UYI";
    {
        "displayName";
        "UYI",
        "symbol";
        "UYI";
    }
    "UYP";
    {
        "displayName";
        "بيزو أوروغواي - 1993-1975",
        "symbol";
        "UYP";
    }
    "UYU";
    {
        "displayName";

```



```

        "بيزو اوروغواي",
        "displayName-count-zero";
        "بيزو اوروغواي",
        "displayName-count-one";
        "بيزو اوروغواي",
        "displayName-count-two";
        "بيزو اوروغواي",
        "displayName-count-few";
        "بيزو اوروغواي",
        "displayName-count-many";
        "بيزو اوروغواي",
        "displayName-count-other";
        "بيزو اوروغواي",
        "symbol";
        "UYU",
        "symbol-alt-narrow";
        "UY$";
    }
    "UZS";
    {
        "displayName";
        "سوم أوزبكستاني",
        "displayName-count-zero";
        "سوم أوزبكستاني",
        "displayName-count-one";
        "سوم أوزبكستاني",
        "displayName-count-two";
        "سوم أوزبكستاني",
        "displayName-count-few";
        "سوم أوزبكستاني",
        "displayName-count-many";
        "سوم أوزبكستاني",
        "displayName-count-other";
        "سوم أوزبكستاني",
        "symbol";
        "UZS";
    }
    "VEB";
    {
        "displayName";
        "بوليفار فنزويلي - 2008-1871",
        "symbol";
        "VEB";
    }
    "VEF";
    {
        "displayName";
        "بوليفار فنزويلي",
        "displayName-count-zero";
        "بوليفار فنزويلي",
        "displayName-count-one";
        "بوليفار فنزويلي",
        "displayName-count-two";
        "بوليفار فنزويلي",
        "displayName-count-few";
        "بوليفار فنزويلي",
        "displayName-count-many";
    }

```

```

        "بوليفار فنزويلي",
        "displayName-count-other";
        "بوليفار فنزويلي",
        "symbol";
        "VEF",
        "symbol-alt-narrow";
        "Bs";
    }
    "VND";
    {
        "displayName";
        "دونج فيتنامي",
        "displayName-count-zero";
        "دونج فيتنامي",
        "displayName-count-one";
        "دونج فيتنامي",
        "displayName-count-two";
        "دونج فيتنامي",
        "displayName-count-few";
        "دونج فيتنامي",
        "displayName-count-many";
        "دونج فيتنامي",
        "displayName-count-other";
        "دونج فيتنامي",
        "symbol";
        "₫",
        "symbol-alt-narrow";
        "₫";
    }
    "VNN";
    {
        "displayName";
        "VNN",
        "symbol";
        "VNN";
    }
    "VUV";
    {
        "displayName";
        "فاتو فانواتو",
        "displayName-count-zero";
        "فاتو فانواتو",
        "displayName-count-one";
        "فاتو فانواتو",
        "displayName-count-two";
        "فاتو فانواتو",
        "displayName-count-few";
        "فاتو فانواتو",
        "displayName-count-many";
        "فاتو فانواتو",
        "displayName-count-other";
        "فاتو فانواتو",
        "symbol";
        "VUV";
    }
    "WST";
    {

```

```

        "displayName";
        "تالا ساموا",
        "displayName-count-zero";
        "تالا ساموا",
        "displayName-count-one";
        "تالا ساموا",
        "displayName-count-two";
        "تالا ساموا",
        "displayName-count-few";
        "تالا ساموا",
        "displayName-count-many";
        "تالا ساموا",
        "displayName-count-other";
        "تالا ساموا",
        "symbol";
        "WST";
    }
    "XAF";
    {
        "displayName";
        "فرنك وسط أفريقي",
        "displayName-count-zero";
        "فرنك وسط أفريقي",
        "displayName-count-one";
        "فرنك وسط أفريقي",
        "displayName-count-two";
        "فرنك وسط أفريقي",
        "displayName-count-few";
        "فرنك وسط أفريقي",
        "displayName-count-many";
        "فرنك وسط أفريقي",
        "displayName-count-other";
        "فرنك وسط أفريقي",
        "symbol";
        "FCFA";
    }
    "XAG";
    {
        "displayName";
        "فضة",
        "symbol";
        "XAG";
    }
    "XAU";
    {
        "displayName";
        "ذهب",
        "symbol";
        "XAU";
    }
    "XBA";
    {
        "displayName";
        "الوحدة الأوروبية المركبة",
        "symbol";
        "XBA";
    }
}

```

```

    "XBB";
    {
      "displayName";
      "الوحدة المالية الأوروبية",
      "symbol";
      "XBB";
    }
    "XBC";
    {
      "displayName";
      "الوحدة الحسابية الأوروبية",
      "symbol";
      "XBC";
    }
    "XBD";
    {
      "displayName";
      "وحدة الحساب الأوروبية (XBD)",
      "symbol";
      "XBD";
    }
    "XCD";
    {
      "displayName";
      "دولار شرق الكاريبي",
      "displayName-count-zero";
      "دولار شرق الكاريبي",
      "displayName-count-one";
      "دولار شرق الكاريبي",
      "displayName-count-two";
      "دولار شرق الكاريبي",
      "displayName-count-few";
      "دولار شرق الكاريبي",
      "displayName-count-many";
      "دولار شرق الكاريبي",
      "displayName-count-other";
      "دولار شرق الكاريبي",
      "symbol";
      "EC$";
      "symbol-alt-narrow";
      "$";
    }
    "XDR";
    {
      "displayName";
      "حقوق السحب الخاصة",
      "symbol";
      "XDR";
    }
    "XEU";
    {
      "displayName";
      "وحدة النقد الأوروبية",
      "symbol";
      "XEU";
    }
    "XFO";

```

```

    {
      "displayName";
      "فرنك فرنسي ذهبي",
      "symbol";
      "XFO";
    }
    "XFU";
    {
      "displayName";
      "فرنك فرنسي (UIC)",
      "symbol";
      "XFU";
    }
    "XOF";
    {
      "displayName";
      "فرنك غرب أفريقي",
      "displayName-count-zero";
      "فرنك غرب أفريقي",
      "displayName-count-one";
      "فرنك غرب أفريقي",
      "displayName-count-two";
      "فرنك غرب أفريقي",
      "displayName-count-few";
      "فرنك غرب أفريقي",
      "displayName-count-many";
      "فرنك غرب أفريقي",
      "displayName-count-other";
      "فرنك غرب أفريقي",
      "symbol";
      "CFA";
    }
    "XPD";
    {
      "displayName";
      "بالاديوم",
      "symbol";
      "XPD";
    }
    "XPF";
    {
      "displayName";
      "فرنك سي إف بي",
      "displayName-count-zero";
      "فرنك سي إف بي",
      "displayName-count-one";
      "فرنك سي إف بي",
      "displayName-count-two";
      "فرنك سي إف بي",
      "displayName-count-few";
      "فرنك سي إف بي",
      "displayName-count-many";
      "فرنك سي إف بي",
      "displayName-count-other";
      "فرنك سي إف بي",
      "symbol";
      "CFPF";
    }
  
```

```

    }
    "XPT";
    {
        "displayName";
        "البلاطين",
        "symbol";
        "XPT";
    }
    "XRE";
    {
        "displayName";
        "XRE",
        "symbol";
        "XRE";
    }
    "XSU";
    {
        "displayName";
        "XSU",
        "symbol";
        "XSU";
    }
    "XTS";
    {
        "displayName";
        "كود اختبار العملة",
        "symbol";
        "XTS";
    }
    "XUA";
    {
        "displayName";
        "XUA",
        "symbol";
        "XUA";
    }
    "XXX";
    {
        "displayName";
        "عملة غير معروفة",
        "displayName-count-zero";
        "(عملة غير معروفة)",
        "displayName-count-one";
        "(عملة غير معروفة)",
        "displayName-count-two";
        "(عملة غير معروفة)",
        "displayName-count-few";
        "(عملة غير معروفة)",
        "displayName-count-many";
        "(عملة غير معروفة)",
        "displayName-count-other";
        "(عملة غير معروفة)",
        "symbol";
        "***";
    }
    "YDD";
    {

```

```

        "displayName";
        "دينار يمني",
        "symbol";
        "YDD";
    }
    "YER";
    {
        "displayName";
        "ﷲ يمني",
        "displayName-count-zero";
        "ﷲ يمني",
        "displayName-count-one";
        "ﷲ يمني",
        "displayName-count-two";
        "ﷲ يمني",
        "displayName-count-few";
        "ﷲ يمني",
        "displayName-count-many";
        "ﷲ يمني",
        "displayName-count-other";
        "ﷲ يمني",
        "symbol";
        "ر.ي.";
    }
    "YUD";
    {
        "displayName";
        "دينار يوغسلافي",
        "symbol";
        "YUD";
    }
    "YUM";
    {
        "displayName";
        "YUM",
        "symbol";
        "YUM";
    }
    "YUN";
    {
        "displayName";
        "دينار يوغسلافي قابل للتحويل",
        "symbol";
        "YUN";
    }
    "YUR";
    {
        "displayName";
        "YUR",
        "symbol";
        "YUR";
    }
    "ZAL";
    {
        "displayName";
        "راند جنوب أفريقيا -مالي",
        "symbol";
    }

```

```

        "ZAL";
    }
    "ZAR";
    {
        "displayName";
        "راند جنوب أفريقيا",
        "displayName-count-zero";
        "راند جنوب أفريقيا",
        "displayName-count-one";
        "راند جنوب أفريقيا",
        "displayName-count-two";
        "راند جنوب أفريقيا",
        "displayName-count-few";
        "راند جنوب أفريقيا",
        "displayName-count-many";
        "راند جنوب أفريقيا",
        "displayName-count-other";
        "راند جنوب أفريقيا",
        "symbol";
        "ZAR",
        "symbol-alt-narrow";
        "R";
    }
    "ZMK";
    {
        "displayName";
        "2012-1968 - زامبي",
        "displayName-count-zero";
        "2012-1968 - زامبي",
        "displayName-count-one";
        "2012-1968 - زامبي",
        "displayName-count-two";
        "2012-1968 - زامبي",
        "displayName-count-few";
        "2012-1968 - زامبي",
        "displayName-count-many";
        "2012-1968 - زامبي",
        "displayName-count-other";
        "2012-1968 - زامبي",
        "symbol";
        "ZMK";
    }
    "ZMW";
    {
        "displayName";
        "زامبي",
        "displayName-count-zero";
        "زامبي",
        "displayName-count-one";
        "زامبي",
        "displayName-count-two";
        "زامبي",
        "displayName-count-few";
        "زامبي",
        "displayName-count-many";
        "زامبي",
        "displayName-count-other";
    }

```



```

        "کواشا زامبي"،
        "symbol";
    "ZMW",
        "symbol-alt-narrow";
    "ZK";
}
"ZRN";
{
    "displayName";
    "زائير زائيري جديد",
    "symbol";
    "ZRN";
}
"ZRZ";
{
    "displayName";
    "زائير زائيري",
    "symbol";
    "ZRZ";
}
"ZWD";
{
    "displayName";
    "دولار زمبابوي",
    "symbol";
    "ZWD";
}
"ZWL";
{
    "displayName";
    "دولار زمبابوي 2009",
    "symbol";
    "ZWL";
}
"ZWR";
{
    "displayName";
    "ZWR",
    "symbol";
    "ZWR";
}
}
}
}
}

```

INDEX.JSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DatePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
  'ar': {
    'datetimepicker': {
      placeholder: 'حدد التاريخ والوقت',
      today: 'اليوم'
    }
  }
});
function App() {
  return <DateTimePickerComponent id="datetimepicker" enableRtl={true}
  locale='ar' />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
// Load the L10n, loadCldr from ej2-base
import { loadCldr, L10n } from '@syncfusion/ej2-base';
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
// load the CLDR data files.
// 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 * as timeZoneNames from './timeZoneNames.json';
loadCldr(numberingSystems, gregorian, numbers, timeZoneNames);
L10n.load({
  'ar': {
    'datetimepicker': {
      placeholder: 'حدد التاريخ والوقت',
      today: 'اليوم'
    }
  }
});
function App() {
  return <DateTimePickerComponent id="datetimepicker" enableRtl={true}
  locale='ar' />
}
ReactDOM.render(<App />, document.getElementById('element'));
```

NUMBERINGSYSTEMS.JSON

```
{
  "supplemental": {
    "version": {
      "_number": "$Revision: 12732 $",
      "_unicodeVersion": "9.0.0",
      "_cldrVersion": "31"
    },
  },
}
```

```

"numberingSystems": {
  "adlm": {
    "_digits": "୧୪୫୬୭୮୯୧୦",
    "_type": "numeric"
  },
  "ahom": {
    "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
    "_type": "numeric"
  },
  "arab": {
    "_digits": "٠١٢٣٤٥٦٧٨٩",
    "_type": "numeric"
  },
  "arabext": {
    "_digits": "٠١٢٣٤٥٦٧٨٩",
    "_type": "numeric"
  },
  "armn": {
    "_rules": "armenian-upper",
    "_type": "algorithmic"
  },
  "armnlow": {
    "_rules": "armenian-lower",
    "_type": "algorithmic"
  },
  "bali": {
    "_digits": "᭐᭑᭒᭓᭔᭕᭖᭗᭘᭙",
    "_type": "numeric"
  },
  "beng": {
    "_digits": "০১২৩৪৫৬৭৮৯",
    "_type": "numeric"
  },
  "bhks": {
    "_digits": "᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉",
    "_type": "numeric"
  },
  "brah": {
    "_digits": "୦୧୨୩୪୫୬୭୮୯",
    "_type": "numeric"
  },
  "cakm": {
    "_digits": "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
    "_type": "numeric"
  },
  "cham": {
    "_digits": "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉ",
    "_type": "numeric"
  },
  "cyr1": {
    "_rules": "cyrillic-lower",
    "_type": "algorithmic"
  },
  "deva": {
    "_digits": "०१२३४५६७८९",
    "_type": "numeric"
  }
}

```

```

    },
    "ethi": {
      "_rules": "ethiopic",
      "_type": "algorithmic"
    },
    "fullwide": {
      "_digits": "0 1 2 3 4 5 6 7 8 9",
      "_type": "numeric"
    },
    "geor": {
      "_rules": "georgian",
      "_type": "algorithmic"
    },
    "grek": {
      "_rules": "greek-upper",
      "_type": "algorithmic"
    },
    "greklow": {
      "_rules": "greek-lower",
      "_type": "algorithmic"
    },
    "gujr": {
      "_digits": "૦ ૧ ૨ ૩ ૪ ૫ ૬ ૭ ૮ ૯",
      "_type": "numeric"
    },
    "guru": {
      "_digits": "੦ ੧ ੨ ੩ ੪ ੫ ੬ ੭ ੮ ੯",
      "_type": "numeric"
    },
    "hanidays": {
      "_rules": "zh/SpelloutRules/spellout-numbering-days",
      "_type": "algorithmic"
    },
    "hanidec": {
      "_digits": "〇 一 二 三 四 五 六 七 八 九",
      "_type": "numeric"
    },
    "hans": {
      "_rules": "zh/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hansfin": {
      "_rules": "zh/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hant": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal",
      "_type": "algorithmic"
    },
    "hantfin": {
      "_rules": "zh_Hant/SpelloutRules/spellout-cardinal-financial",
      "_type": "algorithmic"
    },
    "hebr": {
      "_rules": "hebrew",
      "_type": "algorithmic"
    }
  }

```

```

"hmng": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"java": {
  "_digits": "၀၁၂၃၄၅၆၇၈၉၀၁၂၃၄၅၆၇၈၉",
  "_type": "numeric"
},
"jpan": {
  "_rules": "ja/SpelloutRules/spellout-cardinal",
  "_type": "algorithmic"
},
"jpanfin": {
  "_rules": "ja/SpelloutRules/spellout-cardinal-financial",
  "_type": "algorithmic"
},
"kali": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"khmr": {
  "_digits": "០១២៣៤៥៦៧៨៩",
  "_type": "numeric"
},
"knda": {
  "_digits": "೦೧೨೩೪೫೬೭೮೯",
  "_type": "numeric"
},
"lana": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"lanatham": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"lao": {
  "_digits": "໐໑໒໓໔໕໖໗໘໑",
  "_type": "numeric"
},
"latn": {
  "_digits": "0123456789",
  "_type": "numeric"
},
"lepc": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"limb": {
  "_digits": "□□□□□□□□□□",
  "_type": "numeric"
},
"mathbold": {
  " digits": "0123456789",

```

```
    "_type": "numeric"
  },
  "mathdbl": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathmono": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsanb": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mathsans": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mlym": {
    "_digits": "൦൧൨൩൪൫൬൭൮൯",
    "_type": "numeric"
  },
  "modi": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mong": {
    "_digits": "᠐᠑᠒᠓᠐ᠠᠨᠵᠠᠨᠠᠨᠵᠠᠨ",
    "_type": "numeric"
  },
  "mroo": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  "mtei": {
    "_digits": "ᐐᐑᐒᐓᐔᐕᐖᐗᐘᐙᐚ",
    "_type": "numeric"
  },
  "mymr": {
    "_digits": "၀၁၂၃၄၅၆၇၈",
    "_type": "numeric"
  },
  "mymrshan": {
    "_digits": "0123456789",
    "_type": "numeric"
  },
  "mymrtlng": {
    "_digits": "ᐐᐑᐒᐓᐔᐕᐖᐗᐘᐙᐚ",
    "_type": "numeric"
  },
  "newa": {
    "_digits": "□□□□□□□□□□",
    "_type": "numeric"
  },
  },
```

```
"nkoo": {
    "_digits": "ᠨᠬᠡᠭᠣᠤ",
    "_type": "numeric"
},
"olck": {
    "_digits": "ᠣᠯᠢᠴᠦ᠋ᠭᠦ",
    "_type": "numeric"
},
"orya": {
    "_digits": "ᠣᠷᠶᠠ",
    "_type": "numeric"
},
"osma": {
    "_digits": "ᠣᠰᠮᠠ",
    "_type": "numeric"
},
"roman": {
    "_rules": "roman-upper",
    "_type": "algorithmic"
},
"romanlow": {
    "_rules": "roman-lower",
    "_type": "algorithmic"
},
"saur": {
    "_digits": "ᠰᠠᠷ",
    "_type": "numeric"
},
"shrd": {
    "_digits": "ᠱᠠᠷᠳᠠ",
    "_type": "numeric"
},
"sind": {
    "_digits": "ᠰᠢᠩᠳᠠ",
    "_type": "numeric"
},
"sinh": {
    "_digits": "ᠰᠢᠨᠬᠠ",
    "_type": "numeric"
},
"sora": {
    "_digits": "ᠰᠣᠷᠠ",
    "_type": "numeric"
},
"sund": {
    "_digits": "ᠰᠤᠩᠳᠠ",
    "_type": "numeric"
},
"takr": {
    "_digits": "ᠲᠠᠬᠢᠷ",
    "_type": "numeric"
},
"talv": {
    "_digits": "ᠲᠠᠯᠪᠠ",
    "_type": "numeric"
```

```

    },
    "taml": {
      "_rules": "tamil",
      "_type": "algorithmic"
    },
    "tamldec": {
      "_digits": "0௧௨௩௪௫௬௭௮௯",
      "_type": "numeric"
    },
    "telu": {
      "_digits": "౦౧౨౩౪౫౬౭౮౯",
      "_type": "numeric"
    },
    "thai": {
      "_digits": "๐๑๒๓๔๕๖๗๘๙",
      "_type": "numeric"
    },
    "tibtb": {
      "_digits": "༠༡༢༣༤༥༦༧༨༩",
      "_type": "numeric"
    },
    "tirh": {
      "_digits": "౦౧౨౩౪౫౬౭౮౯",
      "_type": "numeric"
    },
    "vaih": {
      "_digits": "0123456789",
      "_type": "numeric"
    },
    "wara": {
      "_digits": "0123456789",
      "_type": "numeric"
    }
  }
}

```

NUMBERINGSYSTEMS.JSX

```

{
  "supplemental";
  {
    "version";
    {
      "_number";
      "$Revision: 12732 $",
      "_unicodeVersion";
      "9.0.0",
      "_cldrVersion";
      "31";
    }
    "numberingSystems";
    {
      "adlm";
      {

```



```

        "_digits";
        "୧୨୩୪୫୬୭୮୯୦",
        "_type";
        "numeric";
    }
    "ahom";
    {
        "_digits";
        "ᱠᱡᱢᱣᱤᱨᱫᱽᱴᱚᱴ",
        "_type";
        "numeric";
    }
    "arab";
    {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
    }
    "arabext";
    {
        "_digits";
        "٠١٢٣٤٥٦٧٨٩",
        "_type";
        "numeric";
    }
    "armn";
    {
        "_rules";
        "armenian-upper",
        "_type";
        "algorithmic";
    }
    "armnlow";
    {
        "_rules";
        "armenian-lower",
        "_type";
        "algorithmic";
    }
    "bali";
    {
        "_digits";
        "᭐᭄ᭅᭆᭇᭈᭉᭊᭋᭌ᭍᭎᭏",
        "_type";
        "numeric";
    }
    "beng";
    {
        "_digits";
        "୦୧୨୩୪୫୬୭୮୯",
        "_type";
        "numeric";
    }
    "bhks";
    {

```

```

        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "brah";
    {
        "_digits";
        "ᱠᱡᱢᱯᱤᱨᱫᱽᱜᱟᱲ",
        "_type";
        "numeric";
    }
    "cakm";
    {
        "_digits";
        "ᱵᱚᱠᱫᱽᱨᱫᱽᱯᱩᱨ",
        "_type";
        "numeric";
    }
    "cham";
    {
        "_digits";
        "ႤႬႬႬႬႬႬႬႬ",
        "_type";
        "numeric";
    }
    "cyr1";
    {
        "_rules";
        "cyrillic-lower",
        "_type";
        "algorithmic";
    }
    "deva";
    {
        "_digits";
        "ᱠᱣᱟᱨᱢᱟᱝᱞᱟᱲ",
        "_type";
        "numeric";
    }
    "ethi";
    {
        "_rules";
        "ethiopic",
        "_type";
        "algorithmic";
    }
    "fullwide";
    {
        "_digits";
        "0 1 2 3 4 5 6 7 8 9",
        "_type";
        "numeric";
    }
    "geor";
    {

```

```

        "_rules";
        "georgian",
        "_type";
        "algorithmic";
    }
    "grek";
    {
        "_rules";
        "greek-upper",
        "_type";
        "algorithmic";
    }
    "greklow";
    {
        "_rules";
        "greek-lower",
        "_type";
        "algorithmic";
    }
    "gujr";
    {
        "_digits";
        "૦૧૨૩૪૫૬૭૮૯",
        "_type";
        "numeric";
    }
    "guru";
    {
        "_digits";
        "੦੧੨੩੪੫੬੭੮੯",
        "_type";
        "numeric";
    }
    "hanidays";
    {
        "_rules";
        "zh/SpelloutRules/spellout-numbering-days",
        "_type";
        "algorithmic";
    }
    "hanidec";
    {
        "_digits";
        "〇一二三四五六七八九",
        "_type";
        "numeric";
    }
    "hans";
    {
        "_rules";
        "zh/SpelloutRules/spellout-cardinal",
        "_type";
        "algorithmic";
    }
    "hansfin";
    {

```

```
"_rules";
    "zh/SpelloutRules/spellout-cardinal-financial",
        "_type";
    "algorithmic";
}
"hant";
{
    "_rules";
    "zh_Hant/SpelloutRules/spellout-cardinal",
        "_type";
    "algorithmic";
}
"hantfin";
{
    "_rules";
    "zh_Hant/SpelloutRules/spellout-cardinal-financial",
        "_type";
    "algorithmic";
}
"hebr";
{
    "_rules";
    "hebrew",
        "_type";
    "algorithmic";
}
"hmng";
{
    "_digits";
    "□□□□□□□□□□",
        "_type";
    "numeric";
}
"java";
{
    "_digits";
    "᐀ᐁᐂᐃᐄᐅᐆᐇᐈᐉᐊᐋᐌᐍᐎᐏᐐᐑᐒᐓᐔᐕᐖᐗᐘᐙᐚᐛᐜᐝᐞᐟᐠᐡᐢᐣᐤᐥᐦᐧᐨᐩᐪᐫᐬᐭᐮᐯᐰᐱᐲᐳᐴᐵᐶᐷᐸᐹᐺᐻᐼᐽᐾᐿ",
        "_type";
    "numeric";
}
"jpan";
{
    "_rules";
    "ja/SpelloutRules/spellout-cardinal",
        "_type";
    "algorithmic";
}
"jpanfin";
{
    "_rules";
    "ja/SpelloutRules/spellout-cardinal-financial",
        "_type";
    "algorithmic";
}
"kali";
{
```

```

        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "khmr";
    {
        "_digits";
        "០១២៣៤៥៦៧៨៩",
        "_type";
        "numeric";
    }
    "knda";
    {
        "_digits";
        "೦೧೨೩೪೫೬೭೮೯",
        "_type";
        "numeric";
    }
    "lana";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "lanatham";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "laoo";
    {
        "_digits";
        "໐໑໒໓໔໕໖໗໘໑",
        "_type";
        "numeric";
    }
    "latn";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "lepc";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "limb";
    {

```

```

        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mathbold";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathdbl";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathmono";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsanb";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mathsans";
    {
        "_digits";
        "0123456789",
        "_type";
        "numeric";
    }
    "mlym";
    {
        "_digits";
        "൦൧൨൩൪൫൬൭൮൯",
        "_type";
        "numeric";
    }
    "modi";
    {
        "_digits";
        "□□□□□□□□",
        "_type";
        "numeric";
    }
    "mong";
    {

```

```

        "_digits";
        "၀၇၇၇၇၇၇၇",
        "_type";
        "numeric";
    }
    "mroo";
    {
        "_digits";
        "၀၀၀၀၀၀၀၀၀၀",
        "_type";
        "numeric";
    }
    "mtei";
    {
        "_digits";
        "၀၄၈၈၈၈၈၈၈၈၈",
        "_type";
        "numeric";
    }
    "mymr";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "mymrshan";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈",
        "_type";
        "numeric";
    }
    "mymrtlng";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
    "newa";
    {
        "_digits";
        "၀၀၀၀၀၀၀၀၀၀",
        "_type";
        "numeric";
    }
    "nkoo";
    {
        "_digits";
        "၀၁၂၃၄၅၆၇၈၉",
        "_type";
        "numeric";
    }
}

```

```

"olck";
{
  "_digits";
  "0٠١٢٣٤٥٦٧٨٩",
  "_type";
  "numeric";
}
"orya";
{
  "_digits";
  "୦୧୨୩୪୫୬୭୮୯",
  "_type";
  "numeric";
}
"osma";
{
  "_digits";
  "0᱀᱁᱂᱃᱄᱅᱆᱇᱈᱉᱊᱋᱌ᱍᱎᱏ᱐᱑᱒᱓᱔᱕᱖᱗᱘᱙ᱚᱛᱜᱝᱞᱟᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type";
  "numeric";
}
"roman";
{
  "_rules";
  "roman-upper",
  "_type";
  "algorithmic";
}
"romanlow";
{
  "_rules";
  "roman-lower",
  "_type";
  "algorithmic";
}
"saur";
{
  "_digits";
  "ᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type";
  "numeric";
}
"shrd";
{
  "_digits";
  "ᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type";
  "numeric";
}
"sind";
{
  "_digits";
  "ᱠᱡᱢᱣᱤᱥᱦᱧᱨᱩᱪᱫᱬᱭᱮᱱᱲᱳᱴᱵᱶᱷᱸᱹ",
  "_type";
  "numeric";
}

```



```

"sinh";
{
  "_digits";
  "෦෧෨෩෪෫෬෭෮෯",
  "_type";
  "numeric";
}
"sora";
{
  "_digits";
  "୦୧୨୩୪୫୬୭୮୯",
  "_type";
  "numeric";
}
"sund";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"takr";
{
  "_digits";
  "□□□□□□□□",
  "_type";
  "numeric";
}
"talu";
{
  "_digits";
  "౦౧౨౩౪౫౬౭౮౯",
  "_type";
  "numeric";
}
"taml";
{
  "_rules";
  "tamil",
  "_type";
  "algorithmic";
}
"tamldec";
{
  "_digits";
  "௦௧௨௩௪௫௬௭௮௯",
  "_type";
  "numeric";
}
"telu";
{
  "_digits";
  "౦౧౨౩౪౫౬౭౮౯",
  "_type";
  "numeric";
}

```

```

        "thai";
    {
        "_digits";
        "๐๑๒๓๔๕๖๗๘๙",
        "_type";
        "numeric";
    }
    "tibet";
    {
        "_digits";
        "༠༡༢༣༤༥༦༧༨༩",
        "_type";
        "numeric";
    }
    "tirh";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
    "vaii";
    {
        "_digits";
        "ᲠᲡᲢᲣᲤᲥᲦᲧᲨᲩᲪᲫᲬᲭᲮᲯᲰᲱᲲᲳᲴᲵᲶᲷᲸᲹᲺ᲻᲼ᲽᲾᲿ",
        "_type";
        "numeric";
    }
    "wara";
    {
        "_digits";
        "□□□□□□□□□□",
        "_type";
        "numeric";
    }
}

}

```

NUMBERS.JSON

```
{
  "main": {
    "ar": {
      "identity": {
        "version": {
          "_number": "$Revision: 13686 $",
          "_cldrVersion": "32"
        },
        "language": "ar"
      },
      "numbers": {
        "defaultNumberingSystem": "arab",
        "otherNumberingSystems": {
          "native": "arab"
        }
      }
    }
  }
}
```

```

    "minimumGroupingDigits": "1",
    "symbols-numberSystem-arab": {
      "decimal": ",",
      "group": ",",
      "list": ";",
      "percentSign": "%",
      "plusSign": "+",
      "minusSign": "-",
      "exponential": "١٠^",
      "superscriptingExponent": "×",
      "perMille": "‰",
      "infinity": "∞",
      "nan": "ليس رقم",
      "timeSeparator": ":"
    },
    "symbols-numberSystem-latn": {
      "decimal": ".",
      "group": ",",
      "list": ";",
      "percentSign": "%",
      "plusSign": "+",
      "minusSign": "-",
      "exponential": "E",
      "superscriptingExponent": "×",
      "perMille": "‰",
      "infinity": "∞",
      "nan": "ليس رقمًا",
      "timeSeparator": ":"
    },
    "decimalFormats-numberSystem-arab": {
      "standard": "#,##0.###",
      "long": {
        "decimalFormat": {
          "1000-count-zero": "0 ألف",
          "1000-count-one": "0 ألف",
          "1000-count-two": "0 ألف",
          "1000-count-few": "0 آلاف",
          "1000-count-many": "0 ألف",
          "1000-count-other": "0 ألف",
          "10000-count-zero": "00 ألف",
          "10000-count-one": "00 ألف",
          "10000-count-two": "00 ألف",
          "10000-count-few": "00 ألف",
          "10000-count-many": "00 ألف",
          "10000-count-other": "00 ألف",
          "100000-count-zero": "000 ألف",
          "100000-count-one": "000 ألف",
          "100000-count-two": "000 ألف",
          "100000-count-few": "000 ألف",
          "100000-count-many": "000 ألف",
          "100000-count-other": "000 ألف",
          "1000000-count-zero": "0 مليون",
          "1000000-count-one": "0 مليون",
          "1000000-count-two": "0 مليون",
          "1000000-count-few": "0 ملايين",
          "1000000-count-many": "0 مليون",
          "1000000-count-other": "0 مليون",
        }
      }
    }
  }

```

```

        "10000000-count-zero": "00 مليون",
        "10000000-count-one": "00 مليون",
        "10000000-count-two": "00 مليون",
        "10000000-count-few": "00 ملايين",
        "10000000-count-many": "00 مليون",
        "10000000-count-other": "00 مليون",
        "100000000-count-zero": "000 مليون",
        "100000000-count-one": "000 مليون",
        "100000000-count-two": "000 مليون",
        "100000000-count-few": "000 مليون",
        "100000000-count-many": "000 مليون",
        "100000000-count-other": "000 مليون",
        "1000000000-count-zero": "0 مليار",
        "1000000000-count-one": "0 مليار",
        "1000000000-count-two": "0 مليار",
        "1000000000-count-few": "0 مليار",
        "1000000000-count-many": "0 مليار",
        "1000000000-count-other": "0 مليار",
        "10000000000-count-zero": "00 مليار",
        "10000000000-count-one": "00 مليار",
        "10000000000-count-two": "00 مليار",
        "10000000000-count-few": "00 مليار",
        "10000000000-count-many": "00 مليار",
        "10000000000-count-other": "00 مليار",
        "100000000000-count-zero": "000 مليار",
        "100000000000-count-one": "000 مليار",
        "100000000000-count-two": "000 مليار",
        "100000000000-count-few": "000 مليار",
        "100000000000-count-many": "000 مليار",
        "100000000000-count-other": "000 مليار",
        "1000000000000-count-zero": "0 ترليون",
        "1000000000000-count-one": "0 ترليون",
        "1000000000000-count-two": "0 ترليون",
        "1000000000000-count-few": "0 ترليون",
        "1000000000000-count-many": "0 ترليون",
        "1000000000000-count-other": "0 ترليون",
        "10000000000000-count-zero": "00 ترليون",
        "10000000000000-count-one": "00 ترليون",
        "10000000000000-count-two": "00 ترليون",
        "10000000000000-count-few": "00 ترليون",
        "10000000000000-count-many": "00 ترليون",
        "10000000000000-count-other": "00 ترليون",
        "100000000000000-count-zero": "000 ترليون",
        "100000000000000-count-one": "000 ترليون",
        "100000000000000-count-two": "000 ترليون",
        "100000000000000-count-few": "000 ترليون",
        "100000000000000-count-many": "000 ترليون",
        "100000000000000-count-other": "000 ترليون"
    },
    },
    "short": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",
            "1000-count-one": "0 ألف",
            "1000-count-two": "0 ألف",
            "1000-count-few": "0 آلاف",
            "1000-count-many": "0 ألف",
        }
    }
}

```

```
"1000-count-other": "0 ألف",
"10000-count-zero": "00 ألف",
"10000-count-one": "00 ألف",
"10000-count-two": "00 ألف",
"10000-count-few": "00 ألف",
"10000-count-many": "00 ألف",
"10000-count-other": "00 ألف",
"100000-count-zero": "000 ألف",
"100000-count-one": "000 ألف",
"100000-count-two": "000 ألف",
"100000-count-few": "000 ألف",
"100000-count-many": "000 ألف",
"100000-count-other": "000 ألف",
"1000000-count-zero": "0 مليون",
"1000000-count-one": "0 مليون",
"1000000-count-two": "0 مليون",
"1000000-count-few": "0 مليون",
"1000000-count-many": "0 مليون",
"1000000-count-other": "0 مليون",
"10000000-count-zero": "00 مليون",
"10000000-count-one": "00 مليون",
"10000000-count-two": "00 مليون",
"10000000-count-few": "00 مليون",
"10000000-count-many": "00 مليون",
"10000000-count-other": "00 مليون",
"100000000-count-zero": "000 مليون",
"100000000-count-one": "000 مليون",
"100000000-count-two": "000 مليون",
"100000000-count-few": "000 مليون",
"100000000-count-many": "000 مليون",
"100000000-count-other": "000 مليون",
"1000000000-count-zero": "0 مليار",
"1000000000-count-one": "0 مليار",
"1000000000-count-two": "0 مليار",
"1000000000-count-few": "0 مليار",
"1000000000-count-many": "0 مليار",
"1000000000-count-other": "0 مليار",
"10000000000-count-zero": "00 مليار",
"10000000000-count-one": "00 مليار",
"10000000000-count-two": "00 مليار",
"10000000000-count-few": "00 مليار",
"10000000000-count-many": "00 مليار",
"10000000000-count-other": "00 مليار",
"100000000000-count-zero": "0 ترليون",
"100000000000-count-one": "0 ترليون",
"100000000000-count-two": "0 ترليون",
"100000000000-count-few": "0 ترليون",
"100000000000-count-many": "0 ترليون",
"100000000000-count-other": "0 ترليون",
"1000000000000-count-zero": "00 ترليون",
"1000000000000-count-one": "00 ترليون",
```

```

        "10000000000000-count-two": "00 ترليون",
        "10000000000000-count-few": "00 ترليون",
        "10000000000000-count-many": "00 ترليون",
        "10000000000000-count-other": "00 ترليون",
        "10000000000000-count-zero": "000 ترليون",
        "10000000000000-count-one": "000 ترليون",
        "10000000000000-count-two": "000 ترليون",
        "10000000000000-count-few": "000 ترليون",
        "10000000000000-count-many": "000 ترليون",
        "10000000000000-count-other": "000 ترليون"
    }
}
},
"decimalFormats-numberSystem-latn": {
    "standard": "#,##0.###",
    "long": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",
            "1000-count-one": "0 ألف",
            "1000-count-two": "0 ألف",
            "1000-count-few": "0 آلاف",
            "1000-count-many": "0 ألف",
            "1000-count-other": "0 ألف",
            "10000-count-zero": "00 ألف",
            "10000-count-one": "00 ألف",
            "10000-count-two": "00 ألف",
            "10000-count-few": "00 ألف",
            "10000-count-many": "00 ألف",
            "10000-count-other": "00 ألف",
            "100000-count-zero": "000 ألف",
            "100000-count-one": "000 ألف",
            "100000-count-two": "000 ألف",
            "100000-count-few": "000 ألف",
            "100000-count-many": "000 ألف",
            "100000-count-other": "000 ألف",
            "1000000-count-zero": "0 مليون",
            "1000000-count-one": "0 مليون",
            "1000000-count-two": "0 مليون",
            "1000000-count-few": "0 ملايين",
            "1000000-count-many": "0 مليون",
            "1000000-count-other": "0 مليون",
            "10000000-count-zero": "00 مليون",
            "10000000-count-one": "00 مليون",
            "10000000-count-two": "00 مليون",
            "10000000-count-few": "00 ملايين",
            "10000000-count-many": "00 مليون",
            "10000000-count-other": "00 مليون",
            "100000000-count-zero": "000 مليون",
            "100000000-count-one": "000 مليون",
            "100000000-count-two": "000 مليون",
            "100000000-count-few": "000 مليون",
            "100000000-count-many": "000 مليون",
            "100000000-count-other": "000 مليون",
            "1000000000-count-zero": "0 مليار",
            "1000000000-count-one": "0 مليار",
            "1000000000-count-two": "0 مليار",
            "1000000000-count-few": "0 مليار"
        }
    }
}

```

```

        "1000000000-count-many": "0 مليار",
        "1000000000-count-other": "0 مليار",
        "1000000000-count-zero": "00 مليار",
        "1000000000-count-one": "00 مليار",
        "1000000000-count-two": "00 مليار",
        "1000000000-count-few": "00 مليار",
        "1000000000-count-many": "00 مليار",
        "1000000000-count-other": "00 مليار",
        "10000000000-count-zero": "000 مليار",
        "10000000000-count-one": "000 مليار",
        "10000000000-count-two": "000 مليار",
        "10000000000-count-few": "000 مليار",
        "10000000000-count-many": "000 مليار",
        "10000000000-count-other": "000 مليار",
        "100000000000-count-zero": "0 ترليون",
        "100000000000-count-one": "0 ترليون",
        "100000000000-count-two": "0 ترليون",
        "100000000000-count-few": "0 ترليون",
        "100000000000-count-many": "0 ترليون",
        "100000000000-count-other": "0 ترليون",
        "1000000000000-count-zero": "00 ترليون",
        "1000000000000-count-one": "00 ترليون",
        "1000000000000-count-two": "00 ترليون",
        "1000000000000-count-few": "00 ترليون",
        "1000000000000-count-many": "00 ترليون",
        "1000000000000-count-other": "00 ترليون",
        "10000000000000-count-zero": "000 ترليون",
        "10000000000000-count-one": "000 ترليون",
        "10000000000000-count-two": "000 ترليون",
        "10000000000000-count-few": "000 ترليون",
        "10000000000000-count-many": "000 ترليون",
        "10000000000000-count-other": "000 ترليون"
    },
    },
    "short": {
        "decimalFormat": {
            "1000-count-zero": "0 ألف",
            "1000-count-one": "0 ألف",
            "1000-count-two": "0 ألف",
            "1000-count-few": "0 آلاف",
            "1000-count-many": "0 ألف",
            "1000-count-other": "0 ألف",
            "10000-count-zero": "00 ألف",
            "10000-count-one": "00 ألف",
            "10000-count-two": "00 ألف",
            "10000-count-few": "00 ألف",
            "10000-count-many": "00 ألف",
            "10000-count-other": "00 ألف",
            "100000-count-zero": "000 ألف",
            "100000-count-one": "000 ألف",
            "100000-count-two": "000 ألف",
            "100000-count-few": "000 ألف",
            "100000-count-many": "000 ألف",
            "100000-count-other": "000 ألف",
            "1000000-count-zero": "0 مليون",
            "1000000-count-one": "0 مليون",
            "1000000-count-two": "0 مليون",

```

```

        "1000000-count-few": "0 مليون",
        "1000000-count-many": "0 مليون",
        "1000000-count-other": "0 مليون",
        "10000000-count-zero": "00 مليون",
        "10000000-count-one": "00 مليون",
        "10000000-count-two": "00 مليون",
        "10000000-count-few": "00 مليون",
        "10000000-count-many": "00 مليون",
        "10000000-count-other": "00 مليون",
        "100000000-count-zero": "000 مليون",
        "100000000-count-one": "000 مليون",
        "100000000-count-two": "000 مليون",
        "100000000-count-few": "000 مليون",
        "100000000-count-many": "000 مليون",
        "100000000-count-other": "000 مليون",
        "1000000000-count-zero": "0 مليار",
        "1000000000-count-one": "0 مليار",
        "1000000000-count-two": "0 مليار",
        "1000000000-count-few": "0 مليار",
        "1000000000-count-many": "0 مليار",
        "1000000000-count-other": "0 مليار",
        "10000000000-count-zero": "00 مليار",
        "10000000000-count-one": "00 مليار",
        "10000000000-count-two": "00 مليار",
        "10000000000-count-few": "00 مليار",
        "10000000000-count-many": "00 مليار",
        "10000000000-count-other": "00 مليار",
        "100000000000-count-zero": "000 مليار",
        "100000000000-count-one": "000 مليار",
        "100000000000-count-two": "000 مليار",
        "100000000000-count-few": "000 مليار",
        "100000000000-count-many": "000 مليار",
        "100000000000-count-other": "000 مليار",
        "1000000000000-count-zero": "0 ترليون",
        "1000000000000-count-one": "0 ترليون",
        "1000000000000-count-two": "0 ترليون",
        "1000000000000-count-few": "0 ترليون",
        "1000000000000-count-many": "0 ترليون",
        "1000000000000-count-other": "0 ترليون",
        "10000000000000-count-zero": "00 ترليون",
        "10000000000000-count-one": "00 ترليون",
        "10000000000000-count-two": "00 ترليون",
        "10000000000000-count-few": "00 ترليون",
        "10000000000000-count-many": "00 ترليون",
        "10000000000000-count-other": "00 ترليون",
        "100000000000000-count-zero": "000 ترليون",
        "100000000000000-count-one": "000 ترليون",
        "100000000000000-count-two": "000 ترليون",
        "100000000000000-count-few": "000 ترليون",
        "100000000000000-count-many": "000 ترليون",
        "100000000000000-count-other": "000 ترليون"
    },
    },
    "scientificFormats-numberSystem-arab": {
        "standard": "#E0"
    },

```



```

"scientificFormats-numberSystem-latn": {
  "standard": "#E0"
},
"percentFormats-numberSystem-arab": {
  "standard": "#,##0 %"
},
"percentFormats-numberSystem-latn": {
  "standard": "#,##0%"
},
"currencyFormats-numberSystem-arab": {
  "currencySpacing": {
    "beforeCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    },
    "afterCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    }
  },
  "standard": "#,##0.00 ¤",
  "accounting": "#,##0.00 ¤",
  "unitPattern-count-zero": "{0} {1}",
  "unitPattern-count-one": "{0} {1}",
  "unitPattern-count-two": "{0} {1}",
  "unitPattern-count-few": "{0} {1}",
  "unitPattern-count-many": "{0} {1}",
  "unitPattern-count-other": "{0} {1}"
},
"currencyFormats-numberSystem-latn": {
  "currencySpacing": {
    "beforeCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    },
    "afterCurrency": {
      "currencyMatch": "[:^S:]",
      "surroundingMatch": "[:digit:]",
      "insertBetween": " "
    }
  },
  "standard": "¤ #,##0.00",
  "accounting": "¤#,##0.00; (¤#,##0.00)",
  "short": {
    "standard": {
      "1000-count-zero": "¤ 0 ألف",
      "1000-count-one": "¤ 0 ألف",
      "1000-count-two": "¤ 0 ألف",
      "1000-count-few": "¤ 0 ألف",
      "1000-count-many": "¤ 0 ألف",
      "1000-count-other": "¤ 0 ألف",
      "10000-count-zero": "¤ 00 ألف",
      "10000-count-one": "¤ 00 ألف",
      "10000-count-two": "¤ 00 ألف",

```

```
"10000-count-few": "ألف 00",
"10000-count-many": "ألف 00",
"10000-count-other": "ألف 00",
"100000-count-zero": "ألف 000",
"100000-count-one": "ألف 000",
"100000-count-two": "ألف 000",
"100000-count-few": "ألف 000",
"100000-count-many": "ألف 000",
"100000-count-other": "ألف 000",
"1000000-count-zero": "مليون 0",
"1000000-count-one": "مليون 0",
"1000000-count-two": "مليون 0",
"1000000-count-few": "مليون 0",
"1000000-count-many": "مليون 0",
"1000000-count-other": "مليون 0",
"10000000-count-zero": "مليون 00",
"10000000-count-one": "مليون 00",
"10000000-count-two": "مليون 00",
"10000000-count-few": "مليون 00",
"10000000-count-many": "مليون 00",
"10000000-count-other": "مليون 00",
"100000000-count-zero": "مليون 000",
"100000000-count-one": "مليون 000",
"100000000-count-two": "مليون 000",
"100000000-count-few": "مليون 000",
"100000000-count-many": "مليون 000",
"100000000-count-other": "مليون 000",
"1000000000-count-zero": "مليار 0",
"1000000000-count-one": "مليار 0",
"1000000000-count-two": "مليار 0",
"1000000000-count-few": "مليار 0",
"1000000000-count-many": "مليار 0",
"1000000000-count-other": "مليار 0",
"10000000000-count-zero": "مليار 00",
"10000000000-count-one": "مليار 00",
"10000000000-count-two": "مليار 00",
"10000000000-count-few": "مليار 00",
"10000000000-count-many": "مليار 00",
"10000000000-count-other": "مليار 00",
"100000000000-count-zero": "مليار 000",
"100000000000-count-one": "مليار 000",
"100000000000-count-two": "مليار 000",
"100000000000-count-few": "مليار 000",
"100000000000-count-many": "مليار 000",
"100000000000-count-other": "مليار 000",
"1000000000000-count-zero": "ترليون 0",
"1000000000000-count-one": "ترليون 0",
"1000000000000-count-two": "ترليون 0",
"1000000000000-count-few": "ترليون 0",
"1000000000000-count-many": "ترليون 0",
"1000000000000-count-other": "ترليون 0",
"10000000000000-count-zero": "ترليون 00",
"10000000000000-count-one": "ترليون 00",
"10000000000000-count-two": "ترليون 00",
"10000000000000-count-few": "ترليون 00",
"10000000000000-count-many": "ترليون 00",
"10000000000000-count-other": "ترليون 00",
```

```

        "1000000000000000-count-zero": "١٠٠٠ ترليون",
        "1000000000000000-count-one": "١٠٠٠ ترليون",
        "1000000000000000-count-two": "١٠٠٠ ترليون",
        "1000000000000000-count-few": "١٠٠٠ ترليون",
        "1000000000000000-count-many": "١٠٠٠ ترليون",
        "1000000000000000-count-other": "١٠٠٠ ترليون"
    },
    },
    "unitPattern-count-zero": "{0} {1}",
    "unitPattern-count-one": "{0} {1}",
    "unitPattern-count-two": "{0} {1}",
    "unitPattern-count-few": "{0} {1}",
    "unitPattern-count-many": "{0} {1}",
    "unitPattern-count-other": "{0} {1}"
},
"miscPatterns-numberSystem-arab": {
    "atLeast": "+{0}",
    "range": "{0}-{1}"
},
"miscPatterns-numberSystem-latn": {
    "atLeast": "+{0}",
    "range": "{0}-{1}"
},
"minimalPairs": {
    "pluralMinimalPairs-count-zero": "{0} كتاب",
    "pluralMinimalPairs-count-one": "ولد واحد حضر",
    "pluralMinimalPairs-count-two": "ولدان حضرا",
    "pluralMinimalPairs-count-few": "{0} أولاد حضروا",
    "pluralMinimalPairs-count-many": "{0} ولدًا حضروا",
    "pluralMinimalPairs-count-other": "{0} ولد حضروا",
    "other": "اتجه إلى المنعطف الـ {0} يمينًا."
}
}
}
}
}
}

```

NUMBERS.JSX

```

{
    "main";
    {
        "ar";
        {
            "identity";
            {
                "version";
                {
                    "_number";
                    "$Revision: 13686 $",
                    "_cldrVersion";
                    "32";
                }
                "language";
                "ar";
            }
        }
    }
}

```

```

"numbers";
{
  "defaultNumberingSystem";
  "arab",
    "otherNumberingSystems";
  {
    "native";
    "arab";
  }
  "minimumGroupingDigits";
  "1",
    "symbols-numberSystem-arab";
  {
    "decimal";
    ",",
      "group";
    ",",
      "list";
    ":",
      "percentSign";
    "%",
      "plusSign";
    "+",
      "minusSign";
    "-",
      "exponential";
    "١٠",
      "superscriptingExponent";
    "×",
      "perMille";
    "‰",
      "infinity";
    "∞",
      "nan";
    "ليس رقم",
      "timeSeparator";
    ":";
  }
  "symbols-numberSystem-latn";
  {
    "decimal";
    ".",
      "group";
    ",",
      "list";
    ";",
      "percentSign";
    "%",
      "plusSign";
    "+",
      "minusSign";
    "-",
      "exponential";
    "E",
      "superscriptingExponent";
    "×",
      "perMille";
  }
}

```

```

        "%",
        "infinity";
    "∞",
    "nan";
    "ليس رقمًا",
    "timeSeparator";
    ":";
}
"decimalFormats-numberSystem-arab";
{
    "standard";
    "#,##0.###",
    "long";
    {
        "decimalFormat";
        {
            "1000-count-zero";
            "0 ألف",
            "1000-count-one";
            "0 ألف",
            "1000-count-two";
            "0 ألف",
            "1000-count-few";
            "0 آلاف",
            "1000-count-many";
            "0 ألف",
            "1000-count-other";
            "0 ألف",
            "10000-count-zero";
            "00 ألف",
            "10000-count-one";
            "00 ألف",
            "10000-count-two";
            "00 ألف",
            "10000-count-few";
            "00 ألف",
            "10000-count-many";
            "00 ألف",
            "10000-count-other";
            "00 ألف",
            "100000-count-zero";
            "000 ألف",
            "100000-count-one";
            "000 ألف",
            "100000-count-two";
            "000 ألف",
            "100000-count-few";
            "000 ألف",
            "100000-count-many";
            "000 ألف",
            "100000-count-other";
            "000 ألف",
            "1000000-count-zero";
            "0 مليون",
            "1000000-count-one";
            "0 مليون",
            "1000000-count-two";

```

```

"0 مليون",
  "1000000-count-few";
"0 ملايين",
  "1000000-count-many";
"0 مليون",
  "1000000-count-other";
"0 مليون",
  "1000000-count-zero";
"00 مليون",
  "10000000-count-one";
"00 مليون",
  "10000000-count-two";
"00 مليون",
  "10000000-count-few";
"00 ملايين",
  "10000000-count-many";
"00 مليون",
  "10000000-count-other";
"00 مليون",
  "100000000-count-zero";
"000 مليون",
  "100000000-count-one";
"000 مليون",
  "100000000-count-two";
"000 مليون",
  "100000000-count-few";
"000 مليون",
  "100000000-count-many";
"000 مليون",
  "100000000-count-other";
"000 مليون",
  "1000000000-count-zero";
"0 مليار",
  "1000000000-count-one";
"0 مليار",
  "1000000000-count-two";
"0 مليار",
  "1000000000-count-few";
"0 مليار",
  "1000000000-count-many";
"0 مليار",
  "1000000000-count-other";
"0 مليار",
  "10000000000-count-zero";
"00 مليار",
  "10000000000-count-one";
"00 مليار",
  "10000000000-count-two";
"00 مليار",
  "10000000000-count-few";
"00 مليار",
  "10000000000-count-many";
"00 مليار",
  "10000000000-count-other";
"00 مليار",
  "100000000000-count-zero";
"000 مليار",

```

```

        "100000000000-count-one";
        "000 مليار",
        "100000000000-count-two";
        "000 مليار",
        "100000000000-count-few";
        "000 مليار",
        "100000000000-count-many";
        "000 مليار",
        "100000000000-count-other";
        "000 مليار",
        "100000000000-count-zero";
        "0 ترليون",
        "100000000000-count-one";
        "0 ترليون",
        "100000000000-count-two";
        "0 ترليون",
        "100000000000-count-few";
        "0 ترليون",
        "100000000000-count-many";
        "0 ترليون",
        "100000000000-count-other";
        "0 ترليون",
        "100000000000-count-zero";
        "00 ترليون",
        "100000000000-count-one";
        "00 ترليون",
        "100000000000-count-two";
        "00 ترليون",
        "100000000000-count-few";
        "00 ترليون",
        "100000000000-count-many";
        "00 ترليون",
        "100000000000-count-other";
        "00 ترليون",
        "100000000000-count-zero";
        "000 ترليون",
        "100000000000-count-one";
        "000 ترليون",
        "100000000000-count-two";
        "000 ترليون",
        "100000000000-count-few";
        "000 ترليون",
        "100000000000-count-many";
        "000 ترليون",
        "100000000000-count-other";
        "000 ترليون";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-zero";
        "0 ألف",
        "1000-count-one";
        "0 ألف",
        "1000-count-two";
    }
}

```

```
"0 ألف",
  "1000-count-few";
"0 آلاف",
  "1000-count-many";
"0 ألف",
  "1000-count-other";
"0 ألف",
  "10000-count-zero";
"00 ألف",
  "10000-count-one";
"00 ألف",
  "10000-count-two";
"00 ألف",
  "10000-count-few";
"00 ألف",
  "10000-count-many";
"00 ألف",
  "10000-count-other";
"00 ألف",
  "100000-count-zero";
"000 ألف",
  "100000-count-one";
"000 ألف",
  "100000-count-two";
"000 ألف",
  "100000-count-few";
"000 ألف",
  "100000-count-many";
"000 ألف",
  "100000-count-other";
"000 ألف",
  "1000000-count-zero";
"0 مليون",
  "1000000-count-one";
"0 مليون",
  "1000000-count-two";
"0 مليون",
  "1000000-count-few";
"0 مليون",
  "1000000-count-many";
"0 مليون",
  "1000000-count-other";
"0 مليون",
  "10000000-count-zero";
"00 مليون",
  "10000000-count-one";
"00 مليون",
  "10000000-count-two";
"00 مليون",
  "10000000-count-few";
"00 مليون",
  "10000000-count-many";
"00 مليون",
  "10000000-count-other";
"00 مليون",
  "100000000-count-zero";
"000 مليون",
```



```

        "100000000-count-one";
    "000 مليون",
        "100000000-count-two";
    "000 مليون",
        "100000000-count-few";
    "000 مليون",
        "100000000-count-many";
    "000 مليون",
        "100000000-count-other";
    "000 مليون",
        "1000000000-count-zero";
    "0 مليار",
        "1000000000-count-one";
    "0 مليار",
        "1000000000-count-two";
    "0 مليار",
        "1000000000-count-few";
    "0 مليار",
        "1000000000-count-many";
    "0 مليار",
        "1000000000-count-other";
    "0 مليار",
        "10000000000-count-zero";
    "00 مليار",
        "10000000000-count-one";
    "00 مليار",
        "10000000000-count-two";
    "00 مليار",
        "10000000000-count-few";
    "00 مليار",
        "10000000000-count-many";
    "00 مليار",
        "10000000000-count-other";
    "00 مليار",
        "100000000000-count-zero";
    "000 مليار",
        "100000000000-count-one";
    "000 مليار",
        "100000000000-count-two";
    "000 مليار",
        "100000000000-count-few";
    "000 مليار",
        "100000000000-count-many";
    "000 مليار",
        "100000000000-count-other";
    "000 مليار",
        "1000000000000-count-zero";
    "0 ترليون",
        "1000000000000-count-one";
    "0 ترليون",
        "1000000000000-count-two";
    "0 ترليون",
        "1000000000000-count-few";
    "0 ترليون",
        "1000000000000-count-many";
    "0 ترليون",
        "1000000000000-count-other";

```

```

        "0 ترليون",
        "10000000000000-count-zero";
        "00 ترليون",
        "10000000000000-count-one";
        "00 ترليون",
        "10000000000000-count-two";
        "00 ترليون",
        "10000000000000-count-few";
        "00 ترليون",
        "10000000000000-count-many";
        "00 ترليون",
        "10000000000000-count-other";
        "00 ترليون",
        "10000000000000-count-zero";
        "000 ترليون",
        "10000000000000-count-one";
        "000 ترليون",
        "10000000000000-count-two";
        "000 ترليون",
        "10000000000000-count-few";
        "000 ترليون",
        "10000000000000-count-many";
        "000 ترليون",
        "10000000000000-count-other";
        "000 ترليون";
    }
}
}
"decimalFormats-numberSystem-latn";
{
    "standard";
    "#,##0.###",
    "long";
    {
        "decimalFormat";
        {
            "1000-count-zero";
            "0 ألف",
            "1000-count-one";
            "0 ألف",
            "1000-count-two";
            "0 ألف",
            "1000-count-few";
            "0 آلاف",
            "1000-count-many";
            "0 ألف",
            "1000-count-other";
            "0 ألف",
            "10000-count-zero";
            "00 ألف",
            "10000-count-one";
            "00 ألف",
            "10000-count-two";
            "00 ألف",
            "10000-count-few";
            "00 ألف",
            "10000-count-many";
        }
    }
}

```

```
"00 ألف",
    "10000-count-other";
"00 ألف",
    "100000-count-zero";
"000 ألف",
    "100000-count-one";
"000 ألف",
    "100000-count-two";
"000 ألف",
    "100000-count-few";
"000 ألف",
    "100000-count-many";
"000 ألف",
    "100000-count-other";
"000 ألف",
    "1000000-count-zero";
"0 مليون",
    "1000000-count-one";
"0 مليون",
    "1000000-count-two";
"0 مليون",
    "1000000-count-few";
"0 ملايين",
    "1000000-count-many";
"0 مليون",
    "1000000-count-other";
"0 مليون",
    "10000000-count-zero";
"00 مليون",
    "10000000-count-one";
"00 مليون",
    "10000000-count-two";
"00 مليون",
    "10000000-count-few";
"00 ملايين",
    "10000000-count-many";
"00 مليون",
    "10000000-count-other";
"00 مليون",
    "100000000-count-zero";
"000 مليون",
    "100000000-count-one";
"000 مليون",
    "100000000-count-two";
"000 مليون",
    "100000000-count-few";
"000 مليون",
    "100000000-count-many";
"000 مليون",
    "100000000-count-other";
"000 مليون",
    "1000000000-count-zero";
"0 مليار",
    "1000000000-count-one";
"0 مليار",
    "1000000000-count-two";
"0 مليار",
```

```

        "1000000000-count-few";
    "0 مليار",
        "1000000000-count-many";
    "0 مليار",
        "1000000000-count-other";
    "0 مليار",
        "1000000000-count-zero";
    "00 مليار",
        "1000000000-count-one";
    "00 مليار",
        "1000000000-count-two";
    "00 مليار",
        "1000000000-count-few";
    "00 مليار",
        "1000000000-count-many";
    "00 مليار",
        "1000000000-count-other";
    "00 مليار",
        "1000000000-count-zero";
    "000 مليار",
        "1000000000-count-one";
    "000 مليار",
        "1000000000-count-two";
    "000 مليار",
        "1000000000-count-few";
    "000 مليار",
        "1000000000-count-many";
    "000 مليار",
        "1000000000-count-other";
    "000 مليار",
        "1000000000-count-zero";
    "0 ترليون",
        "1000000000000-count-one";
    "0 ترليون",
        "1000000000000-count-two";
    "0 ترليون",
        "1000000000000-count-few";
    "0 ترليون",
        "1000000000000-count-many";
    "0 ترليون",
        "1000000000000-count-other";
    "0 ترليون",
        "1000000000000-count-zero";
    "00 ترليون",
        "1000000000000-count-one";
    "00 ترليون",
        "1000000000000-count-two";
    "00 ترليون",
        "1000000000000-count-few";
    "00 ترليون",
        "1000000000000-count-many";
    "00 ترليون",
        "1000000000000-count-other";
    "00 ترليون",
        "1000000000000-count-zero";
    "000 ترليون",
        "1000000000000-count-one";

```

```

        "000 ترليون",
        "1000000000000000-count-two";
        "000 ترليون",
        "1000000000000000-count-few";
        "000 ترليون",
        "1000000000000000-count-many";
        "000 ترليون",
        "1000000000000000-count-other";
        "000 ترليون";
    }
}
"short";
{
    "decimalFormat";
    {
        "1000-count-zero";
        "0 ألف",
        "1000-count-one";
        "0 ألف",
        "1000-count-two";
        "0 ألف",
        "1000-count-few";
        "0 آلاف",
        "1000-count-many";
        "0 ألف",
        "1000-count-other";
        "0 ألف",
        "10000-count-zero";
        "00 ألف",
        "10000-count-one";
        "00 ألف",
        "10000-count-two";
        "00 ألف",
        "10000-count-few";
        "00 ألف",
        "10000-count-many";
        "00 ألف",
        "10000-count-other";
        "00 ألف",
        "100000-count-zero";
        "000 ألف",
        "100000-count-one";
        "000 ألف",
        "100000-count-two";
        "000 ألف",
        "100000-count-few";
        "000 ألف",
        "100000-count-many";
        "000 ألف",
        "100000-count-other";
        "000 ألف",
        "1000000-count-zero";
        "0 مليون",
        "1000000-count-one";
        "0 مليون",
        "1000000-count-two";
        "0 مليون",
    }
}

```

```

        "1000000-count-few";
    "0 مليون",
        "1000000-count-many";
    "0 مليون",
        "1000000-count-other";
    "0 مليون",
        "10000000-count-zero";
    "00 مليون",
        "10000000-count-one";
    "00 مليون",
        "10000000-count-two";
    "00 مليون",
        "10000000-count-few";
    "00 مليون",
        "10000000-count-many";
    "00 مليون",
        "10000000-count-other";
    "00 مليون",
        "100000000-count-zero";
    "000 مليون",
        "100000000-count-one";
    "000 مليون",
        "100000000-count-two";
    "000 مليون",
        "100000000-count-few";
    "000 مليون",
        "100000000-count-many";
    "000 مليون",
        "100000000-count-other";
    "000 مليون",
        "1000000000-count-zero";
    "0 مليار",
        "1000000000-count-one";
    "0 مليار",
        "1000000000-count-two";
    "0 مليار",
        "1000000000-count-few";
    "0 مليار",
        "1000000000-count-many";
    "0 مليار",
        "1000000000-count-other";
    "0 مليار",
        "10000000000-count-zero";
    "00 مليار",
        "10000000000-count-one";
    "00 مليار",
        "10000000000-count-two";
    "00 مليار",
        "10000000000-count-few";
    "00 مليار",
        "10000000000-count-many";
    "00 مليار",
        "10000000000-count-other";
    "00 مليار",
        "100000000000-count-zero";
    "000 مليار",
        "100000000000-count-one";

```

```

        "000 مليار",
        "100000000000-count-two";
        "000 مليار",
        "100000000000-count-few";
        "000 مليار",
        "100000000000-count-many";
        "000 مليار",
        "100000000000-count-other";
        "000 مليار",
        "100000000000-count-zero";
        "0 ترليون",
        "1000000000000-count-one";
        "0 ترليون",
        "1000000000000-count-two";
        "0 ترليون",
        "1000000000000-count-few";
        "0 ترليون",
        "1000000000000-count-many";
        "0 ترليون",
        "1000000000000-count-other";
        "0 ترليون",
        "1000000000000-count-zero";
        "00 ترليون",
        "10000000000000-count-one";
        "00 ترليون",
        "10000000000000-count-two";
        "00 ترليون",
        "10000000000000-count-few";
        "00 ترليون",
        "10000000000000-count-many";
        "00 ترليون",
        "10000000000000-count-other";
        "00 ترليون",
        "10000000000000-count-zero";
        "000 ترليون",
        "100000000000000-count-one";
        "000 ترليون",
        "100000000000000-count-two";
        "000 ترليون",
        "100000000000000-count-few";
        "000 ترليون",
        "100000000000000-count-many";
        "000 ترليون",
        "100000000000000-count-other";
        "000 ترليون";
    }
}
}
"scientificFormats-numberSystem-arab";
{
    "standard";
    "#E0";
}
"scientificFormats-numberSystem-latn";
{
    "standard";
    "#E0";
}

```

```

    }
    "percentFormats-numberSystem-arab";
    {
        "standard";
        "#,##0 %";
    }
    "percentFormats-numberSystem-latn";
    {
        "standard";
        "#,##0%";
    }
    "currencyFormats-numberSystem-arab";
    {
        "currencySpacing";
        {
            "beforeCurrency";
            {
                "currencyMatch";
                "[:^S:]",
                "surroundingMatch";
                "[:digit:]",
                "insertBetween";
                " ";
            }
            "afterCurrency";
            {
                "currencyMatch";
                "[:^S:]",
                "surroundingMatch";
                "[:digit:]",
                "insertBetween";
                " ";
            }
        }
        "standard";
        "#,##0.00 ¤",
        "accounting";
        "#,##0.00 ¤",
        "unitPattern-count-zero";
        "{0} {1}",
        "unitPattern-count-one";
        "{0} {1}",
        "unitPattern-count-two";
        "{0} {1}",
        "unitPattern-count-few";
        "{0} {1}",
        "unitPattern-count-many";
        "{0} {1}",
        "unitPattern-count-other";
        "{0} {1}";
    }
    "currencyFormats-numberSystem-latn";
    {
        "currencySpacing";
        {
            "beforeCurrency";
            {

```



```

        "currencyMatch";
        "[:^S:]",
        "surroundingMatch";
        "[:digit:]",
        "insertBetween";
        " ";
    }
    "afterCurrency";
    {
        "currencyMatch";
        "[:^S:]",
        "surroundingMatch";
        "[:digit:]",
        "insertBetween";
        " ";
    }
}
"standard";
"¤ #,##0.00",
    "accounting";
"¤#,##0.00; (¤#,##0.00) ",
    "short";
{
    "standard";
    {
        "1000-count-zero";
        "¤ 0 ألف",
        "1000-count-one";
        "¤ 0 ألف",
        "1000-count-two";
        "¤ 0 ألف",
        "1000-count-few";
        "¤ 0 ألف",
        "1000-count-many";
        "¤ 0 ألف",
        "1000-count-other";
        "¤ 0 ألف",
        "10000-count-zero";
        "¤ 00 ألف",
        "10000-count-one";
        "¤ 00 ألف",
        "10000-count-two";
        "¤ 00 ألف",
        "10000-count-few";
        "¤ 00 ألف",
        "10000-count-many";
        "¤ 00 ألف",
        "10000-count-other";
        "¤ 00 ألف",
        "100000-count-zero";
        "¤ 000 ألف",
        "100000-count-one";
        "¤ 000 ألف",
        "100000-count-two";
        "¤ 000 ألف",
        "100000-count-few";
        "¤ 000 ألف",
    }
}

```

```

        "100000-count-many";
        "۰ 000 ألف",
        "100000-count-other";
        "۰ 000 ألف",
        "1000000-count-zero";
        "۰ 0 مليون",
        "1000000-count-one";
        "۰ 0 مليون",
        "1000000-count-two";
        "۰ 0 مليون",
        "1000000-count-few";
        "۰ 0 مليون",
        "1000000-count-many";
        "۰ 0 مليون",
        "1000000-count-other";
        "۰ 0 مليون",
        "10000000-count-zero";
        "۰ 00 مليون",
        "10000000-count-one";
        "۰ 00 مليون",
        "10000000-count-two";
        "۰ 00 مليون",
        "10000000-count-few";
        "۰ 00 مليون",
        "10000000-count-many";
        "۰ 00 مليون",
        "10000000-count-other";
        "۰ 00 مليون",
        "100000000-count-zero";
        "۰ 000 مليون",
        "100000000-count-one";
        "۰ 000 مليون",
        "100000000-count-two";
        "۰ 000 مليون",
        "100000000-count-few";
        "۰ 000 مليون",
        "100000000-count-many";
        "۰ 000 مليون",
        "100000000-count-other";
        "۰ 000 مليون",
        "1000000000-count-zero";
        "۰ 0 مليار",
        "1000000000-count-one";
        "۰ 0 مليار",
        "1000000000-count-two";
        "۰ 0 مليار",
        "1000000000-count-few";
        "۰ 0 مليار",
        "1000000000-count-many";
        "۰ 0 مليار",
        "1000000000-count-other";
        "۰ 0 مليار",
        "10000000000-count-zero";
        "۰ 00 مليار",
        "10000000000-count-one";
        "۰ 00 مليار",
        "10000000000-count-two";

```

```

    "٠٠ مليار",
    "10000000000-count-few";
    "٠٠ مليار",
    "10000000000-count-many";
    "٠٠ مليار",
    "10000000000-count-other";
    "٠٠ مليار",
    "10000000000-count-zero";
    "٠٠٠ مليار",
    "100000000000-count-one";
    "٠٠٠ مليار",
    "100000000000-count-two";
    "٠٠٠ مليار",
    "100000000000-count-few";
    "٠٠٠ مليار",
    "100000000000-count-many";
    "٠٠٠ مليار",
    "100000000000-count-other";
    "٠٠٠ مليار",
    "100000000000-count-zero";
    "٠ ترليون",
    "1000000000000-count-one";
    "٠ ترليون",
    "1000000000000-count-two";
    "٠ ترليون",
    "1000000000000-count-few";
    "٠ ترليون",
    "1000000000000-count-many";
    "٠ ترليون",
    "1000000000000-count-other";
    "٠ ترليون",
    "10000000000000-count-zero";
    "٠٠ ترليون",
    "10000000000000-count-one";
    "٠٠ ترليون",
    "10000000000000-count-two";
    "٠٠ ترليون",
    "10000000000000-count-few";
    "٠٠ ترليون",
    "10000000000000-count-many";
    "٠٠ ترليون",
    "10000000000000-count-other";
    "٠٠ ترليون",
    "10000000000000-count-zero";
    "٠٠٠ ترليون",
    "100000000000000-count-one";
    "٠٠٠ ترليون",
    "100000000000000-count-two";
    "٠٠٠ ترليون",
    "100000000000000-count-few";
    "٠٠٠ ترليون",
    "100000000000000-count-many";
    "٠٠٠ ترليون",
    "100000000000000-count-other";
    "٠٠٠ ترليون";
  }
}

```

```

        "unitPattern-count-zero";
        "{0} {1}",
        "unitPattern-count-one";
        "{0} {1}",
        "unitPattern-count-two";
        "{0} {1}",
        "unitPattern-count-few";
        "{0} {1}",
        "unitPattern-count-many";
        "{0} {1}",
        "unitPattern-count-other";
        "{0} {1}";
    }
    "miscPatterns-numberSystem-arab";
    {
        "atLeast";
        "+{0}",
        "range";
        "{0}-{1}";
    }
    "miscPatterns-numberSystem-latn";
    {
        "atLeast";
        "+{0}",
        "range";
        "{0}-{1}";
    }
    "minimalPairs";
    {
        "pluralMinimalPairs-count-zero";
        "{0} كتاب",
        "pluralMinimalPairs-count-one";
        "ولد واحد حضر",
        "pluralMinimalPairs-count-two";
        "ولدان حضرا",
        "pluralMinimalPairs-count-few";
        "{0} أولاد حضروا",
        "pluralMinimalPairs-count-many";
        "{0} ولدا حضروا",
        "pluralMinimalPairs-count-other";
        "{0} ولد حضروا",
        "other";
        "اتجه إلى المنعطف الـ {0} يميناً";
    }
    }
    }
}

```

TIMEZONENAMES.JSON

```

{
  "main": {
    "ar": {
      "identity": {
        "version": {

```

```

    "_number": "$Revision: 13686 $",
    "_cldrVersion": "32"
  },
  "language": "ar"
},
"dates": {
  "timeZoneNames": {
    "hourFormat": "+HH:mm;-HH:mm",
    "gmtFormat": "0{غرينتش",
    "gmtZeroFormat": "غرينتش",
    "regionFormat": "0{توقيت",
    "regionFormat-type-daylight": "0{الصيفي",
    "regionFormat-type-standard": "0{الرسمي",
    "fallbackFormat": "{1} ({0})",
    "zone": {
      "America": {
        "Adak": {
          "exemplarCity": "أداك"
        },
        "Anchorage": {
          "exemplarCity": "أنشوراج"
        },
        "Anguilla": {
          "exemplarCity": "أنغويلا"
        },
        "Antigua": {
          "exemplarCity": "أنتيغوا"
        },
        "Araguaina": {
          "exemplarCity": "أروجوانيا"
        },
        "Argentina": {
          "Rio_Gallegos": {
            "exemplarCity": "ريو جالييوس"
          },
          "San_Juan": {
            "exemplarCity": "سان خوان"
          },
          "Ushuaia": {
            "exemplarCity": "أشوا"
          },
          "La_Rioja": {
            "exemplarCity": "لا ريوجا"
          },
          "San_Luis": {
            "exemplarCity": "سان لويس"
          },
          "Salta": {
            "exemplarCity": "سالطا"
          },
          "Tucuman": {
            "exemplarCity": "تاكمان"
          }
        },
        "Aruba": {
          "exemplarCity": "أروبا"
        }
      }
    }
  }
}

```

```
"Asuncion": {
  "exemplarCity": "أسونسيون"
},
"Bahia": {
  "exemplarCity": "باهيا"
},
"Bahia_Banderas": {
  "exemplarCity": "باهيا بانديراس"
},
"Barbados": {
  "exemplarCity": "بربادوس"
},
"Belem": {
  "exemplarCity": "بلم"
},
"Belize": {
  "exemplarCity": "بليز"
},
"Blanc-Sablon": {
  "exemplarCity": "بلانك-سابلون"
},
"Boa_Vista": {
  "exemplarCity": "باو فيستا"
},
"Bogota": {
  "exemplarCity": "بوغوتا"
},
"Boise": {
  "exemplarCity": "بويس"
},
"Buenos_Aires": {
  "exemplarCity": "بوينوس آيرس"
},
"Cambridge_Bay": {
  "exemplarCity": "كامبرديج باي"
},
"Campo_Grande": {
  "exemplarCity": "كومبو جراند"
},
"Cancun": {
  "exemplarCity": "كانكون"
},
"Caracas": {
  "exemplarCity": "كاراكاس"
},
"Catamarca": {
  "exemplarCity": "كاتاماركا"
},
"Cayenne": {
  "exemplarCity": "كايين"
},
"Cayman": {
  "exemplarCity": "كايمان"
},
"Chicago": {
  "exemplarCity": "شيكاغو"
},
}
```

```
"Chihuahua": {
  "exemplarCity": "تشياواوا",
},
"Coral_Harbour": {
  "exemplarCity": "كورال هاربر",
},
"Cordoba": {
  "exemplarCity": "كوردوبا",
},
"Costa_Rica": {
  "exemplarCity": "كوستاريكا",
},
"Creston": {
  "exemplarCity": "كريستون",
},
"Cuiaba": {
  "exemplarCity": "كيابا",
},
"Curacao": {
  "exemplarCity": "كوراساو",
},
"Danmarkshavn": {
  "exemplarCity": "دانمرك شافن",
},
"Dawson": {
  "exemplarCity": "داوسان",
},
"Dawson_Creek": {
  "exemplarCity": "داوسن كريك",
},
"Denver": {
  "exemplarCity": "دنفر",
},
"Detroit": {
  "exemplarCity": "ديترويت",
},
"Dominica": {
  "exemplarCity": "دومينيكا",
},
"Edmonton": {
  "exemplarCity": "ايدمونتون",
},
"Eirunepe": {
  "exemplarCity": "ايرونبي",
},
"El_Salvador": {
  "exemplarCity": "السلفادور",
},
"Fort_Nelson": {
  "exemplarCity": "فورت نيلسون",
},
"Fortaleza": {
  "exemplarCity": "فورتاليزا",
},
"Glace_Bay": {
  "exemplarCity": "جلاس باي",
},
}
```

```

"Godthab": {
  "exemplarCity": "غودثاب"
},
"Goose_Bay": {
  "exemplarCity": "جوس باي"
},
"Grand_Turk": {
  "exemplarCity": "غراند ترك"
},
"Grenada": {
  "exemplarCity": "غرينادا"
},
"Guadeloupe": {
  "exemplarCity": "غوادلوب"
},
"Guatemala": {
  "exemplarCity": "غواتيمالا"
},
"Guayaquil": {
  "exemplarCity": "غواياكويل"
},
"Guyana": {
  "exemplarCity": "غيانا"
},
"Halifax": {
  "exemplarCity": "هاليفاكس"
},
"Havana": {
  "exemplarCity": "هافانا"
},
"Hermosillo": {
  "exemplarCity": "هيرموسيلو"
},
"Indiana": {
  "Vincennes": {
    "exemplarCity": "فينسينس"
  },
  "Petersburg": {
    "exemplarCity": "بيتربيرغ"
  },
  "Tell_City": {
    "exemplarCity": "مدينة تل، إنديانا"
  },
  "Knox": {
    "exemplarCity": "كونكس"
  },
  "Winamac": {
    "exemplarCity": "ويناماك"
  },
  "Marengo": {
    "exemplarCity": "مارنجو"
  },
  "Vevay": {
    "exemplarCity": "فيفاي"
  }
},
"Indianapolis": {

```



```

    "exemplarCity": "إنديانابوليس"
  },
  "Inuvik": {
    "exemplarCity": "اينوفيك"
  },
  "Iqaluit": {
    "exemplarCity": "اكويلت"
  },
  "Jamaica": {
    "exemplarCity": "جاماىكا"
  },
  "Jujuy": {
    "exemplarCity": "جوجو"
  },
  "Juneau": {
    "exemplarCity": "جونى"
  },
  "Kentucky": {
    "Monticello": {
      "exemplarCity": "مونتيسيلو"
    }
  },
  "Kralendijk": {
    "exemplarCity": "كرالنديك"
  },
  "La_Paz": {
    "exemplarCity": "لا باز"
  },
  "Lima": {
    "exemplarCity": "ليما"
  },
  "Los_Angeles": {
    "exemplarCity": "لوس انجلوس"
  },
  "Louisville": {
    "exemplarCity": "لويس فيل"
  },
  "Lower_Princes": {
    "exemplarCity": "حي الأمير السفلي"
  },
  "Maceio": {
    "exemplarCity": "ماشيو"
  },
  "Managua": {
    "exemplarCity": "ماناغوا"
  },
  "Manaus": {
    "exemplarCity": "ماناوس"
  },
  "Marigot": {
    "exemplarCity": "ماريغوت"
  },
  "Martinique": {
    "exemplarCity": "المارتينيك"
  },
  "Matamoros": {
    "exemplarCity": "ماتاموروس"
  }

```

```
    },
    "Mazatlan": {
      "exemplarCity": "مازاتلان"
    },
    "Mendoza": {
      "exemplarCity": "ميندوزا"
    },
    "Menominee": {
      "exemplarCity": "مينوميني"
    },
    "Merida": {
      "exemplarCity": "ميريدا"
    },
    "Metlakatla": {
      "exemplarCity": "ميتلاكاتلا"
    },
    "Mexico_City": {
      "exemplarCity": "مدينة المكسيك"
    },
    "Miquelon": {
      "exemplarCity": "مكويلون"
    },
    "Moncton": {
      "exemplarCity": "وينكتون"
    },
    "Monterrey": {
      "exemplarCity": "مونتيري"
    },
    "Montevideo": {
      "exemplarCity": "مونتيفيديو"
    },
    "Montserrat": {
      "exemplarCity": "مونتسيرات"
    },
    "Nassau": {
      "exemplarCity": "ناسو"
    },
    "New_York": {
      "exemplarCity": "نيويورك"
    },
    "Nipigon": {
      "exemplarCity": "نيبيجون"
    },
    "Nome": {
      "exemplarCity": "نوم"
    },
    "Noronha": {
      "exemplarCity": "نوروناه"
    },
    "North_Dakota": {
      "Beulah": {
        "exemplarCity": "بيولا، داكوتا الشمالية"
      },
      "New_Salem": {
        "exemplarCity": "نيو ساليم"
      },
      "Center": {
```

```

        "exemplarCity": "سنتر"
    },
    "Ojinaga": {
        "exemplarCity": "أوجيناغا"
    },
    "Panama": {
        "exemplarCity": "بنما"
    },
    "Pangnirtung": {
        "exemplarCity": "بانجینتینگ"
    },
    "Paramaribo": {
        "exemplarCity": "باراماریبو"
    },
    "Phoenix": {
        "exemplarCity": "فینکس"
    },
    "Port-au-Prince": {
        "exemplarCity": "بورت أو برنس"
    },
    "Port_of_Spain": {
        "exemplarCity": "بورت أوف سبین"
    },
    "Porto_Velho": {
        "exemplarCity": "بورتو فیلو"
    },
    "Puerto_Rico": {
        "exemplarCity": "بورتوریکو"
    },
    "Punta_Arenas": {
        "exemplarCity": "بونتأأریناز"
    },
    "Rainy_River": {
        "exemplarCity": "رانی ریفر"
    },
    "Rankin_Inlet": {
        "exemplarCity": "رانکن انلت"
    },
    "Recife": {
        "exemplarCity": "ریسیف"
    },
    "Regina": {
        "exemplarCity": "ریجینا"
    },
    "Resolute": {
        "exemplarCity": "ریزولوت"
    },
    "Rio_Branco": {
        "exemplarCity": "ریوبرانکو"
    },
    "Santa_Isabel": {
        "exemplarCity": "سانتا إیزابیل"
    },
    "Santarem": {
        "exemplarCity": "سانتاریم"
    },

```

```
"Santiago": {
  "exemplarCity": "سانتياغو"
},
"Santo_Domingo": {
  "exemplarCity": "سانتو دومينغو"
},
"Sao_Paulo": {
  "exemplarCity": "ساو باولو"
},
"Scoresbysund": {
  "exemplarCity": "سكورسبيسند"
},
"Sitka": {
  "exemplarCity": "سیتکا"
},
"St_Barthelemy": {
  "exemplarCity": "سانت بارتيليمي"
},
"St_Johns": {
  "exemplarCity": "سانت جونس"
},
"St_Kitts": {
  "exemplarCity": "سانت کیتس"
},
"St_Lucia": {
  "exemplarCity": "سانت لوشيا"
},
"St_Thomas": {
  "exemplarCity": "سانت توماس"
},
"St_Vincent": {
  "exemplarCity": "سانت فنسنت"
},
"Swift_Current": {
  "exemplarCity": "سوفت کارنت"
},
"Tegucigalpa": {
  "exemplarCity": "تيغوسيغالبا"
},
"Thule": {
  "exemplarCity": "ثيل"
},
"Thunder_Bay": {
  "exemplarCity": "ثندر باي"
},
"Tijuana": {
  "exemplarCity": "تيخوانا"
},
"Toronto": {
  "exemplarCity": "تورونتو"
},
"Tortola": {
  "exemplarCity": "تورتولا"
},
"Vancouver": {
  "exemplarCity": "فانكوفر"
},
},
```

```

    "Whitehorse": {
      "exemplarCity": "وايت هورس"
    },
    "Winnipeg": {
      "exemplarCity": "وينيبيج"
    },
    "Yakutat": {
      "exemplarCity": "ياكوتات"
    },
    "Yellowknife": {
      "exemplarCity": "يلونيف"
    }
  },
  "Atlantic": {
    "Azores": {
      "exemplarCity": "أزورس"
    },
    "Bermuda": {
      "exemplarCity": "برمودا"
    },
    "Canary": {
      "exemplarCity": "كناري"
    },
    "Cape_Verde": {
      "exemplarCity": "الرأس الأخضر"
    },
    "Faeroe": {
      "exemplarCity": "فارو"
    },
    "Madeira": {
      "exemplarCity": "ماديرا"
    },
    "Reykjavik": {
      "exemplarCity": "ريكيافيك"
    },
    "South_Georgia": {
      "exemplarCity": "جورجيا الجنوبية"
    },
    "St_Helena": {
      "exemplarCity": "سانت هيلينا"
    },
    "Stanley": {
      "exemplarCity": "استانلي"
    }
  },
  "Europe": {
    "Amsterdam": {
      "exemplarCity": "أمستردام"
    },
    "Andorra": {
      "exemplarCity": "أندورا"
    },
    "Astrakhan": {
      "exemplarCity": "أستراخان"
    },
    "Athens": {
      "exemplarCity": "أثينا"
    }
  }
}

```

```
    },  
    "Belgrade": {  
      "exemplarCity": "بلغراد"  
    },  
    "Berlin": {  
      "exemplarCity": "برلين"  
    },  
    "Bratislava": {  
      "exemplarCity": "براتيسلافا"  
    },  
    "Brussels": {  
      "exemplarCity": "بروكسل"  
    },  
    "Bucharest": {  
      "exemplarCity": "بوخارست"  
    },  
    "Budapest": {  
      "exemplarCity": "بودابست"  
    },  
    "Busingen": {  
      "exemplarCity": "بوسنغن"  
    },  
    "Chisinau": {  
      "exemplarCity": "تشيسيناو"  
    },  
    "Copenhagen": {  
      "exemplarCity": "كوبنهاغن"  
    },  
    "Dublin": {  
      "long": {  
        "daylight": "توقيت أيرلندا الرسمي"  
      },  
      "exemplarCity": "دبلن"  
    },  
    "Gibraltar": {  
      "exemplarCity": "جبل طارق"  
    },  
    "Guernsey": {  
      "exemplarCity": "غيرنسي"  
    },  
    "Helsinki": {  
      "exemplarCity": "هلسنكي"  
    },  
    "Isle_of_Man": {  
      "exemplarCity": "جزيرة مان"  
    },  
    "Istanbul": {  
      "exemplarCity": "إسطنبول"  
    },  
    "Jersey": {  
      "exemplarCity": "جيرسي"  
    },  
    "Kaliningrad": {  
      "exemplarCity": "كالينجراد"  
    },  
    "Kiev": {  
      "exemplarCity": "كييف"
```

```
    },
    "Kirov": {
      "exemplarCity": "كيروف"
    },
    "Lisbon": {
      "exemplarCity": "لشبونة"
    },
    "Ljubljana": {
      "exemplarCity": "ليوبليانا"
    },
    "London": {
      "long": {
        "daylight": "توقيت بريطانيا الصيفي"
      },
      "exemplarCity": "لندن"
    },
    "Luxembourg": {
      "exemplarCity": "لوكسمبورغ"
    },
    "Madrid": {
      "exemplarCity": "مدريد"
    },
    "Malta": {
      "exemplarCity": "مالطة"
    },
    "Mariehamn": {
      "exemplarCity": "ماريهامن"
    },
    "Minsk": {
      "exemplarCity": "مينسك"
    },
    "Monaco": {
      "exemplarCity": "موناكو"
    },
    "Moscow": {
      "exemplarCity": "موسكو"
    },
    "Oslo": {
      "exemplarCity": "أوسلو"
    },
    "Paris": {
      "exemplarCity": "باريس"
    },
    "Podgorica": {
      "exemplarCity": "بودغوريكا"
    },
    "Prague": {
      "exemplarCity": "براغ"
    },
    "Riga": {
      "exemplarCity": "ريغا"
    },
    "Rome": {
      "exemplarCity": "روما"
    },
    "Samara": {
      "exemplarCity": "سمراء"
```

```
},
"San_Marino": {
  "exemplarCity": "سان مارينو"
},
"Sarajevo": {
  "exemplarCity": "سراييفو"
},
"Saratov": {
  "exemplarCity": "ساراتوف"
},
"Simferopol": {
  "exemplarCity": "سيمفروبول"
},
"Skopje": {
  "exemplarCity": "سكوبي"
},
"Sofia": {
  "exemplarCity": "صوفيا"
},
"Stockholm": {
  "exemplarCity": "ستوكهولم"
},
"Tallinn": {
  "exemplarCity": "تالين"
},
"Tirane": {
  "exemplarCity": "تيرانا"
},
"Ulyanovsk": {
  "exemplarCity": "أوليانوفسك"
},
"Uzhgorod": {
  "exemplarCity": "أوزجروود"
},
"Vaduz": {
  "exemplarCity": "فادوز"
},
"Vatican": {
  "exemplarCity": "الفاتيكان"
},
"Vienna": {
  "exemplarCity": "فيينا"
},
"Vilnius": {
  "exemplarCity": "فيلنيوس"
},
"Volgograd": {
  "exemplarCity": "فولجograd"
},
"Warsaw": {
  "exemplarCity": "وارسو"
},
"Zagreb": {
  "exemplarCity": "زغرب"
},
"Zaporozhye": {
  "exemplarCity": "زابوروزي"
```



```
    },  
    "Zurich": {  
      "exemplarCity": "زيورخ"  
    }  
  },  
  "Africa": {  
    "Abidjan": {  
      "exemplarCity": "أبيدجان"  
    },  
    "Accra": {  
      "exemplarCity": "أكرا"  
    },  
    "Addis_Ababa": {  
      "exemplarCity": "أديس أبابا"  
    },  
    "Algiers": {  
      "exemplarCity": "الجزائر"  
    },  
    "Asmera": {  
      "exemplarCity": "أسمره"  
    },  
    "Bamako": {  
      "exemplarCity": "باماكو"  
    },  
    "Bangui": {  
      "exemplarCity": "بانغوي"  
    },  
    "Banjul": {  
      "exemplarCity": "بانجول"  
    },  
    "Bissau": {  
      "exemplarCity": "بيساو"  
    },  
    "Blantyre": {  
      "exemplarCity": "بلانتيير"  
    },  
    "Brazzaville": {  
      "exemplarCity": "برازافيل"  
    },  
    "Bujumbura": {  
      "exemplarCity": "بوجومبورا"  
    },  
    "Cairo": {  
      "exemplarCity": "القاهرة"  
    },  
    "Casablanca": {  
      "exemplarCity": "الدار البيضاء"  
    },  
    "Ceuta": {  
      "exemplarCity": "سيتا"  
    },  
    "Conakry": {  
      "exemplarCity": "كوناكري"  
    },  
    "Dakar": {  
      "exemplarCity": "داكار"  
    }  
  },  
}
```

```
"Dar_es_Salaam": {
  "exemplarCity": "دار السلام"
},
"Djibouti": {
  "exemplarCity": "جيبوتي"
},
"Douala": {
  "exemplarCity": "دوالا"
},
"El_Aaiun": {
  "exemplarCity": "العيون"
},
"Freetown": {
  "exemplarCity": "فري تاون"
},
"Gaborone": {
  "exemplarCity": "غابورون"
},
"Harare": {
  "exemplarCity": "هراري"
},
"Johannesburg": {
  "exemplarCity": "جوهانسبرغ"
},
"Juba": {
  "exemplarCity": "جوبا"
},
"Kampala": {
  "exemplarCity": "كامبالا"
},
"Khartoum": {
  "exemplarCity": "الخرطوم"
},
"Kigali": {
  "exemplarCity": "كيغالي"
},
"Kinshasa": {
  "exemplarCity": "كينشاسا"
},
"Lagos": {
  "exemplarCity": "لاغوس"
},
"Libreville": {
  "exemplarCity": "ليبرفيل"
},
"Lome": {
  "exemplarCity": "لومي"
},
"Luanda": {
  "exemplarCity": "لواندا"
},
"Lubumbashi": {
  "exemplarCity": "لومبباشا"
},
"Lusaka": {
  "exemplarCity": "لوساكا"
},
}
```

```

    "Malabo": {
      "exemplarCity": "مالابو"
    },
    "Maputo": {
      "exemplarCity": "مابوتو"
    },
    "Maseru": {
      "exemplarCity": "ماسيرو"
    },
    "Mbabane": {
      "exemplarCity": "مباباني"
    },
    "Mogadishu": {
      "exemplarCity": "مقديشيو"
    },
    "Monrovia": {
      "exemplarCity": "مونروفيا"
    },
    "Nairobi": {
      "exemplarCity": "نيروبي"
    },
    "Ndjamena": {
      "exemplarCity": "نجامينا"
    },
    "Niamey": {
      "exemplarCity": "نيامي"
    },
    "Nouakchott": {
      "exemplarCity": "نواكشوط"
    },
    "Ouagadougou": {
      "exemplarCity": "واغادوغو"
    },
    "Porto-Novo": {
      "exemplarCity": "بورتو نوفو"
    },
    "Sao_Tome": {
      "exemplarCity": "ساو تومي"
    },
    "Tripoli": {
      "exemplarCity": "طرابلس"
    },
    "Tunis": {
      "exemplarCity": "تونس"
    },
    "Windhoek": {
      "exemplarCity": "ويندهوك"
    }
  },
  "Asia": {
    "Aden": {
      "exemplarCity": "عدن"
    },
    "Almaty": {
      "exemplarCity": "ألماتي"
    },
    "Amman": {

```

```
    "exemplarCity": "عمان"
  },
  "Anadyr": {
    "exemplarCity": "أندير"
  },
  "Aqttau": {
    "exemplarCity": "أكتاو"
  },
  "Aqtobe": {
    "exemplarCity": "أكتوب"
  },
  "Ashgabat": {
    "exemplarCity": "عشق آباد"
  },
  "Atyrau": {
    "exemplarCity": "أتيراو"
  },
  "Baghdad": {
    "exemplarCity": "بغداد"
  },
  "Bahrain": {
    "exemplarCity": "البحرين"
  },
  "Baku": {
    "exemplarCity": "باكو"
  },
  "Bangkok": {
    "exemplarCity": "بانكوك"
  },
  "Barnaul": {
    "exemplarCity": "بارناول"
  },
  "Beirut": {
    "exemplarCity": "بيروت"
  },
  "Bishkek": {
    "exemplarCity": "بشكيك"
  },
  "Brunei": {
    "exemplarCity": "بروناي"
  },
  "Calcutta": {
    "exemplarCity": "كالكتا"
  },
  "Chita": {
    "exemplarCity": "تشيتا"
  },
  "Choibalsan": {
    "exemplarCity": "تشوبالسان"
  },
  "Colombo": {
    "exemplarCity": "كولومبو"
  },
  "Damascus": {
    "exemplarCity": "دمشق"
  },
  "Dhaka": {
```

```
    "exemplarCity": "دكا"  
  },  
  "Dili": {  
    "exemplarCity": "دیلی"  
  },  
  "Dubai": {  
    "exemplarCity": "دبی"  
  },  
  "Dushanbe": {  
    "exemplarCity": "دوشانبی"  
  },  
  "Famagusta": {  
    "exemplarCity": "فاماغوستا"  
  },  
  "Gaza": {  
    "exemplarCity": "غزة"  
  },  
  "Hebron": {  
    "exemplarCity": "هیبرون (مدینة الخلیل)"  
  },  
  "Hong_Kong": {  
    "exemplarCity": "هونغ كونغ"  
  },  
  "Hovd": {  
    "exemplarCity": "هوفد"  
  },  
  "Irkutsk": {  
    "exemplarCity": "ایرکیتسک"  
  },  
  "Jakarta": {  
    "exemplarCity": "جاکرتا"  
  },  
  "Jayapura": {  
    "exemplarCity": "جایابورا"  
  },  
  "Jerusalem": {  
    "exemplarCity": "القدس"  
  },  
  "Kabul": {  
    "exemplarCity": "کابل"  
  },  
  "Kamchatka": {  
    "exemplarCity": "کامشاتکا"  
  },  
  "Karachi": {  
    "exemplarCity": "کراتھی"  
  },  
  "Katmandu": {  
    "exemplarCity": "کاتماندو"  
  },  
  "Khandyga": {  
    "exemplarCity": "خاندیجا"  
  },  
  "Krasnoyarsk": {  
    "exemplarCity": "کراسنویارسک"  
  },  
  "Kuala_Lumpur": {
```

```
    "exemplarCity": "كوالا لامبور"
  },
  "Kuching": {
    "exemplarCity": "كيشينج"
  },
  "Kuwait": {
    "exemplarCity": "الكويت"
  },
  "Macau": {
    "exemplarCity": "ماكاو"
  },
  "Magadan": {
    "exemplarCity": "مجادن"
  },
  "Makassar": {
    "exemplarCity": "ماكسار"
  },
  "Manila": {
    "exemplarCity": "مانيلا"
  },
  "Muscat": {
    "exemplarCity": "مسقط"
  },
  "Nicosia": {
    "exemplarCity": "نيقوسيا"
  },
  "Novokuznetsk": {
    "exemplarCity": "نوفوكوزنتسك"
  },
  "Novosibirsk": {
    "exemplarCity": "نوفوسبيرسك"
  },
  "Omsk": {
    "exemplarCity": "أومسك"
  },
  "Oral": {
    "exemplarCity": "أورال"
  },
  "Phnom_Penh": {
    "exemplarCity": "بنوم بنه"
  },
  "Pontianak": {
    "exemplarCity": "بونتيانك"
  },
  "Pyongyang": {
    "exemplarCity": "بيونغ يانغ"
  },
  "Qatar": {
    "exemplarCity": "قطر"
  },
  "Qyzylorda": {
    "exemplarCity": "كيزيلوردا"
  },
  "Rangoon": {
    "exemplarCity": "رانغون"
  },
  "Riyadh": {
```

```
    "exemplarCity": "الرياض"
  },
  "Saigon": {
    "exemplarCity": "مدينة هو تشي منه"
  },
  "Sakhalin": {
    "exemplarCity": "سكالين"
  },
  "Samarkand": {
    "exemplarCity": "سمرقند"
  },
  "Seoul": {
    "exemplarCity": "سول"
  },
  "Shanghai": {
    "exemplarCity": "شنغهاي"
  },
  "Singapore": {
    "exemplarCity": "سنغافورة"
  },
  "Srednekolymsk": {
    "exemplarCity": "سريدنكوليمسك"
  },
  "Taipei": {
    "exemplarCity": "تايبيه"
  },
  "Tashkent": {
    "exemplarCity": "تشقند"
  },
  "Tbilisi": {
    "exemplarCity": "تبليسي"
  },
  "Tehran": {
    "exemplarCity": "طهران"
  },
  "Thimphu": {
    "exemplarCity": "تيمفو"
  },
  "Tokyo": {
    "exemplarCity": "طوكيو"
  },
  "Tomsk": {
    "exemplarCity": "تومسك"
  },
  "Ulaanbaatar": {
    "exemplarCity": "آلانباتار"
  },
  "Urumqi": {
    "exemplarCity": "أرومكي"
  },
  "Ust-Nera": {
    "exemplarCity": "أوست نيرا"
  },
  "Vientiane": {
    "exemplarCity": "فيانتيان"
  },
  "Vladivostok": {
```

```

        "exemplarCity": "فلاديفوستك"
    },
    "Yakutsk": {
        "exemplarCity": "ياكتسك"
    },
    "Yekaterinburg": {
        "exemplarCity": "يكاترنبيرج"
    },
    "Yerevan": {
        "exemplarCity": "يريفان"
    }
},
"Indian": {
    "Antananarivo": {
        "exemplarCity": "أنتاناناريفو"
    },
    "Chagos": {
        "exemplarCity": "تشاغوس"
    },
    "Christmas": {
        "exemplarCity": "كريسماس"
    },
    "Cocos": {
        "exemplarCity": "كوكوس"
    },
    "Comoro": {
        "exemplarCity": "جزر القمر"
    },
    "Kerguelen": {
        "exemplarCity": "كيرغويلين"
    },
    "Mahe": {
        "exemplarCity": "ماهي"
    },
    "Maldives": {
        "exemplarCity": "المالديف"
    },
    "Mauritius": {
        "exemplarCity": "موريشيوس"
    },
    "Mayotte": {
        "exemplarCity": "مايوت"
    },
    "Reunion": {
        "exemplarCity": "ريونيون"
    }
},
"Australia": {
    "Adelaide": {
        "exemplarCity": "أديليد"
    },
    "Brisbane": {
        "exemplarCity": "برسبان"
    },
    "Broken_Hill": {
        "exemplarCity": "بروكن هيل"
    }
},

```



```
"Currie": {
  "exemplarCity": "كوري"
},
"Darwin": {
  "exemplarCity": "دارون"
},
"Eucla": {
  "exemplarCity": "أوكلا"
},
"Hobart": {
  "exemplarCity": "هوبارت"
},
"Lindeman": {
  "exemplarCity": "ليندمان"
},
"Lord_Howe": {
  "exemplarCity": "لورد هاو"
},
"Melbourne": {
  "exemplarCity": "ميلبورن"
},
"Perth": {
  "exemplarCity": "برثا"
},
"Sydney": {
  "exemplarCity": "سيدني"
}
},
"Pacific": {
  "Apia": {
    "exemplarCity": "أبيا"
  },
  "Auckland": {
    "exemplarCity": "أوكلاند"
  },
  "Bougainville": {
    "exemplarCity": "بوغانفيل"
  },
  "Chatham": {
    "exemplarCity": "تشاثام"
  },
  "Easter": {
    "exemplarCity": "استر"
  },
  "Efate": {
    "exemplarCity": "إيفات"
  },
  "Enderbury": {
    "exemplarCity": "اندربيرج"
  },
  "Fakaofu": {
    "exemplarCity": "فاكاوفو"
  },
  "Fiji": {
    "exemplarCity": "فيجي"
  },
  "Funafuti": {
```

```
    "exemplarCity": "فونافوتي"
  },
  "Galapagos": {
    "exemplarCity": "جلاباجوس"
  },
  "Gambier": {
    "exemplarCity": "جامبير"
  },
  "Guadalcanal": {
    "exemplarCity": "غواد الكانال"
  },
  "Guam": {
    "exemplarCity": "غوام"
  },
  "Honolulu": {
    "exemplarCity": "هونولولو"
  },
  "Johnston": {
    "exemplarCity": "جونستون"
  },
  "Kiritimati": {
    "exemplarCity": "كيريتي ماتي"
  },
  "Kosrae": {
    "exemplarCity": "كوسرا"
  },
  "Kwajalein": {
    "exemplarCity": "كواجالين"
  },
  "Majuro": {
    "exemplarCity": "ماجورو"
  },
  "Marquesas": {
    "exemplarCity": "ماركيساس"
  },
  "Midway": {
    "exemplarCity": "ميدواي"
  },
  "Nauru": {
    "exemplarCity": "ناورو"
  },
  "Niue": {
    "exemplarCity": "نيوي"
  },
  "Norfolk": {
    "exemplarCity": "نورفولك"
  },
  "Noumea": {
    "exemplarCity": "نوميا"
  },
  "Pago_Pago": {
    "exemplarCity": "باغو باغو"
  },
  "Palau": {
    "exemplarCity": "بالاو"
  },
  "Pitcairn": {
```

```

        "exemplarCity": "بيتكيرن"
    },
    "Ponape": {
        "exemplarCity": "باناب"
    },
    "Port_Moresby": {
        "exemplarCity": "بور مورسبي"
    },
    "Rarotonga": {
        "exemplarCity": "راروتونغا"
    },
    "Saipan": {
        "exemplarCity": "سايبان"
    },
    "Tahiti": {
        "exemplarCity": "تاهيتي"
    },
    "Tarawa": {
        "exemplarCity": "تاراوا"
    },
    "Tongatapu": {
        "exemplarCity": "تونغاتابو"
    },
    "Truk": {
        "exemplarCity": "ترك"
    },
    "Wake": {
        "exemplarCity": "واك"
    },
    "Wallis": {
        "exemplarCity": "واليس"
    }
},
"Arctic": {
    "Longyearbyen": {
        "exemplarCity": "لونجيربين"
    }
},
"Antarctica": {
    "Casey": {
        "exemplarCity": "كاساي"
    },
    "Davis": {
        "exemplarCity": "دافيز"
    },
    "DumontDUrville": {
        "exemplarCity": "دي مونت دو روفيل"
    },
    "Macquarie": {
        "exemplarCity": "ماكوارى"
    },
    "Mawson": {
        "exemplarCity": "ماوسون"
    },
    "McMurdo": {
        "exemplarCity": "ماك موردو"
    }
},

```

```

    "Palmer": {
      "exemplarCity": "بالمير"
    },
    "Rothera": {
      "exemplarCity": "روثيرا"
    },
    "Syowa": {
      "exemplarCity": "سايووا"
    },
    "Troll": {
      "exemplarCity": "ترول"
    },
    "Vostok": {
      "exemplarCity": "فوستوك"
    }
  },
  "Etc": {
    "UTC": {
      "long": {
        "standard": "التوقيت العالمي المنسق"
      },
      "short": {
        "standard": "UTC"
      }
    },
    "Unknown": {
      "exemplarCity": "مدينة غير معروفة"
    }
  },
  "metazone": {
    "Afghanistan": {
      "long": {
        "standard": "توقيت أفغانستان"
      }
    },
    "Africa_Central": {
      "long": {
        "standard": "توقيت وسط أفريقيا"
      }
    },
    "Africa_Eastern": {
      "long": {
        "standard": "توقيت شرق أفريقيا"
      }
    },
    "Africa_Southern": {
      "long": {
        "standard": "توقيت جنوب أفريقيا"
      }
    },
    "Africa_Western": {
      "long": {
        "generic": "توقيت غرب أفريقيا",
        "standard": "توقيت غرب أفريقيا الرسمي",
        "daylight": "توقيت غرب أفريقيا الصيفي"
      }
    }
  }
}

```

```

    },
    "Alaska": {
      "long": {
        "generic": "توقيت ألاسكا",
        "standard": "التوقيت الرسمي لألاسكا",
        "daylight": "توقيت ألاسكا الصيفي"
      }
    },
    "Amazon": {
      "long": {
        "generic": "توقيت الأمازون",
        "standard": "توقيت الأمازون الرسمي",
        "daylight": "توقيت الأمازون الصيفي"
      }
    },
    "America_Central": {
      "long": {
        "generic": "التوقيت المركزي لأمريكا الشمالية",
        "standard": "التوقيت الرسمي المركزي لأمريكا الشمالية",
        "daylight": "التوقيت الصيفي المركزي لأمريكا الشمالية"
      }
    },
    "America_Eastern": {
      "long": {
        "generic": "التوقيت الشرقي لأمريكا الشمالية",
        "standard": "التوقيت الرسمي الشرقي لأمريكا الشمالية",
        "daylight": "التوقيت الصيفي الشرقي لأمريكا الشمالية"
      }
    },
    "America_Mountain": {
      "long": {
        "generic": "التوقيت الجبلي لأمريكا الشمالية",
        "standard": "التوقيت الجبلي الرسمي لأمريكا الشمالية",
        "daylight": "التوقيت الجبلي الصيفي لأمريكا الشمالية"
      }
    },
    "America_Pacific": {
      "long": {
        "generic": "توقيت المحيط الهادي",
        "standard": "توقيت المحيط الهادي الرسمي",
        "daylight": "توقيت المحيط الهادي الصيفي"
      }
    },
    "Anadyr": {
      "long": {
        "generic": "توقيت أنادير",
        "standard": "توقيت أنادير الرسمي",
        "daylight": "التوقيت الصيفي لأنادير"
      }
    },
    "Apia": {
      "long": {
        "generic": "توقيت آبيا",
        "standard": "التوقيت الرسمي لآبيا",
        "daylight": "التوقيت الصيفي لآبيا"
      }
    },
  },

```

```

"Arabic": {
  "long": {
    "generic": "التوقيت العربي",
    "standard": "التوقيت العربي الرسمي",
    "daylight": "التوقيت العربي الصيفي"
  }
},
"Argentina": {
  "long": {
    "generic": "توقيت الأرجنتين",
    "standard": "توقيت الأرجنتين الرسمي",
    "daylight": "توقيت الأرجنتين الصيفي"
  }
},
"Argentina_Western": {
  "long": {
    "generic": "توقيت غرب الأرجنتين",
    "standard": "توقيت غرب الأرجنتين الرسمي",
    "daylight": "توقيت غرب الأرجنتين الصيفي"
  }
},
"Armenia": {
  "long": {
    "generic": "توقيت أرمينيا",
    "standard": "توقيت أرمينيا الرسمي",
    "daylight": "توقيت أرمينيا الصيفي"
  }
},
"Atlantic": {
  "long": {
    "generic": "توقيت الأطلسي",
    "standard": "التوقيت الرسمي الأطلسي",
    "daylight": "التوقيت الصيفي الأطلسي"
  }
},
"Australia_Central": {
  "long": {
    "generic": "توقيت وسط أستراليا",
    "standard": "توقيت وسط أستراليا الرسمي",
    "daylight": "توقيت وسط أستراليا الصيفي"
  }
},
"Australia_CentralWestern": {
  "long": {
    "generic": "توقيت غرب وسط أستراليا",
    "standard": "توقيت غرب وسط أستراليا الرسمي",
    "daylight": "توقيت غرب وسط أستراليا الصيفي"
  }
},
"Australia_Eastern": {
  "long": {
    "generic": "توقيت شرق أستراليا",
    "standard": "توقيت شرق أستراليا الرسمي",
    "daylight": "توقيت شرق أستراليا الصيفي"
  }
},
"Australia_Western": {

```

```

    "long": {
      "generic": "توقيت غرب أستراليا",
      "standard": "توقيت غرب أستراليا الرسمي",
      "daylight": "توقيت غرب أستراليا الصيفي"
    }
  },
  "Azerbaijan": {
    "long": {
      "generic": "توقيت أذربيجان",
      "standard": "توقيت أذربيجان الرسمي",
      "daylight": "توقيت أذربيجان الصيفي"
    }
  },
  "Azores": {
    "long": {
      "generic": "توقيت أزورس",
      "standard": "توقيت أزورس الرسمي",
      "daylight": "توقيت أزورس الصيفي"
    }
  },
  "Bangladesh": {
    "long": {
      "generic": "توقيت بنغلاديش",
      "standard": "توقيت بنغلاديش الرسمي",
      "daylight": "توقيت بنغلاديش الصيفي"
    }
  },
  "Bhutan": {
    "long": {
      "standard": "توقيت بوتان"
    }
  },
  "Bolivia": {
    "long": {
      "standard": "توقيت بوليفيا"
    }
  },
  "Brasilia": {
    "long": {
      "generic": "توقيت برازيليا",
      "standard": "توقيت برازيليا الرسمي",
      "daylight": "توقيت برازيليا الصيفي"
    }
  },
  "Brunei": {
    "long": {
      "standard": "توقيت بروناي"
    }
  },
  "Cape_Verde": {
    "long": {
      "generic": "توقيت الرأس الأخضر",
      "standard": "توقيت الرأس الأخضر الرسمي",
      "daylight": "توقيت الرأس الأخضر الصيفي"
    }
  },
  "Chamorro": {

```

```

        "long": {
            "standard": "توقيت تشامورو"
        }
    },
    "Chatham": {
        "long": {
            "generic": "توقيت تشاتام",
            "standard": "توقيت تشاتام الرسمي",
            "daylight": "توقيت تشاتام الصيفي"
        }
    },
    "Chile": {
        "long": {
            "generic": "توقيت شيلي",
            "standard": "توقيت شيلي الرسمي",
            "daylight": "توقيت شيلي الصيفي"
        }
    },
    "China": {
        "long": {
            "generic": "توقيت الصين",
            "standard": "توقيت الصين الرسمي",
            "daylight": "توقيت الصين الصيفي"
        }
    },
    "Choibalsan": {
        "long": {
            "generic": "توقيت شويبالسان",
            "standard": "توقيت شويبالسان الرسمي",
            "daylight": "التوقيت الصيفي لشويبالسان"
        }
    },
    "Christmas": {
        "long": {
            "standard": "توقيت جزر الكريسماس"
        }
    },
    "Cocos": {
        "long": {
            "standard": "توقيت جزر كوكوس"
        }
    },
    "Colombia": {
        "long": {
            "generic": "توقيت كولومبيا",
            "standard": "توقيت كولومبيا الرسمي",
            "daylight": "توقيت كولومبيا الصيفي"
        }
    },
    "Cook": {
        "long": {
            "generic": "توقيت جزر كوك",
            "standard": "توقيت جزر كوك الرسمي",
            "daylight": "توقيت جزر كوك الصيفي"
        }
    },
    "Cuba": {

```



```

    "long": {
      "generic": "توقيت كوبا",
      "standard": "توقيت كوبا الرسمي",
      "daylight": "توقيت كوبا الصيفي"
    }
  },
  "Davis": {
    "long": {
      "standard": "توقيت دافيز"
    }
  },
  "DumontDURville": {
    "long": {
      "standard": "توقيت دي مونت دو روفيل"
    }
  },
  "East_Timor": {
    "long": {
      "standard": "توقيت تيمور الشرقية"
    }
  },
  "Easter": {
    "long": {
      "generic": "توقيت جزيرة استر",
      "standard": "توقيت جزيرة استر الرسمي",
      "daylight": "توقيت جزيرة استر الصيفي"
    }
  },
  "Ecuador": {
    "long": {
      "standard": "توقيت الإكوادور"
    }
  },
  "Europe_Central": {
    "long": {
      "generic": "توقيت وسط أوروبا",
      "standard": "توقيت وسط أوروبا الرسمي",
      "daylight": "توقيت وسط أوروبا الصيفي"
    }
  },
  "Europe_Eastern": {
    "long": {
      "generic": "توقيت شرق أوروبا",
      "standard": "توقيت شرق أوروبا الرسمي",
      "daylight": "توقيت شرق أوروبا الصيفي"
    }
  },
  "Europe_Further_Eastern": {
    "long": {
      "standard": "التوقيت الأوروبي (أكثر شرقًا)"
    }
  },
  "Europe_Western": {
    "long": {
      "generic": "توقيت غرب أوروبا",
      "standard": "توقيت غرب أوروبا الرسمي",
      "daylight": "توقيت غرب أوروبا الصيفي"
    }
  }
}

```

```

    }
  },
  "Falkland": {
    "long": {
      "generic": "توقيت جزر فوكلاند",
      "standard": "توقيت جزر فوكلاند الرسمي",
      "daylight": "توقيت جزر فوكلاند الصيفي"
    }
  },
  "Fiji": {
    "long": {
      "generic": "توقيت فيجي",
      "standard": "توقيت فيجي الرسمي",
      "daylight": "توقيت فيجي الصيفي"
    }
  },
  "French_Guiana": {
    "long": {
      "standard": "توقيت غايانا الفرنسية"
    }
  },
  "French_Southern": {
    "long": {
      "standard": "توقيت المقاطعات الفرنسية الجنوبية والأنتارتيكية"
    }
  },
  "Galapagos": {
    "long": {
      "standard": "توقيت غلاباغوس"
    }
  },
  "Gambier": {
    "long": {
      "standard": "توقيت جامبير"
    }
  },
  "Georgia": {
    "long": {
      "generic": "توقيت جورجيا",
      "standard": "توقيت جورجيا الرسمي",
      "daylight": "توقيت جورجيا الصيفي"
    }
  },
  "Gilbert_Islands": {
    "long": {
      "standard": "توقيت جزر جيلبرت"
    }
  },
  "GMT": {
    "long": {
      "standard": "توقيت غرينتش"
    }
  },
  "Greenland_Eastern": {
    "long": {
      "generic": "توقيت شرق غرينلاند",
      "standard": "توقيت شرق غرينلاند الرسمي",

```

```

        "daylight": "توقيت شرق غرينلاند الصيفي"
    },
    },
    "Greenland_Western": {
        "long": {
            "generic": "توقيت غرب غرينلاند",
            "standard": "توقيت غرب غرينلاند الرسمي",
            "daylight": "توقيت غرب غرينلاند الصيفي"
        }
    },
    },
    "Guam": {
        "long": {
            "standard": "توقيت غوام"
        }
    },
    },
    "Gulf": {
        "long": {
            "standard": "توقيت الخليج"
        }
    },
    },
    "Guyana": {
        "long": {
            "standard": "توقيت غيانا"
        }
    },
    },
    "Hawaii_Aleutian": {
        "long": {
            "generic": "توقيت هاواي ألوتيان",
            "standard": "توقيت هاواي ألوتيان الرسمي",
            "daylight": "توقيت هاواي ألوتيان الصيفي"
        }
    },
    },
    "Hong_Kong": {
        "long": {
            "generic": "توقيت هونغ كونغ",
            "standard": "توقيت هونغ كونغ الرسمي",
            "daylight": "توقيت هونغ كونغ الصيفي"
        }
    },
    },
    "Hovd": {
        "long": {
            "generic": "توقيت هوفد",
            "standard": "توقيت هوفد الرسمي",
            "daylight": "توقيت هوفد الصيفي"
        }
    },
    },
    "India": {
        "long": {
            "standard": "توقيت الهند"
        }
    },
    },
    "Indian_Ocean": {
        "long": {
            "standard": "توقيت المحيط الهندي"
        }
    },
    },
    "Indochina": {

```

```

        "long": {
            "standard": "توقيت الهند الصينية"
        }
    },
    "Indonesia_Central": {
        "long": {
            "standard": "توقيت وسط إندونيسيا"
        }
    },
    "Indonesia_Eastern": {
        "long": {
            "standard": "توقيت شرق إندونيسيا"
        }
    },
    "Indonesia_Western": {
        "long": {
            "standard": "توقيت غرب إندونيسيا"
        }
    },
    "Iran": {
        "long": {
            "generic": "توقيت إيران",
            "standard": "توقيت إيران الرسمي",
            "daylight": "توقيت إيران الصيفي"
        }
    },
    "Irkutsk": {
        "long": {
            "generic": "توقيت إركوتسك",
            "standard": "توقيت إركوتسك الرسمي",
            "daylight": "توقيت إركوتسك الصيفي"
        }
    },
    "Israel": {
        "long": {
            "generic": "توقيت إسرائيل",
            "standard": "توقيت إسرائيل الرسمي",
            "daylight": "توقيت إسرائيل الصيفي"
        }
    },
    "Japan": {
        "long": {
            "generic": "توقيت اليابان",
            "standard": "توقيت اليابان الرسمي",
            "daylight": "توقيت اليابان الصيفي"
        }
    },
    "Kamchatka": {
        "long": {
            "generic": "توقيت كامشاتكا",
            "standard": "توقيت بيتروبافلوفسك-كامتشاتسكي",
            "daylight": "توقيت بيتروبافلوفسك-كامتشاتسكي الصيفي"
        }
    },
    "Kazakhstan_Eastern": {
        "long": {
            "standard": "توقيت شرق كازاخستان"
        }
    }

```

```

    }
  },
  "Kazakhstan_Western": {
    "long": {
      "standard": "توقيت غرب كازاخستان"
    }
  },
  "Korea": {
    "long": {
      "generic": "توقيت كوريا",
      "standard": "توقيت كوريا الرسمي",
      "daylight": "توقيت كوريا الصيفي"
    }
  },
  "Kosrae": {
    "long": {
      "standard": "توقيت كوسرا"
    }
  },
  "Krasnoyarsk": {
    "long": {
      "generic": "توقيت كراسنويارسك",
      "standard": "توقيت كراسنويارسك الرسمي",
      "daylight": "التوقيت الصيفي لكراسنويارسك"
    }
  },
  "Kyrgystan": {
    "long": {
      "standard": "توقيت قيرغيزستان"
    }
  },
  "Line_Islands": {
    "long": {
      "standard": "توقيت جزر لاين"
    }
  },
  "Lord_Howe": {
    "long": {
      "generic": "توقيت لورد هاو",
      "standard": "توقيت لورد هاو الرسمي",
      "daylight": "التوقيت الصيفي للورد هاو"
    }
  },
  "Macquarie": {
    "long": {
      "standard": "توقيت ماكوارى"
    }
  },
  "Magadan": {
    "long": {
      "generic": "توقيت ماغادان",
      "standard": "توقيت ماغادان الرسمي",
      "daylight": "توقيت ماغادان الصيفي"
    }
  },
  "Malaysia": {
    "long": {

```

```

        "standard": "توقيت ماليزيا"
    },
    },
    "Maldives": {
        "long": {
            "standard": "توقيت المالديف"
        }
    },
    "Marquesas": {
        "long": {
            "standard": "توقيت ماركيساس"
        }
    },
    "Marshall_Islands": {
        "long": {
            "standard": "توقيت جزر مارشال"
        }
    },
    "Mauritius": {
        "long": {
            "generic": "توقيت موريشيوس",
            "standard": "توقيت موريشيوس الرسمي",
            "daylight": "توقيت موريشيوس الصيفي"
        }
    },
    "Mawson": {
        "long": {
            "standard": "توقيت ماوسون"
        }
    },
    "Mexico_Northwest": {
        "long": {
            "generic": "توقيت شمال غرب المكسيك",
            "standard": "التوقيت الرسمي لشمال غرب المكسيك",
            "daylight": "التوقيت الصيفي لشمال غرب المكسيك"
        }
    },
    "Mexico_Pacific": {
        "long": {
            "generic": "توقيت المحيط الهادي للمكسيك",
            "standard": "توقيت المحيط الهادي الرسمي للمكسيك",
            "daylight": "توقيت المحيط الهادي الصيفي للمكسيك"
        }
    },
    "Mongolia": {
        "long": {
            "generic": "توقيت أولان باتور",
            "standard": "توقيت أولان باتور الرسمي",
            "daylight": "توقيت أولان باتور الصيفي"
        }
    },
    "Moscow": {
        "long": {
            "generic": "توقيت موسكو",
            "standard": "توقيت موسكو الرسمي",
            "daylight": "توقيت موسكو الصيفي"
        }
    }
}

```

```

    },
    "Myanmar": {
      "long": {
        "standard": "توقيت میانمار"
      }
    },
    "Nauru": {
      "long": {
        "standard": "توقيت ناورو"
      }
    },
    "Nepal": {
      "long": {
        "standard": "توقيت نيبال"
      }
    },
    "New_Caledonia": {
      "long": {
        "generic": "توقيت كاليدونيا الجديدة",
        "standard": "توقيت كاليدونيا الجديدة الرسمي",
        "daylight": "توقيت كاليدونيا الجديدة الصيفي"
      }
    },
    "New_Zealand": {
      "long": {
        "generic": "توقيت نيوزيلندا",
        "standard": "توقيت نيوزيلندا الرسمي",
        "daylight": "توقيت نيوزيلندا الصيفي"
      }
    },
    "Newfoundland": {
      "long": {
        "generic": "توقيت نيوفاوندلاند",
        "standard": "توقيت نيوفاوندلاند الرسمي",
        "daylight": "توقيت نيوفاوندلاند الصيفي"
      }
    },
    "Niue": {
      "long": {
        "standard": "توقيت نيوي"
      }
    },
    "Norfolk": {
      "long": {
        "standard": "توقيت جزيرة نورفولك"
      }
    },
    "Noronha": {
      "long": {
        "generic": "توقيت فيرناندو دي نورونها",
        "standard": "توقيت فرناندو دي نورونها الرسمي",
        "daylight": "توقيت فرناندو دي نورونها الصيفي"
      }
    },
    "North_Mariana": {
      "long": {
        "standard": "توقيت جزر ماريانا الشمالية"
      }
    }
  }

```

```

    }
  },
  "Novosibirsk": {
    "long": {
      "generic": "توقيت نوفوسيبيرسك",
      "standard": "توقيت نوفوسيبيرسك الرسمي",
      "daylight": "توقيت نوفوسيبيرسك الصيفي"
    }
  },
  "Omsk": {
    "long": {
      "generic": "توقيت أومسك",
      "standard": "توقيت أومسك الرسمي",
      "daylight": "توقيت أومسك الصيفي"
    }
  },
  "Pakistan": {
    "long": {
      "generic": "توقيت باكستان",
      "standard": "توقيت باكستان الرسمي",
      "daylight": "توقيت باكستان الصيفي"
    }
  },
  "Palau": {
    "long": {
      "standard": "توقيت بالاو"
    }
  },
  "Papua_New_Guinea": {
    "long": {
      "standard": "توقيت بابوا غينيا الجديدة"
    }
  },
  "Paraguay": {
    "long": {
      "generic": "توقيت باراغواي",
      "standard": "توقيت باراغواي الرسمي",
      "daylight": "توقيت باراغواي الصيفي"
    }
  },
  "Peru": {
    "long": {
      "generic": "توقيت بيرو",
      "standard": "توقيت بيرو الرسمي",
      "daylight": "توقيت بيرو الصيفي"
    }
  },
  "Philippines": {
    "long": {
      "generic": "توقيت الفلبين",
      "standard": "توقيت الفلبين الرسمي",
      "daylight": "توقيت الفلبين الصيفي"
    }
  },
  "Phoenix_Islands": {
    "long": {
      "standard": "توقيت جزر فينكس"
    }
  }
}

```



```

    }
  },
  "Pierre_Miquelon": {
    "long": {
      "generic": "توقيت سانت بيير وميكلون",
      "standard": "توقيت سانت بيير وميكلون الرسمي",
      "daylight": "توقيت سانت بيير وميكلون الصيفي"
    }
  },
  "Pitcairn": {
    "long": {
      "standard": "توقيت بيتكيرن"
    }
  },
  "Ponape": {
    "long": {
      "standard": "توقيت بونابي"
    }
  },
  "Pyongyang": {
    "long": {
      "standard": "توقيت بيونغ يانغ"
    }
  },
  "Reunion": {
    "long": {
      "standard": "توقيت ريونيون"
    }
  },
  "Rothera": {
    "long": {
      "standard": "توقيت روثيرا"
    }
  },
  "Sakhalin": {
    "long": {
      "generic": "توقيت ساخالين",
      "standard": "توقيت ساخالين الرسمي",
      "daylight": "توقيت ساخالين الصيفي"
    }
  },
  "Samara": {
    "long": {
      "generic": "توقيت سامارا",
      "standard": "توقيت سامارا",
      "daylight": "توقيت سامارا الصيفي"
    }
  },
  "Samoa": {
    "long": {
      "generic": "توقيت ساموا",
      "standard": "توقيت ساموا الرسمي",
      "daylight": "توقيت ساموا الصيفي"
    }
  },
  "Seychelles": {
    "long": {

```

```

        "standard": "توقيت سيشل"
    },
    },
    "Singapore": {
        "long": {
            "standard": "توقيت سنغافورة"
        }
    },
    },
    "Solomon": {
        "long": {
            "standard": "توقيت جزر سليمان"
        }
    },
    },
    "South_Georgia": {
        "long": {
            "standard": "توقيت جنوب جورجيا"
        }
    },
    },
    "Suriname": {
        "long": {
            "standard": "توقيت سورينام"
        }
    },
    },
    "Syowa": {
        "long": {
            "standard": "توقيت سايووا"
        }
    },
    },
    "Tahiti": {
        "long": {
            "standard": "توقيت تاهيتي"
        }
    },
    },
    "Taipei": {
        "long": {
            "generic": "توقيت تايبه",
            "standard": "توقيت تايبه الرسمي",
            "daylight": "توقيت تايبه الصيفي"
        }
    },
    },
    "Tajikistan": {
        "long": {
            "standard": "توقيت طاجكستان"
        }
    },
    },
    "Tokelau": {
        "long": {
            "standard": "توقيت توكيلاو"
        }
    },
    },
    "Tonga": {
        "long": {
            "generic": "توقيت تونغا",
            "standard": "توقيت تونغا الرسمي",
            "daylight": "توقيت تونغا الصيفي"
        }
    },
    },
    },

```

```

"Truk": {
  "long": {
    "standard": "توقيت شوك"
  }
},
"Turkmenistan": {
  "long": {
    "generic": "توقيت تركمانستان",
    "standard": "توقيت تركمانستان الرسمي",
    "daylight": "توقيت تركمانستان الصيفي"
  }
},
"Tuvalu": {
  "long": {
    "standard": "توقيت توفالو"
  }
},
"Uruguay": {
  "long": {
    "generic": "توقيت أوروغواي",
    "standard": "توقيت أوروغواي الرسمي",
    "daylight": "توقيت أوروغواي الصيفي"
  }
},
"Uzbekistan": {
  "long": {
    "generic": "توقيت أوزبكستان",
    "standard": "توقيت أوزبكستان الرسمي",
    "daylight": "توقيت أوزبكستان الصيفي"
  }
},
"Vanuatu": {
  "long": {
    "generic": "توقيت فانواتو",
    "standard": "توقيت فانواتو الرسمي",
    "daylight": "توقيت فانواتو الصيفي"
  }
},
"Venezuela": {
  "long": {
    "standard": "توقيت فنزويلا"
  }
},
"Vladivostok": {
  "long": {
    "generic": "توقيت فلاديفوستوك",
    "standard": "توقيت فلاديفوستوك الرسمي",
    "daylight": "توقيت فلاديفوستوك الصيفي"
  }
},
"Volgograd": {
  "long": {
    "generic": "توقيت فولغوغراد",
    "standard": "توقيت فولغوغراد الرسمي",
    "daylight": "توقيت فولغوغراد الصيفي"
  }
},

```

```

    "Vostok": {
      "long": {
        "standard": "توقيت فوستوك"
      }
    },
    "Wake": {
      "long": {
        "standard": "توقيت جزيرة ويك"
      }
    },
    "Wallis": {
      "long": {
        "standard": "توقيت واليس و فوتونا"
      }
    },
    "Yakutsk": {
      "long": {
        "generic": "توقيت ياكوتسك",
        "standard": "توقيت ياكوتسك الرسمي",
        "daylight": "توقيت ياكوتسك الصيفي"
      }
    },
    "Yekaterinburg": {
      "long": {
        "generic": "توقيت يكاترينبورغ",
        "standard": "توقيت يكاترينبورغ الرسمي",
        "daylight": "توقيت يكاترينبورغ الصيفي"
      }
    }
  }
}

```

TIMEZONENAMES.JSX

```

{
  "main";
  {
    "ar";
    {
      "identity";
      {
        "version";
        {
          "_number";
          "$Revision: 13686 $",
          "_cldrVersion";
          "32";
        }
        "language";
        "ar";
      }
    }
    "dates";
  }
}

```

```

    {
      "timeZoneNames";
      {
        "hourFormat";
        "+HH:mm;-HH:mm",
        "gmtFormat";
        "0{غرينتش",
        "gmtZeroFormat";
        "غرينتش",
        "regionFormat";
        "0{توقيت",
        "regionFormat-type-daylight";
        "الصيفي} 0{توقيت",
        "regionFormat-type-standard";
        "الرسمي} 0{توقيت",
        "fallbackFormat";
        "{1} ({0})",
        "zone";
        {
          "America";
          {
            "Adak";
            {
              "exemplarCity";
              "أداك";
            }
            "Anchorage";
            {
              "exemplarCity";
              "أنشوراج";
            }
            "Anguilla";
            {
              "exemplarCity";
              "أنغويلا";
            }
            "Antigua";
            {
              "exemplarCity";
              "أنتيغوا";
            }
            "Araguaina";
            {
              "exemplarCity";
              "أروجوانيا";
            }
            "Argentina";
            {
              "Rio_Gallegos";
              {
                "exemplarCity";
                "ريو جاليوس";
              }
              "San_Juan";
              {
                "exemplarCity";
                "سان خوان";
              }
            }
          }
        }
      }
    }
  
```

```
}
"Ushuaia";
{
  "exemplarCity";
  "أشوا";
}
"La_Rioja";
{
  "exemplarCity";
  "لا ريوجا";
}
"San_Luis";
{
  "exemplarCity";
  "سان لويس";
}
"Salta";
{
  "exemplarCity";
  "سالطا";
}
"Tucuman";
{
  "exemplarCity";
  "تاكمان";
}
}
"Aruba";
{
  "exemplarCity";
  "أروبا";
}
"Asuncion";
{
  "exemplarCity";
  "أسونسيون";
}
"Bahia";
{
  "exemplarCity";
  "باهيا";
}
"Bahia_Banderas";
{
  "exemplarCity";
  "باهيا بانديراس";
}
"Barbados";
{
  "exemplarCity";
  "بربادوس";
}
"Belem";
{
  "exemplarCity";
  "بلم";
}
```

```
"Belize";
{
  "exemplarCity";
  "بليز";
}
"Blanc-Sablon";
{
  "exemplarCity";
  "بلانك-سابلون";
}
"Boa_Vista";
{
  "exemplarCity";
  "باو فيستا";
}
"Bogota";
{
  "exemplarCity";
  "بوغوتا";
}
"Boise";
{
  "exemplarCity";
  "بويس";
}
"Buenos_Aires";
{
  "exemplarCity";
  "بوينوس آيرس";
}
"Cambridge_Bay";
{
  "exemplarCity";
  "كامبرديج باي";
}
"Campo_Grande";
{
  "exemplarCity";
  "كومبو جراند";
}
"Cancun";
{
  "exemplarCity";
  "كانكون";
}
"Caracas";
{
  "exemplarCity";
  "كاراكاس";
}
"Catamarca";
{
  "exemplarCity";
  "كاتاماركا";
}
"Cayenne";
{
```

```
        "exemplarCity";
        "كايين";
    }
    "Cayman";
    {
        "exemplarCity";
        "كايمان";
    }
    "Chicago";
    {
        "exemplarCity";
        "شيكاغو";
    }
    "Chihuahua";
    {
        "exemplarCity";
        "تشياواوا";
    }
    "Coral_Harbour";
    {
        "exemplarCity";
        "كورانال هاربر";
    }
    "Cordoba";
    {
        "exemplarCity";
        "كوردوبا";
    }
    "Costa_Rica";
    {
        "exemplarCity";
        "كوستاريكا";
    }
    "Creston";
    {
        "exemplarCity";
        "كريستون";
    }
    "Cuiaba";
    {
        "exemplarCity";
        "كيابا";
    }
    "Curacao";
    {
        "exemplarCity";
        "كوراساو";
    }
    "Danmarkshavn";
    {
        "exemplarCity";
        "دانمرك شافن";
    }
    "Dawson";
    {
        "exemplarCity";
        "داوسان";
    }
```



```
}
"Dawson_Creek";
{
  "exemplarCity";
  "داوسن كريك";
}
"Denver";
{
  "exemplarCity";
  "دنفر";
}
"Detroit";
{
  "exemplarCity";
  "ديترويت";
}
"Dominica";
{
  "exemplarCity";
  "دومينيكا";
}
"Edmonton";
{
  "exemplarCity";
  "ايدمونتون";
}
"Eirunepe";
{
  "exemplarCity";
  "ايرونبي";
}
"El_Salvador";
{
  "exemplarCity";
  "السلفادور";
}
"Fort_Nelson";
{
  "exemplarCity";
  "فورت نيلسون";
}
"Fortaleza";
{
  "exemplarCity";
  "فورتاليزا";
}
"Glace_Bay";
{
  "exemplarCity";
  "جلاس باي";
}
"Godthab";
{
  "exemplarCity";
  "غودثاب";
}
"Goose_Bay";
```

```
{
  "exemplarCity";
  "جوس باي";
}
"Grand_Turk";
{
  "exemplarCity";
  "گرانند ترك";
}
"Grenada";
{
  "exemplarCity";
  "غرينادا";
}
"Guadeloupe";
{
  "exemplarCity";
  "غوادلوب";
}
"Guatemala";
{
  "exemplarCity";
  "غواتيمالا";
}
"Guayaquil";
{
  "exemplarCity";
  "غواياكويل";
}
"Guyana";
{
  "exemplarCity";
  "غيانا";
}
"Halifax";
{
  "exemplarCity";
  "هاليفاكس";
}
"Havana";
{
  "exemplarCity";
  "هافانا";
}
"Hermosillo";
{
  "exemplarCity";
  "هيرموسيلو";
}
"Indiana";
{
  "Vincennes";
  {
    "exemplarCity";
    "فينسينس";
  }
}
"Petersburg";
```

```
{
    "exemplarCity";
    "بيترسبرغ";
}
"Tell_City";
{
    "exemplarCity";
    "مدينة تل، إنديانا";
}
"Knox";
{
    "exemplarCity";
    "كونكس";
}
"Winamac";
{
    "exemplarCity";
    "ويناماك";
}
"Marengo";
{
    "exemplarCity";
    "مارنجو";
}
"Vevay";
{
    "exemplarCity";
    "فيفاي";
}
}
"Indianapolis";
{
    "exemplarCity";
    "إنديانابوليس";
}
"Inuvik";
{
    "exemplarCity";
    "اينوفيك";
}
"Iqaluit";
{
    "exemplarCity";
    "اكويلت";
}
"Jamaica";
{
    "exemplarCity";
    "جامايكا";
}
"Jujuy";
{
    "exemplarCity";
    "جوجو";
}
"Juneau";
{
```

```
        "exemplarCity";
        "جوني";
    }
    "Kentucky";
    {
        "Monticello";
        {
            "exemplarCity";
            "مونتي سيلو";
        }
    }
    "Kralendijk";
    {
        "exemplarCity";
        "كرالنديك";
    }
    "La_Paz";
    {
        "exemplarCity";
        "لا باز";
    }
    "Lima";
    {
        "exemplarCity";
        "ليما";
    }
    "Los_Angeles";
    {
        "exemplarCity";
        "لوس انجلوس";
    }
    "Louisville";
    {
        "exemplarCity";
        "لويس فيل";
    }
    "Lower_Princes";
    {
        "exemplarCity";
        "حي الأمير السفلي";
    }
    "Maceio";
    {
        "exemplarCity";
        "ماشيو";
    }
    "Managua";
    {
        "exemplarCity";
        "ماناغوا";
    }
    "Manaus";
    {
        "exemplarCity";
        "ماناوس";
    }
    "Marigot";
```

```
{
  "exemplarCity";
  "ماريغوت";
}
"Martinique";
{
  "exemplarCity";
  "المارتينيك";
}
"Matamoros";
{
  "exemplarCity";
  "ماتاموروس";
}
"Mazatlan";
{
  "exemplarCity";
  "مازااتلان";
}
"Mendoza";
{
  "exemplarCity";
  "ميندوزا";
}
"Menominee";
{
  "exemplarCity";
  "مينوميني";
}
"Merida";
{
  "exemplarCity";
  "ميريديا";
}
"Metlakatla";
{
  "exemplarCity";
  "ميتلاكاتلا";
}
"Mexico_City";
{
  "exemplarCity";
  "مدينة المكسيك";
}
"Miquelon";
{
  "exemplarCity";
  "مكويلون";
}
"Moncton";
{
  "exemplarCity";
  "وينكتون";
}
"Monterrey";
{
  "exemplarCity";
```

```

        "مونتييري";
    }
    "Montevideo";
    {
        "exemplarCity";
        "مونتيڤيديو";
    }
    "Montserrat";
    {
        "exemplarCity";
        "مونتسيرات";
    }
    "Nassau";
    {
        "exemplarCity";
        "ناسو";
    }
    "New_York";
    {
        "exemplarCity";
        "نيويورك";
    }
    "Nipigon";
    {
        "exemplarCity";
        "نيبيجون";
    }
    "Nome";
    {
        "exemplarCity";
        "نوم";
    }
    "Noronha";
    {
        "exemplarCity";
        "نوروناه";
    }
    "North_Dakota";
    {
        "Beulah";
        {
            "exemplarCity";
            "بيولا، داكوتا الشمالية";
        }
        "New_Salem";
        {
            "exemplarCity";
            "نيو ساليم";
        }
        "Center";
        {
            "exemplarCity";
            "سنتر";
        }
    }
    "Ojinaga";
    {

```

```
        "exemplarCity";
        "أوجينا جا";
    }
    "Panama";
    {
        "exemplarCity";
        "بنما";
    }
    "Pangnirtung";
    {
        "exemplarCity";
        "بانجيتينج";
    }
    "Paramaribo";
    {
        "exemplarCity";
        "باراماريبو";
    }
    "Phoenix";
    {
        "exemplarCity";
        "فينكس";
    }
    "Port-au-Prince";
    {
        "exemplarCity";
        "بورت أو برنس";
    }
    "Port_of_Spain";
    {
        "exemplarCity";
        "بورت أو ف سبين";
    }
    "Porto_Velho";
    {
        "exemplarCity";
        "بورتو فيلو";
    }
    "Puerto_Rico";
    {
        "exemplarCity";
        "بورتوريكو";
    }
    "Punta_Arenas";
    {
        "exemplarCity";
        "بونتأ أريناز";
    }
    "Rainy_River";
    {
        "exemplarCity";
        "راني ريفر";
    }
    "Rankin_Inlet";
    {
        "exemplarCity";
        "رانكن انلت";
    }
}
```

```
}
"Recife";
{
  "exemplarCity";
  "ريسيڤ";
}
"Regina";
{
  "exemplarCity";
  "ريجينا";
}
"Resolute";
{
  "exemplarCity";
  "ريزولوت";
}
"Rio_Branco";
{
  "exemplarCity";
  "ريوبرانكو";
}
"Santa_Isabel";
{
  "exemplarCity";
  "سانتا ايزابيل";
}
"Santarem";
{
  "exemplarCity";
  "سانتاريم";
}
"Santiago";
{
  "exemplarCity";
  "سانتياغو";
}
"Santo_Domingo";
{
  "exemplarCity";
  "سانتو دومينغو";
}
"Sao_Paulo";
{
  "exemplarCity";
  "ساو باولو";
}
"Scoresbysund";
{
  "exemplarCity";
  "سكورسبيسند";
}
"Sitka";
{
  "exemplarCity";
  "سيتكا";
}
"St_Barthelemy";
```



```
{
  "exemplarCity";
  "سانت بارتيليمي";
}
"St_Johns";
{
  "exemplarCity";
  "سانت جونز";
}
"St_Kitts";
{
  "exemplarCity";
  "سانت كيتس";
}
"St_Lucia";
{
  "exemplarCity";
  "سانت لوشيا";
}
"St_Thomas";
{
  "exemplarCity";
  "سانت توماس";
}
"St_Vincent";
{
  "exemplarCity";
  "سانت فنسنت";
}
"Swift_Current";
{
  "exemplarCity";
  "سوفت كارنت";
}
"Tegucigalpa";
{
  "exemplarCity";
  "تيغوسيغالبا";
}
"Thule";
{
  "exemplarCity";
  "ثيل";
}
"Thunder_Bay";
{
  "exemplarCity";
  "ثندر باي";
}
"Tijuana";
{
  "exemplarCity";
  "تيخوانا";
}
"Toronto";
{
  "exemplarCity";
```

```

        "تورونتو";
    }
    "Tortola";
    {
        "exemplarCity";
        "تورتولا";
    }
    "Vancouver";
    {
        "exemplarCity";
        "فانكوفر";
    }
    "Whitehorse";
    {
        "exemplarCity";
        "وايت هورس";
    }
    "Winnipeg";
    {
        "exemplarCity";
        "وينيبج";
    }
    "Yakutat";
    {
        "exemplarCity";
        "ياكوتات";
    }
    "Yellowknife";
    {
        "exemplarCity";
        "يلونيف";
    }
}
"Atlantic";
{
    "Azores";
    {
        "exemplarCity";
        "أزورس";
    }
    "Bermuda";
    {
        "exemplarCity";
        "برمودا";
    }
    "Canary";
    {
        "exemplarCity";
        "كناري";
    }
    "Cape_Verde";
    {
        "exemplarCity";
        "الرأس الأخضر";
    }
    "Faeroe";
    {

```

```
        "exemplarCity";
        "فارو";
    }
    "Madeira";
    {
        "exemplarCity";
        "ماديرا";
    }
    "Reykjavik";
    {
        "exemplarCity";
        "ريكيافيك";
    }
    "South_Georgia";
    {
        "exemplarCity";
        "جورجيا الجنوبية";
    }
    "St_Helena";
    {
        "exemplarCity";
        "سانت هيلينا";
    }
    "Stanley";
    {
        "exemplarCity";
        "استانلي";
    }
}
"Europe";
{
    "Amsterdam";
    {
        "exemplarCity";
        "أمستردام";
    }
    "Andorra";
    {
        "exemplarCity";
        "أندورا";
    }
    "Astrakhan";
    {
        "exemplarCity";
        "أستراخان";
    }
    "Athens";
    {
        "exemplarCity";
        "أثينا";
    }
    "Belgrade";
    {
        "exemplarCity";
        "بلغراد";
    }
    "Berlin";
```

```
{
  "exemplarCity";
  "برلين";
}
"Bratislava";
{
  "exemplarCity";
  "براتيسلافا";
}
"Brussels";
{
  "exemplarCity";
  "بروكسل";
}
"Bucharest";
{
  "exemplarCity";
  "بوخارست";
}
"Budapest";
{
  "exemplarCity";
  "بودابست";
}
"Busingen";
{
  "exemplarCity";
  "بوسنغن";
}
"Chisinau";
{
  "exemplarCity";
  "تشيسيناو";
}
"Copenhagen";
{
  "exemplarCity";
  "كوبنهاغن";
}
"Dublin";
{
  "long";
  {
    "daylight";
    "توقيت أيرلندا الرسمي";
  }
  "exemplarCity";
  "دبلن";
}
"Gibraltar";
{
  "exemplarCity";
  "جبل طارق";
}
"Guernsey";
{
  "exemplarCity";
```

```
        "غيرنسي";
    }
    "Helsinki";
    {
        "exemplarCity";
        "هلسنكي";
    }
    "Isle_of_Man";
    {
        "exemplarCity";
        "جزيرة مان";
    }
    "Istanbul";
    {
        "exemplarCity";
        "إسطنبول";
    }
    "Jersey";
    {
        "exemplarCity";
        "جيرسي";
    }
    "Kaliningrad";
    {
        "exemplarCity";
        "كالينجراد";
    }
    "Kiev";
    {
        "exemplarCity";
        "كييف";
    }
    "Kirov";
    {
        "exemplarCity";
        "كيروف";
    }
    "Lisbon";
    {
        "exemplarCity";
        "لشبونة";
    }
    "Ljubljana";
    {
        "exemplarCity";
        "ليوبليانا";
    }
    "London";
    {
        "long";
        {
            "daylight";
            "توقيت بريطانيا الصيفي";
        }
        "exemplarCity";
        "لندن";
    }
}
```

```
"Luxembourg";
{
  "exemplarCity";
  "لوكسمبورغ";
}
"Madrid";
{
  "exemplarCity";
  "مدريد";
}
"Malta";
{
  "exemplarCity";
  "مالطة";
}
"Mariehamn";
{
  "exemplarCity";
  "ماريهامن";
}
"Minsk";
{
  "exemplarCity";
  "مينسك";
}
"Monaco";
{
  "exemplarCity";
  "موناكو";
}
"Moscow";
{
  "exemplarCity";
  "موسكو";
}
"Oslo";
{
  "exemplarCity";
  "أوسلو";
}
"Paris";
{
  "exemplarCity";
  "باريس";
}
"Podgorica";
{
  "exemplarCity";
  "بودغوريكا";
}
"Prague";
{
  "exemplarCity";
  "براغ";
}
"Riga";
{
```

```
        "exemplarCity";
        "ريغا";
    }
    "Rome";
    {
        "exemplarCity";
        "روما";
    }
    "Samara";
    {
        "exemplarCity";
        "سمراء";
    }
    "San_Marino";
    {
        "exemplarCity";
        "سان مارينو";
    }
    "Sarajevo";
    {
        "exemplarCity";
        "سراييفو";
    }
    "Saratov";
    {
        "exemplarCity";
        "ساراتوف";
    }
    "Simferopol";
    {
        "exemplarCity";
        "سيمفروبول";
    }
    "Skopje";
    {
        "exemplarCity";
        "سكوبي";
    }
    "Sofia";
    {
        "exemplarCity";
        "صوفيا";
    }
    "Stockholm";
    {
        "exemplarCity";
        "ستوكهولم";
    }
    "Tallinn";
    {
        "exemplarCity";
        "تالين";
    }
    "Tirane";
    {
        "exemplarCity";
        "تيرانا";
    }
}
```

```
}
"Ulyanovsk";
{
  "exemplarCity";
  "أوليانوفسك";
}
"Uzhgorod";
{
  "exemplarCity";
  "أوزجروود";
}
"Vaduz";
{
  "exemplarCity";
  "فادوز";
}
"Vatican";
{
  "exemplarCity";
  "الفاتيكان";
}
"Vienna";
{
  "exemplarCity";
  "فيينا";
}
"Vilnius";
{
  "exemplarCity";
  "فيلنيوس";
}
"Volgograd";
{
  "exemplarCity";
  "فولجوراد";
}
"Warsaw";
{
  "exemplarCity";
  "وارسو";
}
"Zagreb";
{
  "exemplarCity";
  "زغرب";
}
"Zaporozhye";
{
  "exemplarCity";
  "زابوروزي";
}
"Zurich";
{
  "exemplarCity";
  "زيورخ";
}
}
```



```
"Africa";
{
  "Abidjan";
  {
    "exemplarCity";
    "أبيدجان";
  }
  "Accra";
  {
    "exemplarCity";
    "أكرا";
  }
  "Addis_Ababa";
  {
    "exemplarCity";
    "أديس أبابا";
  }
  "Algiers";
  {
    "exemplarCity";
    "الجزائر";
  }
  "Asmera";
  {
    "exemplarCity";
    "أسمرّة";
  }
  "Bamako";
  {
    "exemplarCity";
    "باماكو";
  }
  "Bangui";
  {
    "exemplarCity";
    "بانغوي";
  }
  "Banjul";
  {
    "exemplarCity";
    "بانجول";
  }
  "Bissau";
  {
    "exemplarCity";
    "بيساو";
  }
  "Blantyre";
  {
    "exemplarCity";
    "بلانتيير";
  }
  "Brazzaville";
  {
    "exemplarCity";
    "برازافيل";
  }
}
```

```
"Bujumbura";
{
  "exemplarCity";
  "بوجومبورا";
}
"Cairo";
{
  "exemplarCity";
  "القاهرة";
}
"Casablanca";
{
  "exemplarCity";
  "الدار البيضاء";
}
"Ceuta";
{
  "exemplarCity";
  "سيتا";
}
"Conakry";
{
  "exemplarCity";
  "كوناكري";
}
"Dakar";
{
  "exemplarCity";
  "داكار";
}
"Dar_es_Salaam";
{
  "exemplarCity";
  "دار السلام";
}
"Djibouti";
{
  "exemplarCity";
  "جيبوتي";
}
"Douala";
{
  "exemplarCity";
  "دوالا";
}
"El_Aaiun";
{
  "exemplarCity";
  "العيون";
}
"Freetown";
{
  "exemplarCity";
  "فري تاون";
}
"Gaborone";
{
```

```
        "exemplarCity";
        "غابورون";
    }
    "Harare";
    {
        "exemplarCity";
        "هراري";
    }
    "Johannesburg";
    {
        "exemplarCity";
        "جوهانسبرغ";
    }
    "Juba";
    {
        "exemplarCity";
        "جوبا";
    }
    "Kampala";
    {
        "exemplarCity";
        "كامبالا";
    }
    "Khartoum";
    {
        "exemplarCity";
        "الخرطوم";
    }
    "Kigali";
    {
        "exemplarCity";
        "كيغالي";
    }
    "Kinshasa";
    {
        "exemplarCity";
        "كينشاسا";
    }
    "Lagos";
    {
        "exemplarCity";
        "لاغوس";
    }
    "Libreville";
    {
        "exemplarCity";
        "ليبرفيل";
    }
    "Lome";
    {
        "exemplarCity";
        "لومي";
    }
    "Luanda";
    {
        "exemplarCity";
        "لواندا";
    }
}
```

```
}
"Lubumbashi";
{
  "exemplarCity";
  "لومبباشا";
}
"Lusaka";
{
  "exemplarCity";
  "لوساكا";
}
"Malabo";
{
  "exemplarCity";
  "مالابو";
}
"Maputo";
{
  "exemplarCity";
  "مابوتو";
}
"Maseru";
{
  "exemplarCity";
  "ماسيرو";
}
"Mbabane";
{
  "exemplarCity";
  "مباباني";
}
"Mogadishu";
{
  "exemplarCity";
  "مقديشو";
}
"Monrovia";
{
  "exemplarCity";
  "مونروفيا";
}
"Nairobi";
{
  "exemplarCity";
  "نيروبي";
}
"Ndjamena";
{
  "exemplarCity";
  "نجامينا";
}
"Niamey";
{
  "exemplarCity";
  "نيامي";
}
"Nouakchott";
```

```

    {
      "exemplarCity";
      "نواكشوط";
    }
    "Ouagadougou";
    {
      "exemplarCity";
      "واغادوغو";
    }
    "Porto-Novo";
    {
      "exemplarCity";
      "بورتو نوفو";
    }
    "Sao_Tome";
    {
      "exemplarCity";
      "ساو تومي";
    }
    "Tripoli";
    {
      "exemplarCity";
      "طرابلس";
    }
    "Tunis";
    {
      "exemplarCity";
      "تونس";
    }
    "Windhoek";
    {
      "exemplarCity";
      "ويندهوك";
    }
  }
  "Asia";
  {
    "Aden";
    {
      "exemplarCity";
      "عدن";
    }
    "Almaty";
    {
      "exemplarCity";
      "ألماتي";
    }
    "Amman";
    {
      "exemplarCity";
      "عمان";
    }
    "Anadyr";
    {
      "exemplarCity";
      "أندير";
    }
  }

```

```
"Aqtau";
{
  "exemplarCity";
  "أكتاو";
}
"Aqtobe";
{
  "exemplarCity";
  "أكتوب";
}
"Ashgabat";
{
  "exemplarCity";
  "عشق آباد";
}
"Atyrau";
{
  "exemplarCity";
  "أتيراو";
}
"Baghdad";
{
  "exemplarCity";
  "بغداد";
}
"Bahrain";
{
  "exemplarCity";
  "البحرين";
}
"Baku";
{
  "exemplarCity";
  "باكو";
}
"Bangkok";
{
  "exemplarCity";
  "بانكوك";
}
"Barnaul";
{
  "exemplarCity";
  "بارناول";
}
"Beirut";
{
  "exemplarCity";
  "بيروت";
}
"Bishkek";
{
  "exemplarCity";
  "بشكيك";
}
"Brunei";
{
```

```
        "exemplarCity";
        "بروناي";
    }
    "Calcutta";
    {
        "exemplarCity";
        "كالكتا";
    }
    "Chita";
    {
        "exemplarCity";
        "تشيتا";
    }
    "Choibalsan";
    {
        "exemplarCity";
        "تشوبالسان";
    }
    "Colombo";
    {
        "exemplarCity";
        "كولومبو";
    }
    "Damascus";
    {
        "exemplarCity";
        "دمشق";
    }
    "Dhaka";
    {
        "exemplarCity";
        "دكا";
    }
    "Dili";
    {
        "exemplarCity";
        "ديلي";
    }
    "Dubai";
    {
        "exemplarCity";
        "دبي";
    }
    "Dushanbe";
    {
        "exemplarCity";
        "دوشانبي";
    }
    "Famagusta";
    {
        "exemplarCity";
        "فاماغوستا";
    }
    "Gaza";
    {
        "exemplarCity";
        "غزة";
    }
```

```
}
"Hebron";
{
  "exemplarCity";
  "هيبرون (مدينة الخليل)";
}
"Hong_Kong";
{
  "exemplarCity";
  "هونغ كونغ";
}
"Hovd";
{
  "exemplarCity";
  "هوفد";
}
"Irkutsk";
{
  "exemplarCity";
  "ايركيتسك";
}
"Jakarta";
{
  "exemplarCity";
  "جاكرتا";
}
"Jayapura";
{
  "exemplarCity";
  "جاياپورا";
}
"Jerusalem";
{
  "exemplarCity";
  "القدس";
}
"Kabul";
{
  "exemplarCity";
  "كابول";
}
"Kamchatka";
{
  "exemplarCity";
  "كامتشاتكا";
}
"Karachi";
{
  "exemplarCity";
  "كراتشي";
}
"Katmandu";
{
  "exemplarCity";
  "كاتماندو";
}
"Khandyga";
```



```
{
  "exemplarCity";
  "خاندیجا";
}
"Krasnoyarsk";
{
  "exemplarCity";
  "کراسنویارسک";
}
"Kuala_Lumpur";
{
  "exemplarCity";
  "کوالا لامپور";
}
"Kuching";
{
  "exemplarCity";
  "کیشینگ";
}
"Kuwait";
{
  "exemplarCity";
  "الکویت";
}
"Macau";
{
  "exemplarCity";
  "ماکاو";
}
"Magadan";
{
  "exemplarCity";
  "مجادن";
}
"Makassar";
{
  "exemplarCity";
  "ماکسار";
}
"Manila";
{
  "exemplarCity";
  "مانیلا";
}
"Muscat";
{
  "exemplarCity";
  "مسقط";
}
"Nicosia";
{
  "exemplarCity";
  "نیقوسیا";
}
"Novokuznetsk";
{
  "exemplarCity";
```

```

        "نوفوكوزنتسك";
    }
    "Novosibirsk";
    {
        "exemplarCity";
        "نوفوسبیرسك";
    }
    "Omsk";
    {
        "exemplarCity";
        "أومسك";
    }
    "Oral";
    {
        "exemplarCity";
        "أورال";
    }
    "Phnom_Penh";
    {
        "exemplarCity";
        "بنوم بنه";
    }
    "Pontianak";
    {
        "exemplarCity";
        "بونتیانك";
    }
    "Pyongyang";
    {
        "exemplarCity";
        "بیونغ یانغ";
    }
    "Qatar";
    {
        "exemplarCity";
        "قطر";
    }
    "Qyzylorda";
    {
        "exemplarCity";
        "کیزیلورد ا";
    }
    "Rangoon";
    {
        "exemplarCity";
        "رانغون";
    }
    "Riyadh";
    {
        "exemplarCity";
        "الریاض";
    }
    "Saigon";
    {
        "exemplarCity";
        "مدینة هو تشي منه";
    }
}

```

```
"Sakhalin";
{
  "exemplarCity";
  "سكالين";
}
"Samarkand";
{
  "exemplarCity";
  "سمرقند";
}
"Seoul";
{
  "exemplarCity";
  "سول";
}
"Shanghai";
{
  "exemplarCity";
  "شنغهاي";
}
"Singapore";
{
  "exemplarCity";
  "سنغافورة";
}
"Srednekolymsk";
{
  "exemplarCity";
  "سريدنكوليمسك";
}
"Taipei";
{
  "exemplarCity";
  "تايبه";
}
"Tashkent";
{
  "exemplarCity";
  "تشقند";
}
"Tbilisi";
{
  "exemplarCity";
  "تبليسي";
}
"Tehran";
{
  "exemplarCity";
  "تهران";
}
"Thimphu";
{
  "exemplarCity";
  "تيمفو";
}
"Tokyo";
{
```

```

        "exemplarCity";
        "طوكيو";
    }
    "Tomsk";
    {
        "exemplarCity";
        "تومسك";
    }
    "Ulaanbaatar";
    {
        "exemplarCity";
        "آلانباتار";
    }
    "Urumqi";
    {
        "exemplarCity";
        "أرومكي";
    }
    "Ust-Nera";
    {
        "exemplarCity";
        "أوست نيرا";
    }
    "Vientiane";
    {
        "exemplarCity";
        "فيانتيان";
    }
    "Vladivostok";
    {
        "exemplarCity";
        "فلاديفوستك";
    }
    "Yakutsk";
    {
        "exemplarCity";
        "ياكتسك";
    }
    "Yekaterinburg";
    {
        "exemplarCity";
        "يكاترنبيرج";
    }
    "Yerevan";
    {
        "exemplarCity";
        "يريفان";
    }
}
"Indian";
{
    "Antananarivo";
    {
        "exemplarCity";
        "أنتاناناريفو";
    }
}
"Chagos";

```

```
{
  "exemplarCity";
  "تشا غوس";
}
"Christmas";
{
  "exemplarCity";
  "كريسماس";
}
"Cocos";
{
  "exemplarCity";
  "كوكوس";
}
"Comoro";
{
  "exemplarCity";
  "جزر القمر";
}
"Kerguelen";
{
  "exemplarCity";
  "كيرغويلين";
}
"Mahe";
{
  "exemplarCity";
  "ماهي";
}
"Maldives";
{
  "exemplarCity";
  "المالديف";
}
"Mauritius";
{
  "exemplarCity";
  "موريشيوس";
}
"Mayotte";
{
  "exemplarCity";
  "مايوت";
}
"Reunion";
{
  "exemplarCity";
  "ريونيون";
}
}
"Australia";
{
  "Adelaide";
  {
    "exemplarCity";
    "أديليد";
  }
}
```

```
"Brisbane";
{
  "exemplarCity";
  "برسبان";
}
"Broken_Hill";
{
  "exemplarCity";
  "بروكن هيل";
}
"Currie";
{
  "exemplarCity";
  "كوري";
}
"Darwin";
{
  "exemplarCity";
  "دارون";
}
"Eucla";
{
  "exemplarCity";
  "أوكلا";
}
"Hobart";
{
  "exemplarCity";
  "هوبارت";
}
"Lindeman";
{
  "exemplarCity";
  "ليندمان";
}
"Lord_Howe";
{
  "exemplarCity";
  "لورد هاو";
}
"Melbourne";
{
  "exemplarCity";
  "ميلبورن";
}
"Perth";
{
  "exemplarCity";
  "برثا";
}
"Sydney";
{
  "exemplarCity";
  "سيدني";
}
}
"Pacific";
```

```
{
  "Apia";
  {
    "exemplarCity";
    "أبيا";
  }
  "Auckland";
  {
    "exemplarCity";
    "أوكلاند";
  }
  "Bougainville";
  {
    "exemplarCity";
    "بوغانفيل";
  }
  "Chatham";
  {
    "exemplarCity";
    "تشاثام";
  }
  "Easter";
  {
    "exemplarCity";
    "استر";
  }
  "Efate";
  {
    "exemplarCity";
    "إفاته";
  }
  "Enderbury";
  {
    "exemplarCity";
    "اندربيرج";
  }
  "Fakaofu";
  {
    "exemplarCity";
    "فاكاوفو";
  }
  "Fiji";
  {
    "exemplarCity";
    "فيجي";
  }
  "Funafuti";
  {
    "exemplarCity";
    "فونافوتي";
  }
  "Galapagos";
  {
    "exemplarCity";
    "جلاباجوس";
  }
  "Gambier";
```

```
{
  "exemplarCity";
  "جامبير";
}
"Guadalcanal";
{
  "exemplarCity";
  "غواد الكانال";
}
"Guam";
{
  "exemplarCity";
  "غوام";
}
"Honolulu";
{
  "exemplarCity";
  "هونولولو";
}
"Johnston";
{
  "exemplarCity";
  "جونستون";
}
"Kiritimati";
{
  "exemplarCity";
  "كيريتي ماتي";
}
"Kosrae";
{
  "exemplarCity";
  "كوسرا";
}
"Kwajalein";
{
  "exemplarCity";
  "كواجالين";
}
"Majuro";
{
  "exemplarCity";
  "ماجورو";
}
"Marquesas";
{
  "exemplarCity";
  "ماركيساس";
}
"Midway";
{
  "exemplarCity";
  "ميدواي";
}
"Nauru";
{
  "exemplarCity";
```



```
        "ناورو";
    }
    "Niue";
    {
        "exemplarCity";
        "نيوي";
    }
    "Norfolk";
    {
        "exemplarCity";
        "نورفولك";
    }
    "Noumea";
    {
        "exemplarCity";
        "نوميا";
    }
    "Pago_Pago";
    {
        "exemplarCity";
        "باغو باغو";
    }
    "Palau";
    {
        "exemplarCity";
        "بالاو";
    }
    "Pitcairn";
    {
        "exemplarCity";
        "بيتكيرن";
    }
    "Ponape";
    {
        "exemplarCity";
        "باناب";
    }
    "Port_Moresby";
    {
        "exemplarCity";
        "بور مورسبي";
    }
    "Rarotonga";
    {
        "exemplarCity";
        "راروتونغا";
    }
    "Saipan";
    {
        "exemplarCity";
        "سايبان";
    }
    "Tahiti";
    {
        "exemplarCity";
        "تاهيتي";
    }
}
```

```
"Tarawa";
{
  "exemplarCity";
  "تاراوا";
}
"Tongatapu";
{
  "exemplarCity";
  "تونغاتابو";
}
"Truk";
{
  "exemplarCity";
  "ترك";
}
"Wake";
{
  "exemplarCity";
  "واك";
}
"Wallis";
{
  "exemplarCity";
  "واليس";
}
}
"Arctic";
{
  "Longyearbyen";
  {
    "exemplarCity";
    "لونغيربين";
  }
}
"Antarctica";
{
  "Casey";
  {
    "exemplarCity";
    "كاساي";
  }
  "Davis";
  {
    "exemplarCity";
    "دافيز";
  }
  "DumontDUrville";
  {
    "exemplarCity";
    "دي مونت دو روفيل";
  }
  "Macquarie";
  {
    "exemplarCity";
    "ماكوارى";
  }
  "Mawson";
```

```

        {
            "exemplarCity";
            "ماوسون";
        }
        "McMurdo";
        {
            "exemplarCity";
            "ماك موردو";
        }
        "Palmer";
        {
            "exemplarCity";
            "بالمير";
        }
        "Rothera";
        {
            "exemplarCity";
            "روثيرا";
        }
        "Syowa";
        {
            "exemplarCity";
            "سايووا";
        }
        "Troll";
        {
            "exemplarCity";
            "ترول";
        }
        "Vostok";
        {
            "exemplarCity";
            "فوستوك";
        }
    }
    "Etc";
    {
        "UTC";
        {
            "long";
            {
                "standard";
                "التوقيت العالمي المنسق";
            }
            "short";
            {
                "standard";
                "UTC";
            }
        }
    }
    "Unknown";
    {
        "exemplarCity";
        "مدينة غير معروفة";
    }
}

```

```
"metazone";
{
  "Afghanistan";
  {
    "long";
    {
      "standard";
      "توقيت أفغانستان";
    }
  }
  "Africa_Central";
  {
    "long";
    {
      "standard";
      "توقيت وسط أفريقيا";
    }
  }
  "Africa_Eastern";
  {
    "long";
    {
      "standard";
      "توقيت شرق أفريقيا";
    }
  }
  "Africa_Southern";
  {
    "long";
    {
      "standard";
      "توقيت جنوب أفريقيا";
    }
  }
  "Africa_Western";
  {
    "long";
    {
      "generic";
      "توقيت غرب أفريقيا",
      "standard";
      "توقيت غرب أفريقيا الرسمي",
      "daylight";
      "توقيت غرب أفريقيا الصيفي";
    }
  }
  "Alaska";
  {
    "long";
    {
      "generic";
      "توقيت ألاسكا",
      "standard";
      "التوقيت الرسمي لألاسكا",
      "daylight";
      "توقيت ألاسكا الصيفي";
    }
  }
}
```

```

    }
    "Amazon";
    {
        "long";
        {
            "generic";
            "توقيت الأمازون",
            "standard";
            "توقيت الأمازون الرسمي",
            "daylight";
            "توقيت الأمازون الصيفي";
        }
    }
    "America_Central";
    {
        "long";
        {
            "generic";
            "التوقيت المركزي لأمريكا الشمالية",
            "standard";
            "التوقيت الرسمي المركزي لأمريكا الشمالية",
            "daylight";
            "التوقيت الصيفي المركزي لأمريكا الشمالية";
        }
    }
    "America_Eastern";
    {
        "long";
        {
            "generic";
            "التوقيت الشرقي لأمريكا الشمالية",
            "standard";
            "التوقيت الرسمي الشرقي لأمريكا الشمالية",
            "daylight";
            "التوقيت الصيفي الشرقي لأمريكا الشمالية";
        }
    }
    "America_Mountain";
    {
        "long";
        {
            "generic";
            "التوقيت الجبلي لأمريكا الشمالية",
            "standard";
            "التوقيت الجبلي الرسمي لأمريكا الشمالية",
            "daylight";
            "التوقيت الجبلي الصيفي لأمريكا الشمالية";
        }
    }
    "America_Pacific";
    {
        "long";
        {
            "generic";
            "توقيت المحيط الهادي",
            "standard";
            "توقيت المحيط الهادي الرسمي",

```

```

        "daylight";
        "توقيت المحيط الهادي الصيفي";
    }
}
"Anadyr";
{
    "long";
    {
        "generic";
        "توقيت أنادير",
        "standard";
        "توقيت أنادير الرسمي",
        "daylight";
        "التوقيت الصيفي لأنادير";
    }
}
"Apia";
{
    "long";
    {
        "generic";
        "توقيت آبيا",
        "standard";
        "التوقيت الرسمي لآبيا",
        "daylight";
        "التوقيت الصيفي لآبيا";
    }
}
"Arabian";
{
    "long";
    {
        "generic";
        "التوقيت العربي",
        "standard";
        "التوقيت العربي الرسمي",
        "daylight";
        "التوقيت العربي الصيفي";
    }
}
"Argentina";
{
    "long";
    {
        "generic";
        "توقيت الأرجنتين",
        "standard";
        "توقيت الأرجنتين الرسمي",
        "daylight";
        "توقيت الأرجنتين الصيفي";
    }
}
"Argentina_Western";
{
    "long";
    {
        "generic";

```

```

        "توقيت غرب الأرجنتين",
        "standard";
        "توقيت غرب الأرجنتين الرسمي",
        "daylight";
        "توقيت غرب الأرجنتين الصيفي";
    }
}
"Armenia";
{
    "long";
    {
        "generic";
        "توقيت أرمينيا",
        "standard";
        "توقيت أرمينيا الرسمي",
        "daylight";
        "توقيت أرمينيا الصيفي";
    }
}
"Atlantic";
{
    "long";
    {
        "generic";
        "توقيت الأطلسي",
        "standard";
        "التوقيت الرسمي الأطلسي",
        "daylight";
        "التوقيت الصيفي الأطلسي";
    }
}
"Australia_Central";
{
    "long";
    {
        "generic";
        "توقيت وسط أستراليا",
        "standard";
        "توقيت وسط أستراليا الرسمي",
        "daylight";
        "توقيت وسط أستراليا الصيفي";
    }
}
"Australia_CentralWestern";
{
    "long";
    {
        "generic";
        "توقيت غرب وسط أستراليا",
        "standard";
        "توقيت غرب وسط أستراليا الرسمي",
        "daylight";
        "توقيت غرب وسط أستراليا الصيفي";
    }
}
"Australia_Eastern";
{

```

```

        "long";
        {
            "generic";
            "توقيت شرق أستراليا",
            "standard";
            "توقيت شرق أستراليا الرسمي",
            "daylight";
            "توقيت شرق أستراليا الصيفي";
        }
    }
    "Australia_Western";
    {
        "long";
        {
            "generic";
            "توقيت غرب أستراليا",
            "standard";
            "توقيت غرب أستراليا الرسمي",
            "daylight";
            "توقيت غرب أستراليا الصيفي";
        }
    }
    "Azerbaijan";
    {
        "long";
        {
            "generic";
            "توقيت أذربيجان",
            "standard";
            "توقيت أذربيجان الرسمي",
            "daylight";
            "توقيت أذربيجان الصيفي";
        }
    }
    "Azores";
    {
        "long";
        {
            "generic";
            "توقيت أزورس",
            "standard";
            "توقيت أزورس الرسمي",
            "daylight";
            "توقيت أزورس الصيفي";
        }
    }
    "Bangladesh";
    {
        "long";
        {
            "generic";
            "توقيت بنغلاديش",
            "standard";
            "توقيت بنغلاديش الرسمي",
            "daylight";
            "توقيت بنغلاديش الصيفي";
        }
    }

```



```
}
"Bhutan";
{
  "long";
  {
    "standard";
    "توقيت بوتان";
  }
}
"Bolivia";
{
  "long";
  {
    "standard";
    "توقيت بوليفيا";
  }
}
"Brasilia";
{
  "long";
  {
    "generic";
    "توقيت برازيليا",
    "standard";
    "توقيت برازيليا الرسمي",
    "daylight";
    "توقيت برازيليا الصيفي";
  }
}
"Brunei";
{
  "long";
  {
    "standard";
    "توقيت بروناي";
  }
}
"Cape_Verde";
{
  "long";
  {
    "generic";
    "توقيت الرأس الأخضر",
    "standard";
    "توقيت الرأس الأخضر الرسمي",
    "daylight";
    "توقيت الرأس الأخضر الصيفي";
  }
}
"Chamorro";
{
  "long";
  {
    "standard";
    "توقيت تشامورو";
  }
}
}
```

```
"Chatham";
{
  "long";
  {
    "generic";
    "توقيت تشاتام",
    "standard";
    "توقيت تشاتام الرسمي",
    "daylight";
    "توقيت تشاتام الصيفي";
  }
}
"Chile";
{
  "long";
  {
    "generic";
    "توقيت شيلي",
    "standard";
    "توقيت شيلي الرسمي",
    "daylight";
    "توقيت شيلي الصيفي";
  }
}
"China";
{
  "long";
  {
    "generic";
    "توقيت الصين",
    "standard";
    "توقيت الصين الرسمي",
    "daylight";
    "توقيت الصين الصيفي";
  }
}
"Choibalsan";
{
  "long";
  {
    "generic";
    "توقيت شويبالسان",
    "standard";
    "توقيت شويبالسان الرسمي",
    "daylight";
    "التوقيت الصيفي لشويبالسان";
  }
}
"Christmas";
{
  "long";
  {
    "standard";
    "توقيت جزر الكريسماس";
  }
}
"Cocos";
```

```

    {
      "long";
      {
        "standard";
        "توقيت جزر كوكوس";
      }
    }
    "Colombia";
    {
      "long";
      {
        "generic";
        "توقيت كولومبيا",
        "standard";
        "توقيت كولومبيا الرسمي",
        "daylight";
        "توقيت كولومبيا الصيفي";
      }
    }
    "Cook";
    {
      "long";
      {
        "generic";
        "توقيت جزر كوك",
        "standard";
        "توقيت جزر كوك الرسمي",
        "daylight";
        "توقيت جزر كوك الصيفي";
      }
    }
    "Cuba";
    {
      "long";
      {
        "generic";
        "توقيت كوبا",
        "standard";
        "توقيت كوبا الرسمي",
        "daylight";
        "توقيت كوبا الصيفي";
      }
    }
    "Davis";
    {
      "long";
      {
        "standard";
        "توقيت دافيز";
      }
    }
    "DumontDUrville";
    {
      "long";
      {
        "standard";
        "توقيت دي مونت دو روفيل";
      }
    }
  }

```

```

    }
  }
  "East_Timor";
  {
    "long";
    {
      "standard";
      "توقيت تيمور الشرقية";
    }
  }
  "Easter";
  {
    "long";
    {
      "generic";
      "توقيت جزيرة استر",
      "standard";
      "توقيت جزيرة استر الرسمي",
      "daylight";
      "توقيت جزيرة استر الصيفي";
    }
  }
  "Ecuador";
  {
    "long";
    {
      "standard";
      "توقيت الإكوادور";
    }
  }
  "Europe_Central";
  {
    "long";
    {
      "generic";
      "توقيت وسط أوروبا",
      "standard";
      "توقيت وسط أوروبا الرسمي",
      "daylight";
      "توقيت وسط أوروبا الصيفي";
    }
  }
  "Europe_Eastern";
  {
    "long";
    {
      "generic";
      "توقيت شرق أوروبا",
      "standard";
      "توقيت شرق أوروبا الرسمي",
      "daylight";
      "توقيت شرق أوروبا الصيفي";
    }
  }
  "Europe_Further_Eastern";
  {
    "long";

```

```

        {
            "standard";
            "التوقيت الأوروبي ( أكثر شرقًا )";
        }
    }
    "Europe_Western";
    {
        "long";
        {
            "generic";
            "توقيت غرب أوروبا",
            "standard";
            "توقيت غرب أوروبا الرسمي",
            "daylight";
            "توقيت غرب أوروبا الصيفي";
        }
    }
    "Falkland";
    {
        "long";
        {
            "generic";
            "توقيت جزر فوكلاند",
            "standard";
            "توقيت جزر فوكلاند الرسمي",
            "daylight";
            "توقيت جزر فوكلاند الصيفي";
        }
    }
    "Fiji";
    {
        "long";
        {
            "generic";
            "توقيت فيجي",
            "standard";
            "توقيت فيجي الرسمي",
            "daylight";
            "توقيت فيجي الصيفي";
        }
    }
    "French_Guiana";
    {
        "long";
        {
            "standard";
            "توقيت غايانا الفرنسية";
        }
    }
    "French_Southern";
    {
        "long";
        {
            "standard";
            "توقيت المقاطعات الفرنسية الجنوبية";
        }
    }
    "والأنتارتيكية";
}

```

```

    }
    "Galapagos";
    {
        "long";
        {
            "standard";
            "توقيت غلابا غوس";
        }
    }
    "Gambier";
    {
        "long";
        {
            "standard";
            "توقيت جامبير";
        }
    }
    "Georgia";
    {
        "long";
        {
            "generic";
            "توقيت جورجيا",
            "standard";
            "توقيت جورجيا الرسمي",
            "daylight";
            "توقيت جورجيا الصيفي";
        }
    }
    "Gilbert_Islands";
    {
        "long";
        {
            "standard";
            "توقيت جزر جيلبرت";
        }
    }
    "GMT";
    {
        "long";
        {
            "standard";
            "توقيت غرينتش";
        }
    }
    "Greenland_Eastern";
    {
        "long";
        {
            "generic";
            "توقيت شرق غرينلاند",
            "standard";
            "توقيت شرق غرينلاند الرسمي",
            "daylight";
            "توقيت شرق غرينلاند الصيفي";
        }
    }
}

```

```
"Greenland_Western";
{
  "long";
  {
    "generic";
    "توقيت غرب غرينلاند",
    "standard";
    "توقيت غرب غرينلاند الرسمي",
    "daylight";
    "توقيت غرب غرينلاند الصيفي";
  }
}
"Guam";
{
  "long";
  {
    "standard";
    "توقيت غوام";
  }
}
"Gulf";
{
  "long";
  {
    "standard";
    "توقيت الخليج";
  }
}
"Guyana";
{
  "long";
  {
    "standard";
    "توقيت غيانا";
  }
}
"Hawaii_Aleutian";
{
  "long";
  {
    "generic";
    "توقيت هاواي ألوتيان",
    "standard";
    "توقيت هاواي ألوتيان الرسمي",
    "daylight";
    "توقيت هاواي ألوتيان الصيفي";
  }
}
"Hong_Kong";
{
  "long";
  {
    "generic";
    "توقيت هونغ كونغ",
    "standard";
    "توقيت هونغ كونغ الرسمي",
    "daylight";
  }
}
```

```

        "توقيت هونغ كونغ الصيفي";
    }
}
"Hovd";
{
    "long";
    {
        "generic";
        "توقيت هوفد",
        "standard";
        "توقيت هوفد الرسمي",
        "daylight";
        "توقيت هوفد الصيفي";
    }
}
"India";
{
    "long";
    {
        "standard";
        "توقيت الهند";
    }
}
"Indian_Ocean";
{
    "long";
    {
        "standard";
        "توقيت المحيط الهندي";
    }
}
"Indochina";
{
    "long";
    {
        "standard";
        "توقيت الهند الصينية";
    }
}
"Indonesia_Central";
{
    "long";
    {
        "standard";
        "توقيت وسط إندونيسيا";
    }
}
"Indonesia_Eastern";
{
    "long";
    {
        "standard";
        "توقيت شرق إندونيسيا";
    }
}
"Indonesia_Western";
{

```



```

        "long";
        {
            "standard";
            "توقيت غرب إندونيسيا";
        }
    }
    "Iran";
    {
        "long";
        {
            "generic";
            "توقيت إيران",
            "standard";
            "توقيت إيران الرسمي",
            "daylight";
            "توقيت إيران الصيفي";
        }
    }
    "Irkutsk";
    {
        "long";
        {
            "generic";
            "توقيت إركوتسك",
            "standard";
            "توقيت إركوتسك الرسمي",
            "daylight";
            "توقيت إركوتسك الصيفي";
        }
    }
    "Israel";
    {
        "long";
        {
            "generic";
            "توقيت إسرائيل",
            "standard";
            "توقيت إسرائيل الرسمي",
            "daylight";
            "توقيت إسرائيل الصيفي";
        }
    }
    "Japan";
    {
        "long";
        {
            "generic";
            "توقيت اليابان",
            "standard";
            "توقيت اليابان الرسمي",
            "daylight";
            "توقيت اليابان الصيفي";
        }
    }
    "Kamchatka";
    {
        "long";

```

```

        {
            "generic";
            "توقيت كامشاتكا",
            "standard";
            "توقيت بيتروبافلوفسك-كامشاتسكي",
            "daylight";
            "توقيت بيتروبافلوفسك-كامشاتسكي الصيفي";
        }
    }
    "Kazakhstan_Eastern";
    {
        "long";
        {
            "standard";
            "توقيت شرق كازاخستان";
        }
    }
    "Kazakhstan_Western";
    {
        "long";
        {
            "standard";
            "توقيت غرب كازاخستان";
        }
    }
    "Korea";
    {
        "long";
        {
            "generic";
            "توقيت كوريا",
            "standard";
            "توقيت كوريا الرسمي",
            "daylight";
            "توقيت كوريا الصيفي";
        }
    }
    "Kosrae";
    {
        "long";
        {
            "standard";
            "توقيت كوسرا";
        }
    }
    "Krasnoyarsk";
    {
        "long";
        {
            "generic";
            "توقيت كراسنويارسك",
            "standard";
            "توقيت كراسنويارسك الرسمي",
            "daylight";
            "التوقيت الصيفي لكراسنويارسك";
        }
    }
}

```

```
"Kyrgystan";
{
  "long";
  {
    "standard";
    "توقيت قيرغيزستان";
  }
}
"Line_Islands";
{
  "long";
  {
    "standard";
    "توقيت جزر لاين";
  }
}
"Lord_Howe";
{
  "long";
  {
    "generic";
    "توقيت لورد هاو",
    "standard";
    "توقيت لورد هاو الرسمي",
    "daylight";
    "التوقيت الصيفي للورد هاو";
  }
}
"Macquarie";
{
  "long";
  {
    "standard";
    "توقيت ماكوارى";
  }
}
"Magadan";
{
  "long";
  {
    "generic";
    "توقيت ماغادان",
    "standard";
    "توقيت ماغادان الرسمي",
    "daylight";
    "توقيت ماغادان الصيفي";
  }
}
"Malaysia";
{
  "long";
  {
    "standard";
    "توقيت ماليزيا";
  }
}
"Maldives";
```

```
{
  "long";
  {
    "standard";
    "توقيت المالديف";
  }
}
"Marquesas";
{
  "long";
  {
    "standard";
    "توقيت ماركيساس";
  }
}
"Marshall_Islands";
{
  "long";
  {
    "standard";
    "توقيت جزر مارشال";
  }
}
"Mauritius";
{
  "long";
  {
    "generic";
    "توقيت موريشيوس",
    "standard";
    "توقيت موريشيوس الرسمي",
    "daylight";
    "توقيت موريشيوس الصيفي";
  }
}
"Mawson";
{
  "long";
  {
    "standard";
    "توقيت ماوسون";
  }
}
"Mexico_Northwest";
{
  "long";
  {
    "generic";
    "توقيت شمال غرب المكسيك",
    "standard";
    "التوقيت الرسمي لشمال غرب المكسيك",
    "daylight";
    "التوقيت الصيفي لشمال غرب المكسيك";
  }
}
"Mexico_Pacific";
{
```

```

        "long";
        {
            "generic";
            "توقيت المحيط الهادي للمكسيك",
            "standard";
            "توقيت المحيط الهادي الرسمي للمكسيك",
            "daylight";
            "توقيت المحيط الهادي الصيفي للمكسيك";
        }
    }
    "Mongolia";
    {
        "long";
        {
            "generic";
            "توقيت أولان باتور",
            "standard";
            "توقيت أولان باتور الرسمي",
            "daylight";
            "توقيت أولان باتور الصيفي";
        }
    }
    "Moscow";
    {
        "long";
        {
            "generic";
            "توقيت موسكو",
            "standard";
            "توقيت موسكو الرسمي",
            "daylight";
            "توقيت موسكو الصيفي";
        }
    }
    "Myanmar";
    {
        "long";
        {
            "standard";
            "توقيت ميانمار";
        }
    }
    "Nauru";
    {
        "long";
        {
            "standard";
            "توقيت ناورو";
        }
    }
    "Nepal";
    {
        "long";
        {
            "standard";
            "توقيت نيبال";
        }
    }

```

```
}
    "New_Caledonia";
    {
        "long";
        {
            "generic";
            "توقيت كاليدونيا الجديدة",
            "standard";
            "توقيت كاليدونيا الجديدة الرسمي",
            "daylight";
            "توقيت كاليدونيا الجديدة الصيفي";
        }
    }
    "New_Zealand";
    {
        "long";
        {
            "generic";
            "توقيت نيوزيلندا",
            "standard";
            "توقيت نيوزيلندا الرسمي",
            "daylight";
            "توقيت نيوزيلندا الصيفي";
        }
    }
    "Newfoundland";
    {
        "long";
        {
            "generic";
            "توقيت نيوفاوندلاند",
            "standard";
            "توقيت نيوفاوندلاند الرسمي",
            "daylight";
            "توقيت نيوفاوندلاند الصيفي";
        }
    }
    "Niue";
    {
        "long";
        {
            "standard";
            "توقيت نيوي";
        }
    }
    "Norfolk";
    {
        "long";
        {
            "standard";
            "توقيت جزيرة نورفولك";
        }
    }
    "Noronha";
    {
        "long";
        {
```

```

        "generic";
        "توقيت فيرناندو دي نورونها",
        "standard";
        "توقيت فرناندو دي نورونها الرسمي",
        "daylight";
        "توقيت فرناندو دي نورونها الصيفي";
    }
}
"North_Mariana";
{
    "long";
    {
        "standard";
        "توقيت جزر ماريانا الشمالية";
    }
}
"Novosibirsk";
{
    "long";
    {
        "generic";
        "توقيت نوفوسيبيرسك",
        "standard";
        "توقيت نوفوسيبيرسك الرسمي",
        "daylight";
        "توقيت نوفوسيبيرسك الصيفي";
    }
}
"Omsk";
{
    "long";
    {
        "generic";
        "توقيت أومسك",
        "standard";
        "توقيت أومسك الرسمي",
        "daylight";
        "توقيت أومسك الصيفي";
    }
}
"Pakistan";
{
    "long";
    {
        "generic";
        "توقيت باكستان",
        "standard";
        "توقيت باكستان الرسمي",
        "daylight";
        "توقيت باكستان الصيفي";
    }
}
"Palau";
{
    "long";
    {
        "standard";

```

```

        "توقيت بالاو";
    }
}
"Papua_New_Guinea";
{
    "long";
    {
        "standard";
        "توقيت بابوا غينيا الجديدة";
    }
}
"Paraguay";
{
    "long";
    {
        "generic";
        "توقيت باراغواي",
        "standard";
        "توقيت باراغواي الرسمي",
        "daylight";
        "توقيت باراغواي الصيفي";
    }
}
"Peru";
{
    "long";
    {
        "generic";
        "توقيت بيرو",
        "standard";
        "توقيت بيرو الرسمي",
        "daylight";
        "توقيت بيرو الصيفي";
    }
}
"Philippines";
{
    "long";
    {
        "generic";
        "توقيت الفلبين",
        "standard";
        "توقيت الفلبين الرسمي",
        "daylight";
        "توقيت الفلبين الصيفي";
    }
}
"Phoenix_Islands";
{
    "long";
    {
        "standard";
        "توقيت جزر فينكس";
    }
}
"Pierre_Miquelon";
{

```



```

        "long";
        {
            "generic";
            "توقيت سانت بيير وميكلون",
            "standard";
            "توقيت سانت بيير وميكلون الرسمي",
            "daylight";
            "توقيت سانت بيير وميكلون الصيفي";
        }
    }
    "Pitcairn";
    {
        "long";
        {
            "standard";
            "توقيت بيتكيرن";
        }
    }
    "Ponape";
    {
        "long";
        {
            "standard";
            "توقيت بونابي";
        }
    }
    "Pyongyang";
    {
        "long";
        {
            "standard";
            "توقيت بيونغ يانغ";
        }
    }
    "Reunion";
    {
        "long";
        {
            "standard";
            "توقيت ريونيون";
        }
    }
    "Rothera";
    {
        "long";
        {
            "standard";
            "توقيت روثيرا";
        }
    }
    "Sakhalin";
    {
        "long";
        {
            "generic";
            "توقيت ساخالين",
            "standard";
        }
    }

```

```

        "توقيت ساخالين الرسمي",
        "daylight";
        "توقيت ساخالين الصيفي";
    }
}
"Samara";
{
    "long";
    {
        "generic";
        "توقيت سامارا",
        "standard";
        "توقيت سامارا",
        "daylight";
        "توقيت سامارا الصيفي";
    }
}
"Samoa";
{
    "long";
    {
        "generic";
        "توقيت ساموا",
        "standard";
        "توقيت ساموا الرسمي",
        "daylight";
        "توقيت ساموا الصيفي";
    }
}
"Seychelles";
{
    "long";
    {
        "standard";
        "توقيت سيشل";
    }
}
"Singapore";
{
    "long";
    {
        "standard";
        "توقيت سنغافورة";
    }
}
"Solomon";
{
    "long";
    {
        "standard";
        "توقيت جزر سليمان";
    }
}
"South_Georgia";
{
    "long";
    {

```

```

        "standard";
        "توقيت جنوب جورجيا";
    }
}
"Suriname";
{
    "long";
    {
        "standard";
        "توقيت سورينام";
    }
}
"Syowa";
{
    "long";
    {
        "standard";
        "توقيت سايووا";
    }
}
"Tahiti";
{
    "long";
    {
        "standard";
        "توقيت تاهيتي";
    }
}
"Taipei";
{
    "long";
    {
        "generic";
        "توقيت تايبه",
        "standard";
        "توقيت تايبه الرسمي",
        "daylight";
        "توقيت تايبه الصيفي";
    }
}
"Tajikistan";
{
    "long";
    {
        "standard";
        "توقيت طاجكستان";
    }
}
"Tokelau";
{
    "long";
    {
        "standard";
        "توقيت توكيلاو";
    }
}
"Tonga";

```

```

    {
      "long";
      {
        "generic";
        "توقيت تونغنا",
        "standard";
        "توقيت تونغنا الرسمي",
        "daylight";
        "توقيت تونغنا الصيفي";
      }
    }
    "Truk";
    {
      "long";
      {
        "standard";
        "توقيت شوك";
      }
    }
    "Turkmenistan";
    {
      "long";
      {
        "generic";
        "توقيت تركمانستان",
        "standard";
        "توقيت تركمانستان الرسمي",
        "daylight";
        "توقيت تركمانستان الصيفي";
      }
    }
    "Tuvalu";
    {
      "long";
      {
        "standard";
        "توقيت توفالو";
      }
    }
    "Uruguay";
    {
      "long";
      {
        "generic";
        "توقيت أوروغواي",
        "standard";
        "توقيت أوروغواي الرسمي",
        "daylight";
        "توقيت أوروغواي الصيفي";
      }
    }
    "Uzbekistan";
    {
      "long";
      {
        "generic";
        "توقيت أوزبكستان",

```

```

        "standard";
        "توقيت أوزبكستان الرسمي",
        "daylight";
        "توقيت أوزبكستان الصيفي";
    }
}
"Vanuatu";
{
    "long";
    {
        "generic";
        "توقيت فانواتو",
        "standard";
        "توقيت فانواتو الرسمي",
        "daylight";
        "توقيت فانواتو الصيفي";
    }
}
"Venezuela";
{
    "long";
    {
        "standard";
        "توقيت فنزويلا";
    }
}
"Vladivostok";
{
    "long";
    {
        "generic";
        "توقيت فلاديفوستوك",
        "standard";
        "توقيت فلاديفوستوك الرسمي",
        "daylight";
        "توقيت فلاديفوستوك الصيفي";
    }
}
"Volgograd";
{
    "long";
    {
        "generic";
        "توقيت فولغوغراد",
        "standard";
        "توقيت فولغوغراد الرسمي",
        "daylight";
        "توقيت فولغوغراد الصيفي";
    }
}
"Vostok";
{
    "long";
    {
        "standard";
        "توقيت فوستوك";
    }
}

```

```

    }
    "Wake";
    {
        "long";
        {
            "standard";
            "توقيت جزيرة ويك";
        }
    }
    "Wallis";
    {
        "long";
        {
            "standard";
            "توقيت واليس و فوتونا";
        }
    }
    "Yakutsk";
    {
        "long";
        {
            "generic";
            "توقيت ياكوتسك",
            "standard";
            "توقيت ياكوتسك الرسمي",
            "daylight";
            "توقيت ياكوتسك الصيفي";
        }
    }
    "Yekaterinburg";
    {
        "long";
        {
            "generic";
            "توقيت يكاترينبورغ",
            "standard";
            "توقيت يكاترينبورغ الرسمي",
            "daylight";
            "توقيت يكاترينبورغ الصيفي";
        }
    }
}
}
}
}
}
}
}
}
}
}

```

Strict mode in React Datetimepicker component

The [strictMode](#) is an act, that allows the user to enter only the valid date and time within the specified min/max range in textbox. If the input entered is invalid, then the component will stay with the previous value. Else, if the date and time is out of range, then the component will set the date to the min/max value.

The following example demonstrates the DateTimePicker in `strictMode` with min/max range of 5/5/2019 2:00 AM to

5/25/2019 2:00 AM. Here, it allows to enter only the valid date and time within the specified range. If you are trying to enter the out-of-range value as like 5/28/2019, then the value will set to the `max` value as 5/25/2019 2:00 AM. Since the value 28 is greater than to `max` value of 25. Or else if you are trying to enter the invalid date, then the value will stay with the previous value.

[Class-component]

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  minDate = new Date('5/5/2019 2:00 AM');
  maxDate = new Date('5/25/2019 2:00 AM');
  dateValue = new Date('5/28/2019 2:00 AM');
  enable = true;
  render() {
    return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={this.enable}
max={this.maxDate}/>;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private minDate: Date = new Date('5/5/2019 2:00 AM');
  private maxDate: Date = new Date('5/25/2019 2:00 AM');
  private dateValue: Date = new Date('5/28/2019 2:00 AM');
  private enable: boolean = true;
  public render() {
    return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={this.enable}
max={this.maxDate} />;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  const minDate = new Date('5/5/2019 2:00 AM');
  const maxDate = new Date('5/25/2019 2:00 AM');
```

```

const dateValue = new Date('5/28/2019 2:00 AM');
const enable = true;
return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={enable} max={maxDate}/>;
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const minDate: Date = new Date('5/5/2019 2:00 AM');
    const maxDate: Date = new Date('5/25/2019 2:00 AM');
    const dateValue: Date = new Date('5/28/2019 2:00 AM');
    const enable: boolean = true;
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={enable} max={maxDate} />;
}
ReactDOM.render(<App />, document.getElementById('element'));

```

By default, the DateTimePicker act in strictMode `false` state, that allows to enter the invalid or out-of-range datetime in textbox.

If the datetime is out-of-range or invalid, then the model value will be set to `out of range` datetime value or `null` respectively with highlighted `error` class to indicates the datetime is out of range or invalid.

The following example demonstrates the strictMode as `false`. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid datetime value, then the model value will be set to `out of range` datetime value or `null` respectively with highlighted `error` class to indicates the datetime is out of range or invalid.

[Class-component]

INDEX.JSX

```

import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    minDate = new Date('5/5/2019 2:00 AM');
    maxDate = new Date('5/25/2019 2:00 AM');
    dateValue = new Date('5/28/2019 2:00 AM');
    enable = false;
    render() {
        return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={this.enable}
max={this.maxDate} placeholder='Select a date and time' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX


```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    private minDate: Date =new Date('5/5/2019 2:00 AM');
    private maxDate: Date =new Date('5/25/2019 2:00 AM');
    private dateValue: Date = new Date('5/28/2019 2:00 AM');
    private enable: boolean = false;
    public render() {
        return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={this.enable}
max={this.maxDate} placeholder='Select a date and time' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const minDate = new Date('5/5/2019 2:00 AM');
    const maxDate = new Date('5/25/2019 2:00 AM');
    const dateValue = new Date('5/28/2019 2:00 AM');
    const enable = false;
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={enable} max={maxDate} placeholder='Select a date
and time' />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const minDate: Date =new Date('5/5/2019 2:00 AM');
    const maxDate: Date =new Date('5/25/2019 2:00 AM');
    const dateValue: Date = new Date('5/28/2019 2:00 AM');
    const enable: boolean = false;
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={enable} max={maxDate} placeholder='Select a date
and time' />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date time range in React Datetimepicker component

DateTimePicker provides an option to select a date and time value within a specified range by using the [min](#) and [max](#) properties. Always the min value has to be lesser than the max value.

When the min and max properties are configured and the selected datetime value is out-of-range or invalid, then the model value will be set to **out of range** datetime value or **null** respectively with highlighted **error** class to indicates the datetime is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The below example allows selecting a date within the range from 7th to 27th day in a month.

[Class-component]

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  minDate = new Date(new Date().getFullYear(), new Date().getMonth(), 7,
0, 0, 0);
  maxDate = new Date(new Date().getFullYear(), new Date().getMonth(), 27,
new Date().getHours(), new Date().getMinutes(), new Date().getSeconds());
  dateValue = new Date(new Date().setDate(14));
  render() {
    return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={true}
max={this.maxDate}/>;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private minDate: Date =new Date(new Date().getFullYear(), new
Date().getMonth(), 7, 0, 0, 0);
  private maxDate: Date =new Date(new Date().getFullYear(), new
Date().getMonth(), 27,new Date().getHours(),new Date().getMinutes(),new
Date().getSeconds());
  private dateValue: Date = new Date(new Date().setDate(14));
  public render() {
    return <DateTimePickerComponent id="datetimepicker"
value={this.dateValue} min={this.minDate} strictMode={true}
max={this.maxDate} />;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const minDate = new Date(new Date().getFullYear(), new
Date().getMonth(), 7, 0, 0, 0);
    const maxDate = new Date(new Date().getFullYear(), new
Date().getMonth(), 27, new Date().getHours(), new Date().getMinutes(), new
Date().getSeconds());
    const dateValue = new Date(new Date().setDate(14));
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={true} max={maxDate}/>;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const minDate: Date = new Date(new Date().getFullYear(), new
Date().getMonth(), 7, 0, 0, 0);
    const maxDate: Date = new Date(new Date().getFullYear(), new
Date().getMonth(), 27, new Date().getHours(), new Date().getMinutes(), new
Date().getSeconds());
    const dateValue: Date = new Date(new Date().setDate(14));
    return <DateTimePickerComponent id="datetimepicker" value={dateValue}
min={minDate} strictMode={true} max={maxDate} />;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

If the value of `min` or `max` properties changed through code behind, then you have to update the `value` property to set within the range.

Date time masking in React Datetimepicker component

DateTimePicker has `enableMask` property that provides the option to enable the built-in date masking support. Also, you must inject the `MaskedDateTime` module to enable the masking support.

[Class-component]

INDEX.JSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
export default class App extends React.Component {
    render() {
        return <DateTimePickerComponent enableMask={true}><Inject
services={[MaskedDateTime]} /></DateTimePickerComponent>;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
export default class App extends React.Component<{}, {}> {
  public render() {
    return <DateTimePickerComponent enableMask={true}><Inject
    services=[MaskedDateTime] /></DateTimePickerComponent>
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]**INDEX.JSX**

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
  return <DateTimePickerComponent enableMask={true}><Inject
  services=[MaskedDateTime] /></DateTimePickerComponent>;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
  return <DateTimePickerComponent enableMask={true}><Inject
  services=[MaskedDateTime] /></DateTimePickerComponent>
}
ReactDOM.render(<App />, document.getElementById('element'));
```

The mask pattern is defined based on the provided date format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys** | **Actions** |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the date and time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of DateTimePicker component with mask.

[Class-component]

INDEX.JSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
'@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
export default class App extends React.Component {
  render() {
    return (<div>
      /* specifies the masked DateTimePicker component without
format */
      <DateTimePickerComponent enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
      <br />
      <br />
      /* specifies the masked DateTimePicker component with
format */
      <DateTimePickerComponent format='dd-MM-yyyy hh:mm a'
enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
    </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
'@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
export default class App extends React.Component<{}, {}> {
  public render() {
    return (
      <div>
        /* specifies the masked DateTimePicker component without
format */
        <DateTimePickerComponent enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
        <br />
        <br />
        /* specifies the masked DateTimePicker component with
format */
        <DateTimePickerComponent format='dd-MM-yyyy hh:mm a'
enableMask={true}><Inject services=[MaskedDateTime]
/></DateTimePickerComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

[Functional-component]**INDEX.JSX**

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
    return (<div>
        /* specifies the masked DateTimePicker component without
        format */
        <DateTimePickerComponent enableMask={true}><Inject
        services={ [MaskedDateTime] }/></DateTimePickerComponent>
        <br />
        <br />
        /* specifies the masked DateTimePicker component with
        format */
        <DateTimePickerComponent format='dd-MM-yyyy hh:mm a'
        enableMask={true}><Inject
        services={ [MaskedDateTime] }/></DateTimePickerComponent>
        </div>);
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
    return (
        <div>
            /* specifies the masked DateTimePicker component without
            format */
            <DateTimePickerComponent enableMask={true}><Inject
            services={ [MaskedDateTime] } /></DateTimePickerComponent>
            <br />
            <br />
            /* specifies the masked DateTimePicker component with
            format */
            <DateTimePickerComponent format='dd-MM-yyyy hh:mm a'
            enableMask={true}><Inject services={ [MaskedDateTime] }
            /></DateTimePickerComponent>
            </div>
        );
    }
    ReactDOM.render(<App />, document.getElementById('element'));
```

Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of date and time co-ordinates such as `day`, `month`, `year`, `hour` etc.

While changing to a culture other than `English`, ensure that locale text for the concerned culture is loaded through load method of `L10n` class for mask placeholder values like below.

```
`ts
```

```
//Load the L10n from ej2-base
import { L10n } from '@syncfusion/ej2-base';
//load the locale object to set the localized mask placeholder value
L10n.load({
  'de': {
    datetimepicker: { day: 'Tag', month: 'Monat', year: 'Jahr', hour: 'Stunde', minute: 'Minute',
      second: 'Sekunden' }
  }
});
`
```

The following example demonstrates default and customized mask placeholder value.

[Class-component]

INDEX.JSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
export default class App extends React.Component {
  maskPlaceholderValue;
  constructor(props) {
    super(props);
    this.maskPlaceholderValue = { day: 'd', month: 'M', year: 'y', hour:
    'h', minute: 'm', second: 's' };
  }
  render() {
    return (<div>
      /* specifies the masked DateTimePicker component without
      mask placeholder */
      <DateTimePickerComponent enableMask={true}><Inject
      services={ [MaskedDateTime] }/></DateTimePickerComponent>
      <br />
      <br />
      /* specifies the masked DateTimePicker component with mask
      placeholder */
      <DateTimePickerComponent enableMask={true}
      maskPlaceholder={this.maskPlaceholderValue}><Inject
      services={ [MaskedDateTime] }/></DateTimePickerComponent>
      </div>);
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
```

```

import * as React from 'react';
export default class App extends React.Component<{}, {}> {
  private maskPlaceholderValue: object;
  constructor(props: {}) {
    super(props);
    this.maskPlaceholderValue = {day: 'd', month: 'M', year: 'y', hour: 'h',
minute: 'm', second: 's'};
  }
  public render() {
    return (
      <div>
        /* specifies the masked DateTimePicker component without
mask placeholder */
        <DateTimePickerComponent enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
        <br />
        <br />
        /* specifies the masked DateTimePicker component with mask
placeholder */
        <DateTimePickerComponent enableMask={true}
maskPlaceholder={this.maskPlaceholderValue}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
      </div>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

[Functional-component]

INDEX.JSX

```

import { DateTimePickerComponent, Inject, MaskedDateTime } from
'@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
  let maskPlaceholderValue;
  maskPlaceholderValue = { day: 'd', month: 'M', year: 'y', hour: 'h',
minute: 'm', second: 's' };
  return (<div>
    /* specifies the masked DateTimePicker component without
mask placeholder */
    <DateTimePickerComponent enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
    <br />
    <br />
    /* specifies the masked DateTimePicker component with mask
placeholder */
    <DateTimePickerComponent enableMask={true}
maskPlaceholder={maskPlaceholderValue}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
  </div>);
}
ReactDOM.render(<App />, document.getElementById('element'));

```


INDEX.TSX

```
import { DateTimePickerComponent, Inject, MaskedDateTime } from
 '@syncfusion/ej2-react-calendars';
import * as ReactDOM from 'react-dom';
import * as React from 'react';
function App() {
    let maskPlaceholderValue: object;
    maskPlaceholderValue = {day: 'd', month: 'M', year: 'y', hour: 'h',
minute: 'm', second: 's'};
    return (
        <div>
            /* specifies the masked DateTimePicker component without
mask placeholder */
            <DateTimePickerComponent enableMask={true}><Inject
services=[MaskedDateTime] /></DateTimePickerComponent>
            <br />
            <br />
            /* specifies the masked DateTimePicker component with mask
placeholder */
            <DateTimePickerComponent enableMask={true}
maskPlaceholder={maskPlaceholderValue}><Inject services=[MaskedDateTime]
/></DateTimePickerComponent>
        </div>
    );
}
ReactDOM.render(<App />, document.getElementById('element'));
```

Customization in React Datetimepicker component

The DateTimePicker is available for UI customization that can be achieved by using available properties and events in the component.

Day and Time Cell format

The DateTimePicker is available for UI customization based on your application requirements. It can be achieved by using [renderDayCell](#) event that provides an option to customize each day cell on rendering.

The following example disables the weekends of every month by using `renderDayCell` event.

[Class-component]**INDEX.JSX**

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    onRenderCell(args) {
        if (args.date.getDay() === 0 || args.date.getDay() === 6) {
            // sets isDisabled to true to disable the date.
            args.isDisabled = true;
        }
    }
    render() {
        return <DateTimePickerComponent id="datetimepicker"
renderDayCell={this.onRenderCell} placeholder="Select a date and time"/>;
    }
}
```

```

}
;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { DateTimePickerComponent, RenderDayCellEventArgs } from
 '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    public onRenderCell(args: RenderDayCellEventArgs): void {
        if ((args.date as Date).getDay() === 0 || (args.date as
 Date).getDay() === 6) {
            // sets isDisabled to true to disable the date.
            args.isDisabled = true;
        }
    }
    public render() {
        return <DateTimePickerComponent id="datetimepicker"
 renderDayCell={this.onRenderCell} placeholder="Select a date and time"/>
    }
};
ReactDOM.render(<App />, document.getElementById('element'));

```

[Functional-component]**INDEX.JSX**

```

import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    function onRenderCell(args) {
        if (args.date.getDay() === 0 || args.date.getDay() === 6) {
            // sets isDisabled to true to disable the date.
            args.isDisabled = true;
        }
    }
    return <DateTimePickerComponent id="datetimepicker"
 renderDayCell={onRenderCell} placeholder="Select a date and time"/>;
}
;
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { DateTimePickerComponent, RenderDayCellEventArgs } from
 '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    function onRenderCell(args: RenderDayCellEventArgs): void {
        if ((args.date as Date).getDay() === 0 || (args.date as
 Date).getDay() === 6) {

```

```

        // sets isDisabled to true to disable the date.
        args.isDisabled = true;
    }
}
return <DateTimePickerComponent id="datetimepicker"
renderDayCell={onRenderCell} placeholder="Select a date and time"/>
};
ReactDOM.render(<App />, document.getElementById('element'));

```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

[Class-component]

INDEX.JSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    render() {
        return <DateTimePickerComponent id="datetimepicker"
floatLabelType="Auto" placeholder="Select a date and time"/>;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    public render() {
        return <DateTimePickerComponent id="datetimepicker"
floatLabelType="Auto" placeholder="Select a date and time"/>;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

[Functional-component]

INDEX.JSX

```

// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <DateTimePickerComponent id="datetimepicker"
floatLabelType="Auto" placeholder="Select a date and time"/>;
}

```

```
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    return <DateTimePickerComponent id="datetimepicker"
    floatLabelType="Auto" placeholder="Select a date and time"/>;
}
ReactDOM.render(<App />, document.getElementById('element'));
```

See Also

- [How to disable the DateTimePicker component](#)
- [How to customize the DateTimePicker day header](#)

Accessibility in React Datetimepicker component

The DateTimePicker component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DateTimePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Web accessibility defines a way to make web content and web applications more accessible to disabled people. It especially helps the dynamic content change and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

DateTimePicker provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA supports is achieved through the attributes like `aria-expanded`, `aria-disabled`, `aria-activedescendant` applied to the input element.

To know about the accessibility of Calendar refer to the Calendar's [Accessibility](#) section.

It helps to provide information about the widget for assistive technology to the disabled person in screen reader.

- **Aria-expanded:** attributes indicates the state of a collapsible element.
- **Aria-disabled:** attribute indicates the disabled state of this DateTimePicker component.
- **Aria-activedescendent:** attribute helps in managing the current active child of the DateTimePicker component.

Keyboard Interaction

You can use the following keys to interact with the DateTimePicker. The component implements the keyboard navigation support by following the [WAI-ARIA practices](#).

DateTimePicker supports the below list of shortcut keys.

Input Navigation

Before opening the popup, use the below list of keys to `DateTimePicker` control the popup element.

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

| --- | --- |

| **Alt + Down Arrow** | **Open the select popup** |

| Alt + Down Arrow + Alt + Down Arrow | Toggle between two popup |

Calendar Navigation

Use the below list of keys to interact with the Calendar after the DatePicker popup has opened.

| Press | To do this |

| --- | --- |

| Upper Arrow | Focus the previous week date. |

| Down Arrow | Focus the next week date. |

| Left Arrow | Focus the previous date. |

| Right Arrow | Focus the next date. |

| Home | Focus the first date in the month. |

| End | Focus the last date in the month. |

| Page Up | Focus the same date in the previous month. |

| Page Down | Focus the same date in the next month. |

| Enter | Select the currently focused date. |

| Shift + Page Up | Focus the same date in the previous year. |

| Shift + Page Down | Focus the same date in the previous year. |

| Control + Upper Arrow | Moves into the inner level of view like month-year, year-decade |

| Control + Down Arrow | Moves out from the depth level view like decade-year, year-month |

| Control + Home | Focus the starting date in the current year. |

| Control + End | Focus the ending date in the current year. |

Use the below list of shortcut keys to interact with the TimePicker after the TimePicker Popup has opened.

| Press | To do this |

| --- | --- |

| Upper Arrow | Navigate and select the previous item. |

| Down Arrow | Navigate and select the next item. |

| Left Arrow | Move the cursor towards arrow key pressed direction. |

| Right Arrow | Move the cursor towards arrow key pressed direction. |

| Home | Navigate and select the first item. |

| End | Navigate and select the last item. |

| Enter | Select the currently focused item and close the popup. |

| Alt + Upper Arrow | Close the popup. |

| Alt + Down Arrow | Open the popup. |

| Esc | Close the popup. |

To focus the DateTimePicker component use the **alt+t** keys.

[Class-component]

INDEX.JSX

```
{% raw %}
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  datetimepickerInstance;
  componentDidMount() {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.datetimepickerInstance.focusIn();
      }
    };
  }
  render() {
    return <DateTimePickerComponent id="datetimepicker"
placeholder="Select a date and time" ref={ (scope) => {
this.datetimepickerInstance = scope; } }/>;
  }
};
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
{% raw %}
import { DateTimePicker, DateTimePickerComponent } from '@syncfusion/ej2-
react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private datetimepickerInstance: DateTimePicker;
  public componentDidMount() {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.datetimepickerInstance.focusIn();
      }
    };
  }
  public render() {
    return <DateTimePickerComponent id="datetimepicker"
placeholder="Select a date and time" ref={ (scope) => {
(this.datetimepickerInstance as DateTimePicker | null) = scope; } }/>;
  }
};
```

```

    }
  };
  ReactDOM.render(<App />, document.getElementById('element'));
  {% endraw %}

```

[Functional-component]

INDEX.JSX

```

{% raw %}
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let datetimepickerInstance;
  React.useEffect(() => {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.datetimepickerInstance.focusIn();
      }
    };
  }, []);
  return <DateTimePickerComponent id="datetimepicker" placeholder="Select
a date and time" ref={(scope) => { datetimepickerInstance = scope; }}/>;
}
;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DateTimePicker, DateTimePickerComponent } from '@syncfusion/ej2-
react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let datetimepickerInstance: DateTimePicker;
  React.useEffect(() => {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.datetimepickerInstance.focusIn();
      }
    };
  }, []);

  return <DateTimePickerComponent id="datetimepicker" placeholder="Select
a date and time" ref={(scope) => { (datetimepickerInstance as DateTimePicker
| null) = scope; }}/>;
};
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```


Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Style appearance in React Datetimepicker 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 DateTimePicker 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 {
font-size: 20px;
height: 40px;
}
`
```

Customizing the DateTimePicker icons element

Use the following CSS to customize the DateTimePicker icons element

```
`css
/ To specify background color and font size /
.e-datetime-wrapper .e-input-group-icon.e-date-icon, .e-datetime-wrapper .e-input-group-icon.e-time-
icon {
font-size: 16px;
background-color: blanchedalmond;
}
`
```

Customizing the time picker popup in the DateTimePicker

Use the following CSS to customize the time picker popup in the DateTimePicker

```
`css
/ To specify height /
```

```
.e-datetimepicker.e-popup {  
height: 100px;  
}  
`
```

Customizing the Calendar popup of the DateTimePicker

Please check the below section, to customize the style and appearance of the Calendar component in the DateTimePicker

[Customizing Calendar's style and appearance](#)



Full screen mode support in mobiles and tablets

The DateTimePicker component's full-screen mode feature enables users to view the component popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the DateTimePicker component, simply set the [fullScreenMode](#) API value to `true`. This action will extend the calendar and time popup element to occupy the entire screen on mobile devices.

```
`typescript  
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';  
import * as React from 'react';  
import * as ReactDOM from 'react-dom';  
export default class App extends React.Component<{}, {}> {  
  private mobileMode:boolean = true;  
  public render() {  
    return <DateTimePickerComponent id="datetimepicker" fullScreenMode={this.mobileMode}/>;  
  }  
}  
ReactDOM.render(<App />, document.getElementById('element'));
```

Default Sample

12/15/2017 2:00 PM



How To

Disable the datetimepicker component in React Datetimepicker component

To disable the DateTimePicker, use its [enable](#) property to **false**.

INDEX.JSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  enable = false;
  render() {
    return <DateTimePickerComponent id="datetimepicker"
    enabled={this.enable} placeholder='Select a date and time' />;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

INDEX.TSX

```
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private enable:boolean=false;
  public render() {
    return <DateTimePickerComponent id="datetimepicker"
    enabled={this.enable} placeholder='Select a date and time' />;
  }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

Customize the datetimepicker day header in React Datetimepicker component

You can change the format of the day that to be displayed in header using [dayHeaderFormat](#) property. By default, the format is **Short**.

You can find the possible formats on below.

Name	Description
----- -----	
Short	Sets the short format of day name (like Su) in day header.
Narrow	Sets the single character of day name (like S) in day header.
Abbreviated	Sets the min format of day name (like Sun) in day header.
Wide	Sets the long format of day name (like Sunday) in day header.

INDEX.JSX

```
{% raw %}
// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component {
  datetimepickerObj;
  floatLabelObj;
  floatData;
  fields;
  value = 'Short';
  constructor(props) {
    super(props);
    this.floatData = [
      { Id: 'Short', Label: 'Short' },
      { Id: 'Narrow', Label: 'Narrow' },
      { Id: 'Abbreviated', Label: 'Abbreviated' },
      { Id: 'Wide', Label: 'Wide' }
    ];
    this.fields = { text: 'Label', value: 'Id' };
  }
  formatHandler(args) {
    this.datetimepickerObj.dayHeaderFormat = args.value;
  }
  render() {
    return (
      <div id='container'>
        <div id='datetimepicker'>
          <DateTimePickerComponent ref={(datetimepicker) =>
            this.datetimepickerObj = datetimepicker} dayHeaderFormat="Short"/>
        </div>
        <div id="format">
          <label className="custom-input-label">Header Format
            Types</label>
          <DropDownListComponent id="select" value={this.value}
            dataSource={this.floatData} ref={(dropdownlist) => { this.floatLabelObj =
              dropdownlist; }} fields={this.fields}
            change={this.formatHandler.bind(this)} />
        </div>
      </div>);
  }
}
;
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
// import the datetimepickercomponent
import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from "react";
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public datetimepickerObj: DateTimePickerComponent;
  public floatLabelObj: DropDownListComponent;
  private floatData: { [key: string]: Object }[];
  private fields: object;
  private value: string = 'Short';

```

```

    constructor(props: {}) {
        super(props);
        this.floatData = [
            { Id: 'Short', Label: 'Short' },
            { Id: 'Narrow', Label: 'Narrow' },
            { Id: 'Abbreviated', Label: 'Abbreviated' },
            { Id: 'Wide', Label: 'Wide' }
        ];
        this.fields = { text: 'Label', value: 'Id' };
    }

    private formatHandler(args: any): void {
        this.datetimepickerObj.dayHeaderFormat = args.value;
    }

    public render(): JSX.Element {
        return (
            <div id='container'>
                <div id='datetimepicker'>
                    <DateTimePickerComponent ref={(datetimepicker) =>
this.datetimepickerObj = datetimepicker} dayHeaderFormat="Short"/>
                </div>
                <div id="format">
                    <label className="custom-input-label">Header Format
Types</label>
                    <DropDownListComponent id="select" value = {this.value}
dataSource={this.floatData} ref={(dropdownlist) => { this.floatLabelObj =
dropdownlist }} fields={this.fields} change={this.formatHandler.bind(this)}
/>
                </div>
            </div>
        );
    }
};
ReactDOM.render(<App />, document.getElementById('element'));
{% endraw %}

```

Set the placeholder in React Datetimepicker component

The following example demonstrates how to set [placeholder](#) in the DateTimePicker component.

Using `placeholder` you can display a short hint in the input element.

INDEX.JSX

```

import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    render() {
        return <DateTimePickerComponent id="datetimepicker"
placeholder='Select a date and time' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));

```

INDEX.TSX

```

import { DateTimePickerComponent } from '@syncfusion/ej2-react-calendars';

```

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    public render() {
        return <DateTimePickerComponent id="datetimepicker"
placeholder='Select a date and time' />;
    }
}
ReactDOM.render(<App />, document.getElementById('element'));
```

Ej1 api migration in React Datetimepicker component

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

DateTime Selection

{% raw %}

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Setting the value	Property: value	Property: value</i>

DateTime Format

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Display the datetime format	Property: dateTimeFormat	Property: format
Day header format	Property: dayHeaderFormat	Not Applicable

Calendar Views

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Start view	Property: startLevel	Property: start
Depth view	Property: depthLevel	Property: depth

Date Range

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Minimum datetime	Property: minDateTime	Property: min
Maximum datetime	Property: maxDateTime	Property: max

Disabled Dates

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Disabled dates	Not Applicable	Can be achieved by <code>disabledDatetime(args) { /Date need to be disabled/ if (args.date.getDay() === 0 args.date.getDay() === 6) { args.isDisabled = true; }}</code>

Customization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
CSS Class	Property: cssClass	Property: cssClass
Show/Hide the today button	Can be achieved by.e-datetime-popup.e-popup.e-custom-class .e-button-container { display: none !important;}	Property: showTodayButton
Show/Hide the other month dates	Property: showOtherMonths	Can be achieved by.e-DateTimePicker .e-calendar .e-content tr.e-month-hide, .e-DateTimePicker .e-calendar .e-content td.e-other-month > .e-day { visibility: none;}.e-DateTimePicker .e-calendar .e-content td.e-month-hide, .e-DateTimePicker .e-calendar .e-content td.e-other-month { pointer-events: none; touch-action: none;}
Show/Hide the popup button	Property: showPopupButton	Event: focusonFocus(args) { this.show();}.e-control-wrapper .e-input-group-icon.e-date-icon { display: none;}
Enable/Disable the rounded corner	Property: showRoundedCorner	Can be achieved by.e-control-wrapper.e-custom-style.e-date-wrapper.e-input-group { border-radius: 4px;}
Skip a month	Property: stepMonths	Can be achieved byonOpen(args) { this.navigateTo('Year', new Date("03/18/2018"));}
Show/Hide the tooltip	Property: showTooltip	Not Applicable
Interval	Property: interval	Property: step
Button text	Property: buttonTextvar buttonText = { done: "Ok" }	Can be achieved byL10n.load({ 'en': { 'datetimepicker': { today: 'Now' } } });
Enable/Disable the animation	Property: enableAnimation	Not Applicable
FocusIn method	Not Applicable	Method: focusIn()create(args){ this.focusIn();}

FocusOut method	Not Applicable	Method: focusOut()create(args){ this.focusIn(); this.focusOut();}
Prevent popup close	Not Applicable	Event: CloseonClose(args) { <i>/Triggers when the popup gets close/</i> args.cancel = true;}
Prevent popup open	Not Applicable	Event: OpenonOpen(args) { <i>/Triggers when the popup gets close/</i> 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 <pre>var validationRules = {required: {true}};\$.validator.setDefaults({errorClass: 'e-validation-error', errorPlacement: function (error, element) {\$(error).insertAfter(element.closest(".e-widget")); } });</pre>	Can be achieved by <pre>var options = { rules: { 'datetimepicker': { required: true } } };var formObject = new ej.inputs.FormValidator('#form-element', options);</pre>
Validation message	Property: validationMessage <pre>var validationRules = {required: {true}};var validationMessage = {required: "Required value"};\$.validator.setDefaults({errorClass: 'e-validation-error', errorPlacement: function (error, element) {\$(error).insertAfter(element.closest(".e-widget")); } });</pre>	Can be achieved by <pre>var options = { rules: { 'datetimepicker': { required: true } }, customPlacement: (inputElement, errorElement) => { inputElement.parentElement.parentElement.appendChild(errorElement); } };var formObject = new ej.inputs.FormValidator('#form-element', options);`</pre>

Common

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

Behavior	API in Essential JS 1	API in Essential JS 2
Width	Property: width	Property: Width
Readonly	Property: readOnly	Property: readonly
Show/Hide the clear button	Not Applicable	Property: showClearButton
Height	Property: height	Can be achieved by.e-control-wrapper.e-custom-style.e-date-wrapper.e-input-group { height: 35px;}
Html Attributes	Property: htmlAttributesvar htmlAttributes = {required:"required"}	Not Applicable
Show/Hide the week number	Property: weekNumber	Property: weekNumber
Watermark text	Property: watermarkText	Property: placeholder
Disable/Enable	Property: enabled	Property: enabled
Enable/Disable the textbox editing	Property: AllowEdit	Property: AllowEdit
zIndex	Can be achieved by.e-datetime-popup.e-popup.e-custom-class { z-index: 100 !important;}	Property: zIndex
Specify the placeholder text behavior	Not Applicable	Property: floatLabelType
Event callback for each cell creation	Not Applicable	Event: renderDayCellonRenderCell() {/ code block */ }
FocusIn event	Event: FocusInfunction onFocus() { /Triggers when the popup gets focus/ }	Event: focusonFocus() {/ code block */}
FocusOut event	Event: focusOutfunction onFocusout() { /Triggers when the popup gets focusout/ }	Event: bluronBlur() {/ code block */}
Change event	Event: changefunction onChange() { /Triggers when the value is changed/ }	Event: changeonChange() { console.log("changed") }

Created event	Event: createfunction onCreate() { /Triggers when the control is created/ }	Event: createdonCreate() { console.log("Component Created") }
Destroy event	Event: Destroyfunction onDestroy() { /Triggers when the control is destroyed/ }	Event: destroyedonDestroyed(args) { console.log("destroyed")}onChange(args){ this.destroy();}

Globalization

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Locale	Property: locale	Property: locale
First day of week	Property: startDay	Property: firstDayOfWeek

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: Closefunction onClose() { /Triggers when the poupup gets closed/ }	Event: closeonClose() { /Triggers when the poupup gets closed/ }
Hide	Method: hide()onCreate(args) { var datetimeObject = \$("#datetimepicker").data("ejDateTimePicker"); datetimeObject.show(); datetimeObject.hide();}	Method: hide()onCreate(args){ this.show(); this.hide();}
Open	Event: openfunction onOpen(args){/ code block */}	Event: openonOpen(args){/ code block */}
Show	Method: show()function onCreate(args) { var datetimeObject = \$("#datetimepicker").data("ejDateTimePicker"); datetimeObject.show();}	Method: show()onCreate(args){ this.show();}

[View Navigation](#)

<!-- markdownlint-disable MD033 -->

Behavior	API in Essential JS 1	API in Essential JS 2
Navigate to specific month	Not Applicable	Method: navigateTo()onOpen(args) { this.navigateTo('Year', new Date("03/18/2018"));} }
Navigation callback	Not Applicable	Event: navigatedonNavigated() {/ code block */ }
Enable/Disable the drill down	Property: timeDrillDownvar timeDrillDown = { showMeridian: true , interval: 10 , enabled: true }	Not Applicable

{% endraw %}

[Diagram](#)[Getting Started](#)[Getting started](#)

This section explains the steps required to create a simple diagram and demonstrates the basic usage of the diagram control.

[Dependencies](#)

The following list of dependencies are required to use the **Diagram** component in your application.

```
`javascript
|-- @syncfusion/ej2-react-diagrams
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-diagrams
|-- @syncfusion/ej2-react-base
```

Installation and configuration

You can use [create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
npm install -g create-react-app
```

- To setup basic `React` sample use following commands.

```
create-react-app quickstart --scripts-version=react-scripts-ts
```

```
cd quickstart
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [Node Package Manager](#) public registry. You can choose the component that you want to install. For this application, we are going to use `Diagram` component.

To install `Diagram` component, use the following command

```
`bash
```

```
npm install @syncfusion/ej2-react-diagrams --save
```

Adding Style sheet to the Application

Add `Diagram` component's styles as given below in `App.css`.

```
`css
```

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

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

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

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

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

Adding Diagram component to the Application

- To include the `Diagram` component in application import the `DiagramComponent` from `ej2-react-diagrams` package.
- Then add the `Diagram` component as shown in below code example.

```
[src/index.tsx]
```

```
`ts
import * as React from "react";
import "./App.css";
// import the DiagramComponent
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
  return (
    <DiagramComponent id="container" />
  );
}
```

Run the application

Now run the `npm start` command in the console, it will run your application and open the browser window.

```
npm start
```

The below examples shows the basic Diagram component.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
  return (<DiagramComponent id="container" width={"100%"}
    height={"350px"} />);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
  return (
    <DiagramComponent id="container" width={"100%"} height={"350px"} />
  );
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
```


Module Injection

The diagram component is divided into individual feature-wise modules. In order to use a particular feature, inject the required module. The following list describes the module names and their description.

- **BpmnDiagrams** - Inject this provider to add built-in BPMN Shapes to diagrams.
- **ConnectorBridging** - Inject this provider to add bridges to connectors.
- **ConnectorEditing** - Inject this provider to edit the segments for connector.
- **ComplexHierarchicalTree** - Inject this provider to complex hierarchical tree like structure.
- **DataBinding** - Inject this provider to populate nodes from given data source.
- **DiagramContextMenu** - Inject this provider to manipulate context menu.
- **HierarchicalTree** - Inject this provider to use hierarchical tree like structure.
- **LayoutAnimation** - Inject this provider animation to layouts.
- **MindMap** - Inject this provider to use mind map.
- **PrintAndExport** - Inject this provider to print or export the objects.
- **RadialTree** - Inject this provider to use Radial tree like structure.
- **Snapping** - Inject this provider to Snap the objects.
- **SymmetricLayout** - Inject this provider to render layout in symmetrical method.
- **UndoRedo** - Inject this provider to revert and restore the changes.

These modules should be imported and injected into the Diagram component using **Diagram.Inject** method as follows.

```
`javascript
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  DiagramComponent,
  HierarchicalTree,
  MindMap,
  RadialTree,
  ComplexHierarchicalTree,
  DataBinding,
  Snapping,
  PrintAndExport,
  BpmnDiagrams,
  SymmetricLayout,
  ConnectorBridging,
  UndoRedo,
  LayoutAnimation,
```

```

DiagramContextMenu,
ConnectorEditing
} from "@syncfusion/ej2-react-diagrams";
export default function App() {
  return (
    <DiagramComponent id="container">
      <Inject
        services={[
          HierarchicalTree,
          MindMap,
          RadialTree,
          ComplexHierarchicalTree,
          DataBinding,
          Snapping,
          PrintAndExport,
          BpmnDiagrams,
          SymmetricLayout,
          ConnectorBridging,
          UndoRedo,
          LayoutAnimation,
          DiagramContextMenu,
          ConnectorEditing
        ]}
      />
    </DiagramComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

```

Flow Diagram

Create and Add Node

Create and add a **node** (JSON data) with specific position, size, label, and shape.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
    const nodes = [
        {
            id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 100,
            annotations: [{
                id: 'label1',
                content: 'Start'
            }],
            shape: { type: 'Flow', shape: 'Terminator' }
        }
    ];
    return (<DiagramComponent id="container" width={"100%"} height={"350px"}
nodes={nodes}/>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, NodeModel } from "@syncfusion/ej2-react-diagrams";
export default function App() {
    const nodes: NodeModel[] = [
        {
            id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 100,
            annotations: [{
                id: 'label1',
                content: 'Start'
            }],
            shape: { type: 'Flow', shape: 'Terminator' }
        }
    ];
    return (
        <DiagramComponent
            id="container"
            width={"100%"}
            height={"350px"}
            nodes={nodes}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
```

Connect two Nodes with a Connector

Add two node to the diagram as shown in the previous example. Connect these nodes by adding a connector using the `connector` property and refer the source and target end by using the `sourceNode` and `targetNode` properties.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
  const nodes = [
    {
      id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 50,
      annotations: [{
        id: 'labell1',
        content: 'Start'
      }],
      shape: { type: 'Flow', shape: 'Terminator' }
    },
    {
      id: 'Init', width: 140, height: 50, offsetX: 300, offsetY: 140,
      shape: { type: 'Flow', shape: 'Process' },
      annotations: [{ content: 'var i = 0;' }]
    }
  ];
  const connectors = [
    {
      id: "connector1",
      sourceID: "Start",
      targetID: "Init"
    }
  ];
  return (<DiagramComponent id="container" width={"100%"} height={"350px"}
    nodes={nodes} connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  DiagramComponent,
  NodeModel,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
export default function App() {
  const nodes: NodeModel[] = [
    {
      id: 'Start', width: 140, height: 50, offsetX: 300, offsetY: 50,
      annotations: [{
        id: 'labell1',
        content: 'Start'
      }],
      shape: { type: 'Flow', shape: 'Terminator' }
    },
    {
      id: 'Init', width: 140, height: 50, offsetX: 300, offsetY: 140,
      shape: { type: 'Flow', shape: 'Process' },
      annotations: [{ content: 'var i = 0;' }]
    }
  ];

```

```

];
const connectors: ConnectorModel[] = [
  {
    id: "connector1",
    sourceID: "Start",
    targetID: "Init"
  }
];
return (
  <DiagramComponent
    id="container"
    width={"100%"}
    height={"350px"}
    nodes={nodes}
    connectors={connectors}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);

```

Default values for all `nodes` and `connectors` can be set using the `getNodeDefaults` and `getConnectorDefaults` properties, respectively. For example, if all nodes have the same width and height, such properties can be moved into `getNodeDefaults`.

[Complete Flow Diagram](#)

Similarly we can add required nodes and connectors to form a complete flow diagram.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
export default function App() {
  const nodes = [
    {
      id: "node1",
      offsetY: 50,
      shape: { type: "Flow", shape: "Terminator" },
      annotations: [
        {
          content: "Start"
        }
      ]
    },
    {
      id: "node2",
      offsetY: 140,
      shape: { type: "Flow", shape: "Process" },
      annotations: [
        {
          content: "var i = 0;"
        }
      ]
    }
  ],

```

```

    {
      id: "node3",
      offsetY: 230,
      shape: { type: "Flow", shape: "Decision" },
      annotations: [
        {
          content: "i < 10?"
        }
      ]
    },
    {
      id: "node4",
      offsetY: 320,
      shape: { type: "Flow", shape: "PreDefinedProcess" },
      annotations: [
        {
          content: 'print("Hello!!");'
        }
      ]
    },
    {
      id: "node5",
      offsetY: 410,
      shape: { type: "Flow", shape: "Process" },
      annotations: [
        {
          content: "i++;"
        }
      ]
    },
    {
      id: "node6",
      offsetY: 500,
      shape: { type: "Flow", shape: "Terminator" },
      annotations: [
        {
          content: "End"
        }
      ]
    }
  ];
  const connectors = [
    {
      id: "connector1",
      sourceID: "node1",
      targetID: "node2"
    },
    {
      id: "connector2",
      sourceID: "node2",
      targetID: "node3"
    },
    {
      id: "connector3",
      sourceID: "node3",
      targetID: "node4",
      annotations: [{ content: "Yes" }]
    }
  ]

```

```

    },
    {
      id: "connector4",
      sourceID: "node3",
      targetID: "node6",
      annotations: [{ content: "No" }],
      type: 'Orthogonal',
      segments: [
        { type: 'Orthogonal', length: 50, direction: "Right" },
        { type: 'Orthogonal', length: 300, direction: "Bottom" }
      ]
    },
    {
      id: "connector5",
      sourceID: "node4",
      targetID: "node5"
    },
    {
      id: "connector6",
      sourceID: "node5",
      targetID: "node3",
      type: 'Orthogonal',
      segments: [
        { length: 50, type: 'Orthogonal', direction: "Left" },
        { length: 200, type: 'Orthogonal', direction: "Top" }
      ]
    }
  ];
  return (<DiagramComponent id="container" width={"100%"} height={"600px"}
nodes={nodes} connectors={connectors} getNodeDefaults={(node) => {
  node.height = 50;
  node.width = 140;
  node.offsetX = 300;
  return node;
}} getConnectorDefaults={(obj) => {
  obj.type = "Orthogonal";
  obj.targetDecorator = { shape: 'Arrow', width: 10, height: 10 };
  return obj;
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  DiagramComponent,
  NodeModel,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
export default function App() {
  const nodes: NodeModel[] = [

```

```
{
  id: "node1",
  offsetY: 50,
  shape: { type: "Flow", shape: "Terminator" },
  annotations: [
    {
      content: "Start"
    }
  ]
},
{
  id: "node2",
  offsetY: 140,
  shape: { type: "Flow", shape: "Process" },
  annotations: [
    {
      content: "var i = 0;"
    }
  ]
},
{
  id: "node3",
  offsetY: 230,
  shape: { type: "Flow", shape: "Decision" },
  annotations: [
    {
      content: "i < 10?"
    }
  ]
},
{
  id: "node4",
  offsetY: 320,
  shape: { type: "Flow", shape: "PreDefinedProcess" },
  annotations: [
    {
      content: 'print("Hello!!");'
    }
  ]
},
{
  id: "node5",
  offsetY: 410,
  shape: { type: "Flow", shape: "Process" },
  annotations: [
    {
      content: "i++;"
    }
  ]
},
{
  id: "node6",
  offsetY: 500,
  shape: { type: "Flow", shape: "Terminator" },
  annotations: [
    {
      content: "End"
    }
  ]
}
```



```

    }
  ]
}
];
const connectors: ConnectorModel[] = [
  {
    id: "connector1",
    sourceID: "node1",
    targetID: "node2"
  },
  {
    id: "connector2",
    sourceID: "node2",
    targetID: "node3"
  },
  {
    id: "connector3",
    sourceID: "node3",
    targetID: "node4",
    annotations: [{ content: "Yes" }]
  },
  {
    id: "connector4",
    sourceID: "node3",
    targetID: "node6",
    annotations: [{ content: "No" }],
    type: 'Orthogonal',
    segments: [
      { type: 'Orthogonal', length: 50, direction: "Right" },
      { type: 'Orthogonal', length: 300, direction: "Bottom" }
    ]
  },
  {
    id: "connector5",
    sourceID: "node4",
    targetID: "node5"
  },
  {
    id: "connector6",
    sourceID: "node5",
    targetID: "node3",
    type: 'Orthogonal',
    segments: [
      { length: 50, type: 'Orthogonal', direction: "Left" },
      { length: 200, type: 'Orthogonal', direction: "Top" }
    ]
  }
];
return (
  <DiagramComponent
    id="container"
    width={"100%"}
    height={"600px"}
    nodes={nodes}
    connectors={connectors}
    getNodeDefaults={(node: NodeModel): NodeModel => {
      node.height = 50;

```

```

        node.width = 140;
        node.offsetX = 300;
        return node;
    }}
    getConnectorDefaults=(obj: ConnectorModel): ConnectorModel => {
        obj.type = "Orthogonal";
        obj.targetDecorator = { shape: 'Arrow', width: 10, height: 10 };
        return obj;
    }}
    />
    );
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Automatic Organization Chart

In 'Flow Diagram' section we saw how to create a diagram manually, now let us see how to create and position diagram automatically.

Business object (Employee information)

Define Employee Information as JSON data. The following code example shows an employee array whose, **Name** is used as an unique identifier and **ReportingPerson** is used to identify the person to whom an employee report to, in the organization.

```

`ts
const data: Object[] = [
{
  Name: "Elizabeth",
  Role: "Director"
},
{
  Name: "Christina",
  ReportingPerson: "Elizabeth",
  Role: "Manager"
},
{
  Name: "Yoshi",
  ReportingPerson: "Christina",
  Role: "Lead"
},
{
  Name: "Philip",

```

```

ReportingPerson: "Christina",
Role: "Lead"
},
{
Name: "Yang",
ReportingPerson: "Elizabeth",
Role: "Manager"
},
{
Name: "Roland",
ReportingPerson: "Yang",
Role: "Lead"
},
{
Name: "Yvonne",
ReportingPerson: "Yang",
Role: "Lead"
}
];
`

```

[Map data source](#)

You can configure the above "Employee Information" with diagram, so that the nodes and connectors are automatically generated using the mapping properties. The following code example show how `dataSourceSettings` is used to map ID and parent with property name identifiers for employee information.

```

`ts
export default function App() {
const data: Object[] = [
{
Name: "Elizabeth",
Role: "Director"
},
{
Name: "Christina",

```

```
ReportingPerson: "Elizabeth",
Role: "Manager"
},
{
Name: "Yoshi",
ReportingPerson: "Christina",
Role: "Lead"
},
{
Name: "Philip",
ReportingPerson: "Christina",
Role: "Lead"
},
{
Name: "Yang",
ReportingPerson: "Elizabeth",
Role: "Manager"
},
{
Name: "Roland",
ReportingPerson: "Yang",
Role: "Lead"
},
{
Name: "Yvonne",
ReportingPerson: "Yang",
Role: "Lead"
}
];
const dataSettings: object = {
id: "Name",
parentId: "ReportingPerson",
dataManager: new DataManager(data as JSON[])
}
```

```

}
return (
  <DiagramComponent
    id="container"
    width={"100%"}
    height={"350px"}
    dataSourceSettings={dataSettings}

    <Inject services={[HierarchicalTree, DataBinding]} />
  </DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

```

Visualize employee

Following code examples indicate how to define the default appearance of node and connector using default settings.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
import { HierarchicalTree, Inject, DataBinding } from "@syncfusion/ej2-react-diagrams";
import { DataManager } from "@syncfusion/ej2-data";
export default function App() {
  const data = [
    {
      Name: "Elizabeth",
      Role: "Director"
    },
    {
      Name: "Christina",
      ReportingPerson: "Elizabeth",
      Role: "Manager"
    },
    {
      Name: "Yoshi",
      ReportingPerson: "Christina",
      Role: "Lead"
    },
    {
      Name: "Philip",

```

```

        ReportingPerson: "Christina",
        Role: "Lead"
    },
    {
        Name: "Yang",
        ReportingPerson: "Elizabeth",
        Role: "Manager"
    },
    {
        Name: "Roland",
        ReportingPerson: "Yang",
        Role: "Lead"
    },
    {
        Name: "Yvonne",
        ReportingPerson: "Yang",
        Role: "Lead"
    }
];
const dataSettings = {
    id: "Name",
    parentId: "ReportingPerson",
    dataManager: new DataManager(data)
};
const layoutSetting = { type: "OrganizationalChart" };
return (<DiagramComponent id="container" width={"100%"} height={"350px"}
dataSourceSettings={dataSettings} layout={layoutSetting}
getNodeDefaults=(node) => {
    let codes = {
        Director: "rgb(0, 139,139)",
        Manager: "rgb(30, 30,113)",
        Lead: "rgb(0, 100,0)"
    };
    node.width = 70;
    node.height = 30;
    node.annotations = [
        { content: node.data.Name, style: { color: "white" } }
    ];
    node.style.fill = codes[node.data.Role];
    return node;
}} getConnectorDefaults=(connector) => {
    connector.type = "Orthogonal";
    connector.cornerRadius = 7;
    return connector;
})>
    <Inject services={[HierarchicalTree, DataBinding]}/>
</DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";

```

```

import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
import {
  Node,
  HierarchicalTree,
  Diagram,
  NodeModel,
  ConnectorModel,
  Inject,
  DataBinding
} from "@syncfusion/ej2-react-diagrams";
import { DataManager } from "@syncfusion/ej2-data";
export interface EmployeeInfo {
  Name: string;
  Role: string;
  color: string;
}
export default function App() {
  const data: Object[] = [
    {
      Name: "Elizabeth",
      Role: "Director"
    },
    {
      Name: "Christina",
      ReportingPerson: "Elizabeth",
      Role: "Manager"
    },
    {
      Name: "Yoshi",
      ReportingPerson: "Christina",
      Role: "Lead"
    },
    {
      Name: "Philip",
      ReportingPerson: "Christina",
      Role: "Lead"
    },
    {
      Name: "Yang",
      ReportingPerson: "Elizabeth",
      Role: "Manager"
    },
    {
      Name: "Roland",
      ReportingPerson: "Yang",
      Role: "Lead"
    },
    {
      Name: "Yvonne",
      ReportingPerson: "Yang",
      Role: "Lead"
    }
  ];
  const dataSettings: object = {
    id: "Name",
    parentId: "ReportingPerson",
  
```

```

    dataManager: new DataManager(data as JSON[])
  }
  const layoutSetting: object= { type: "OrganizationalChart" }
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"350px"}
      dataSourceSettings={dataSettings}
      layout={layoutSetting}
      getNodeDefaults={(node: NodeModel): NodeModel => {
        let codes: Object = {
          Director: "rgb(0, 139,139)",
          Manager: "rgb(30, 30,113)",
          Lead: "rgb(0, 100,0)"
        };
        node.width = 70;
        node.height = 30;
        node.annotations = [
          { content: (node.data as EmployeeInfo).Name, style: { color:
"white" } }
        ];
        node.style.fill = codes[(node.data as EmployeeInfo).Role];
        return node;
      }}
      getConnectorDefaults={(connector: ConnectorModel): ConnectorModel => {
        connector.type = "Orthogonal";
        connector.cornerRadius = 7;
        return connector;
      }}
    >
    <Inject services={[HierarchicalTree, DataBinding]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

You can refer to our [React Diagram](#) feature tour page for its groundbreaking feature representations. You can also explore our [React Diagram example](#) that shows how to render the Diagram in React.

Creating a Next.js Application Using Syncfusion React Components

This section provides a step-by-step guide for setting up a Next.js application and integrating the Syncfusion React Diagram component.

What is Next.js?

[Next.js](#) is a React framework that makes it easy to build fast, SEO-friendly, and user-friendly web applications. It provides features such as server-side rendering, automatic code splitting, routing, and API routes, making it an excellent choice for building modern web applications.

Prerequisites

Before getting started with the Next.js application, ensure the following prerequisites are met:

- [Node.js 16.8](#) or later.

- The application is compatible with macOS, Windows, and Linux operating systems.

Create a Next.js application

To create a new **Next.js** application, use one of the commands that are specific to either NPM or Yarn.

NPM

```
npx create-next-app@latest
```

YARN

```
yarn create next-app
```

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: Users can specify the name of the project directly. Let's specify the name of the project as **ej2-nextjs-diagram**.

CMD

```
√ What is your project named? » ej2-nextjs-diagram
```

2. Select the required packages.

CMD

```
√ What is your project named? ... ej2-nextjs-diagram
√ Would you like to use TypeScript? ... No / `Yes`
√ Would you like to use ESLint? ... No / `Yes`
√ Would you like to use Tailwind CSS? ... `No` / Yes
√ Would you like to use `src/` directory? ... No / `Yes`
√ Would you like to use App Router? (recommended) ... No / `Yes`
√ Would you like to customize the default import alias? ... `No` / Yes
Creating a new Next.js app in D:\ej2-nextjs-diagram.
```

3. Once complete the above mentioned steps to create **ej2-nextjs-diagram**, navigate to the directory using the below command:

CMD

```
cd ej2-nextjs-diagram
```

The application is ready to run with default settings. Now, let's add Syncfusion components to the project.

Install Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

Here, the [React Diagram component](#) is used in the project. To install the React Diagram component, use the following command:

NPM

```
npm install @syncfusion/ej2-react-diagrams --save
```

YARN

```
yarn add @syncfusion/ej2-react-diagrams
```

Import Syncfusion CSS styles

Syncfusion React components come with [built-in themes](#), which are available in the installed packages. It's easy to adapt the Syncfusion React components to match the style of your application by referring to one of the built-in themes.

Import the **Material** theme into the **src/app/globals.css** file and removed the existing styles in that file, as shown below:

GLOBALS.CSS

```
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-diagrams/styles/material.css";
```

To know more about built-in themes and CSS reference for individual components, refer to the [themes](#) section.

Add Syncfusion React component

Follow the below steps to add the React Diagram component to the Next.js project:

1. Before adding the Diagram component to your markup, create a **datasource.tsx** file within the **src/app/** folder and add the Diagram component data.

DATASOURCE.TSX

```
export let data: object[] = [{
  'Id': 'parent',
  'Name': 'Maria Anders',
  'Designation': 'Managing Director',
  'IsExpand': 'true',
  'RatingColor': '#C34444'
},
{
  'Id': 1,
  'Name': 'Ana Trujillo',
  'Designation': 'Project Manager',
  'IsExpand': 'false',
  'RatingColor': '#68C2DE',
  'ReportingPerson': 'parent'
},
{
  'Id': 2,
  'Name': 'Anto Moreno',
  'Designation': 'Project Lead',
```

```
'IsExpand': 'false',
'RatingColor': '#93B85A',
'ReportingPerson': 'parent'
},
{
  'Id': 3,
  'Name': 'Thomas Hardy',
  'Designation': 'Senior S/w Engg',
  'IsExpand': 'false',
  'RatingColor': '#68C2DE',
  'ReportingPerson': 1
},
{
  'Id': 4,
  'Name': 'Christina kaff',
  'Designation': 'S/w Engg',
  'IsExpand': 'false',
  'RatingColor': '#93B85A',
  'ReportingPerson': 2
},
{
  'Id': 5,
  'Name': 'Hanna Moos',
  'Designation': 'Project Trainee',
  'IsExpand': 'true',
  'RatingColor': '#D46E89',
  'ReportingPerson': 2
}
]];
```

2.Then, import and define the Diagram component in the **src/app/page.tsx** file, as shown below:

{% raw %}

NPM

```
npm run dev
```

YARN

```
yarn run dev
```

To learn more about the functionality of the Diagram component, refer to the [documentation](#).

[View the NEXT.js Diagram sample in the GitHub repository.](#)

Getting Started with the Diagram Component in the Preact Framework

This article provides a step-by-step guide for setting up a [Preact](#) project and integrating the Syncfusion React Diagram component.

Preact is a fast and lightweight JavaScript library for building user interfaces. It's often used as an alternative to larger frameworks like React. The key difference is that Preact is designed to be smaller in size and faster in performance, making it a good choice for projects where file size and load times are critical factors.

Prerequisites

[System requirements for Syncfusion React UI components](#)

Set up the Preact project

To create a new **Preact** project, use one of the commands that are specific to either NPM or Yarn.

```
`bash
```

```
npm init preact
```

```
,
```

or

```
`bash
```

```
yarn init preact
```

```
,
```

Using one of the above commands will lead you to set up additional configurations for the project, as below:

1\ Define the project name: We can specify the name of the project directly. Let's specify the name of the project as **my-project** for this article.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project directory:

```
| my-project
```

```
—
```

```
,
```

2\ Choose **JavaScript** as the framework variant to build this Preact project using JavaScript and React.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project language:

```
| > JavaScript
```

```
| TypeScript
```

```
—
```

```
,
```

3\ Then configure the project as below for this article.

```
`bash
```

T Preact - Fast 3kB alternative to React with the same modern API

|

- Use router?

| Yes / > No

—

|

- Prerender app (SSG)?

| Yes / > No

—

|

- Use ESLint?

| Yes / > No

—

,

5\ Upon completing the aforementioned steps to create `my-project`, run the following command to jump into the project directory:

```
`bash
```

```
cd my-project
```

,

Now that `my-project` is ready to run with default settings, let's add Syncfusion components to the project.

Add the Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

This article uses the [React Diagram component](#) as an example. To use the React Diagram component in the project, the `@syncfusion/ej2-react-diagrams` package needs to be installed using the following command:

```
`bash
```

```
npm install @syncfusion/ej2-react-diagrams --save
```

,

or

```
`bash
```

```
yarn add @syncfusion/ej2-react-diagrams
```

Import Syncfusion CSS styles

You can import themes for the Syncfusion React component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and [Theme Studio](#). Refer to [themes topic](#) to know more about built-in themes and different ways to refer to theme's in a React project.

In this article, the **Material 3** theme is applied using CSS styles, which are available in installed packages. The necessary **Material 3** CSS styles for the Diagram component and its dependents were imported into the **src/style.css** file.

~/SRC/STYLE.CSS

```
@import "../node_modules/@syncfusion/ej2-base/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-popups/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-splitbuttons/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-navigations/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-react-
diagrams/styles/material3.css";
```

The order of importing CSS styles should be in line with its dependency graph.

Add the Syncfusion React component

Follow the below steps to add the React Diagram component to the Vite project:

1\. Before adding the Diagram component to your markup, create a **datasource.jsx** file within the **src** folder and add the Diagram component data.

~/SRC/DATASOURCE.JSX

```
export let data = [{
  'Id': 'parent',
  'Name': 'Maria Anders',
  'Designation': 'Managing Director',
  'IsExpand': 'true',
  'RatingColor': '#C34444'
},
{
  'Id': 1,
  'Name': 'Ana Trujillo',
  'Designation': 'Project Manager',
  'IsExpand': 'false',
  'RatingColor': '#68C2DE',
  'ReportingPerson': 'parent'
},
{
  'Id': 2,
  'Name': 'Anto Moreno',
  'Designation': 'Project Lead',
  'IsExpand': 'false',
  'RatingColor': '#93B85A',
  'ReportingPerson': 'parent'
},
{
  'Id': 3,
```

```

'Name': 'Thomas Hardy',
'Designation': 'Senior S/w Engg',
'IsExpand': 'false',
'RatingColor': '#68C2DE',
'ReportingPerson': 1
},
{
  'Id': 4,
  'Name': 'Christina kaff',
  'Designation': 'S/w Engg',
  'IsExpand': 'false',
  'RatingColor': '#93B85A',
  'ReportingPerson': 2
},
{
  'Id': 5,
  'Name': 'Hanna Moos',
  'Designation': 'Project Trainee',
  'IsExpand': 'true',
  'RatingColor': '#D46E89',
  'ReportingPerson': 2
}
]]];

```

2\ Then, import and define the Diagram component in the **src/index.jsx** file, as shown below:

~/SRC/INDEX.JSX

```

{% raw %}
import { DataManager, Query } from '@syncfusion/ej2-data';
import { StackPanel, TextElement, DataBinding, HierarchicalTree,
DiagramComponent, Inject } from '@syncfusion/ej2-react-diagrams';
import { data } from './datasource';
import { render } from 'preact';
export default function App() {
  let items = new DataManager(data, new Query().take(7));
  return (
    <>
    <DiagramComponent id="container" height={'450px'} layout={{
      type: 'HierarchicalTree',
      margin: {
        top: 20,
      },
    }} dataSourceSettings={{
      id: 'Id',
      parentId: 'ReportingPerson',
      dataManager: items,
    }} getNodeDefaults=(node) => {
      node.height = 50;
      node.style.fill = '#6BA5D7';
      node.borderColor = 'white';
      node.style.strokeColor = 'white';
      return node;
    }} getConnectorDefaults=(obj) => {
      obj.style.strokeColor = '#6BA5D7';
      obj.style.fill = '#6BA5D7';
      obj.style.strokeWidth = 2;
    }}
    </>
  );
}

```

```

obj.targetDecorator.style.fill = '#6BA5D7';
obj.targetDecorator.style.strokeColor = '#6BA5D7';
obj.targetDecorator.shape = 'None';
obj.type = 'Orthogonal';
return obj;
}} setNodeTemplate=(obj) => {
let content = new StackPanel();
content.id = obj.id + '_outerstack';
content.style.strokeColor = 'darkgreen';
content.style.fill = '#6BA5D7';
content.orientation = 'Horizontal';
content.padding = {
left: 5,
right: 10,
top: 5,
bottom: 5,
};
let innerStack = new StackPanel();
innerStack.style.strokeColor = 'none';
innerStack.style.fill = '#6BA5D7';
innerStack.margin = {
left: 5,
right: 0,
top: 0,
bottom: 0,
};
innerStack.id = obj.id + '_innerstack';
let text = new TextElement();
text.content = obj.data['Name'];
text.style.color = 'white';
text.style.strokeColor = 'none';
text.style.fill = 'none';
text.id = obj.id + '_text1';
let desigText = new TextElement();
desigText.margin = {
left: 0,
right: 0,
top: 5,
bottom: 0,
};
desigText.content = obj.data['Designation'];
desigText.style.color = 'white';
desigText.style.strokeColor = 'none';
desigText.style.fill = 'none';
desigText.style.textWrapping = 'Wrap';
desigText.id = obj.id + '_desig';
innerStack.children = [text, desigText];
content.children = [innerStack];
return content;
}} >
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
</>
)
}
render(<App />, document.querySelector('#app'));
{% enddraw %}

```


Run the project

To run the project, use the following command:

```
`bash
```

```
npm run dev
```

```
,
```

or

```
`bash
```

```
yarn run dev
```

```
,
```

The output will appear as follows:

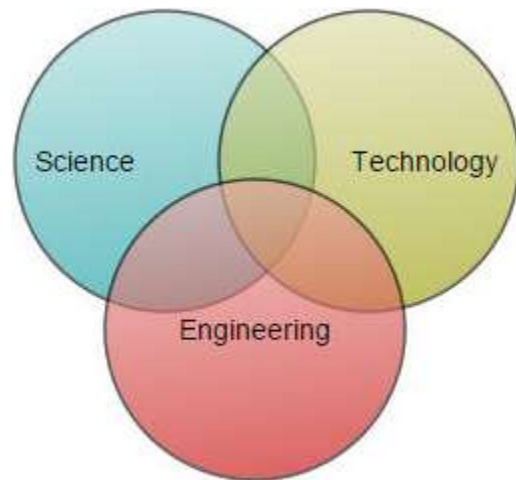


See also

[Getting Started with the Syncfusion React UI Component](#)

Nodes in React Diagram component

Nodes are graphical objects used to visually represent the geometrical information, process flow, internal business procedure, entity, or any other kind of data.



<!-- markdownlint-disable MD033 -->

Create node

A node can be created and added to the diagram, either programmatically or interactively. Nodes are stacked on the diagram area from bottom to top in the order they are added.

Add node through nodes collection

To create a node, define the [node](#) object and add that to nodes collection of the diagram model. The following code example illustrates how to add a node to the diagram.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Text(label) added to the node
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Text(label) added to the node
}];
// initialize Diagram component
```

```
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Add/Remove node at runtime

- Nodes can be added at runtime by using public method, add and can be removed at runtime by using public method,

remove. On adding node at runtime, the nodes collection is changed and the [collectionChange](#) event will trigger.

- The node's ID property is used to define the name of the node and its further used to find the node at runtime and do any customization.

The following code illustrates how to add a node.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = {
  // Size of the node
  width: 100,
  height: 100,
};
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'} created={() =>
    {
      // Add node
      diagramInstance.add(node);
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel = {
  // Size of the node
  width: 100,
  height: 100,
};
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      created={() => {
        // Add node
        diagramInstance.add(node);
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Add node from palette

Nodes can be predefined and added to the palette, and can be dropped into the diagram when needed. For more information

about adding nodes from symbol palette, refer to [Symbol Palette](#).

- Once you drag a node/connector from the palette to the diagram, the following events can be used to do your customization.
- When a symbol is dragged into diagram from symbol palette, the [dragEnter](#) event gets triggered.
- When a symbol is dragged over diagram, the [dragOver](#) event gets triggered.
- When a symbol is dragged and dropped from symbol palette to diagram area, the [drop](#) event gets triggered.
- When a symbol is dragged outside of the diagram, the [dragLeave](#) event gets triggered.

Create node through data source

Nodes can be generated automatically with the information provided through data source. The default properties for

these nodes are fetched from default settings. For more information about data source, refer to Data Binding.

Draw nodes

Nodes can be interactively drawn by clicking and dragging the diagram surface by using `NodeDrawingTool`. For more

information about drawing nodes, refer to Draw Nodes.

Position

- Position of a node is controlled by using its [offsetX](#) and [offsetY](#) properties. By default, these offset properties represent the distance between the origin of the diagram's page and node's center point.
- You may expect this offset values to represent the distance between page origin and node's top-left corner instead of center. The Pivot property helps to solve this problem. Default value of node's [pivot](#) point is (0.5, 0.5), that means center of the node.
- The size of the node can be controlled by using its [width](#) and

[height](#) properties.

- Rotation of a node is controlled by using its [rotateAngle](#) property.

The following table illustrates how pivot relates offset values with node boundaries.

Pivot	Offset
-----	-----
(0.5,0.5)	offsetX and offsetY values are considered as the node's center point.
(0,0)	offsetX and offsetY values are considered as the top-left corner of the node.
(1,1)	offsetX and offsetY values are considered as the bottom-right corner of the node.

The following code illustrates how to change the `pivot` value.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  pivot: {
```

```

        x: 0,
        y: 0
    },
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    }
    // Text(label) added to the node
  }];
let diagramInstance;
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
created={() => {
    diagramInstance.select([diagramInstance.nodes[0]]);
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  pivot: {
    x: 0,
    y: 0
  },
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  }
  // Text(label) added to the node
}];
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}

```

```

    height={'600px'}
    nodes={node}
    created={() => {
      diagramInstance.select([diagramInstance.nodes[0]]);
    }}
    // render initialized Diagram
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Flip

The diagram Provides support to flip the node. [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip

[Horizontal](#) is used to change the element in horizontal direction.

- VerticalFlip

[Vertical](#) is used to change the element in vertical direction

- Both

[Both](#) which involves both vertical and horizontal changes of the element.

The following code illustrates how to provide the mirror image of the original element.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  // Flip the node in Horizontal Direction
  flip: 'Horizontal',
  shape: {
    type: 'Basic',
    shape: 'RightTriangle',
  },
},

```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        }
        // Text(label) added to the node
    }];
let diagramInstance;
function App() {
    return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
    created={() => {
        diagramInstance.select([diagramInstance.nodes[0]]);
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    BasicShapeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    // Flip the node in Horizontal Direction
    flip: 'Horizontal',
    shape: {
        type: 'Basic',
        shape: 'RightTriangle',
    },
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    }
    // Text(label) added to the node
}];
let diagramInstance: DiagramComponent;
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={ (diagram) => (diagramInstance = diagram)}
            width={'100%'}

```



```

        height={'600px'}
        nodes={node}
        created={() => {
            diagramInstance.select([diagramInstance.nodes[0]]);
        }}
        // render initialized Diagram
    />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Note: The flip is also applicable for group and BPMN shapes.

Appearance

- The appearance of a node can be customized by changing its [fill](#) color, [borderColor](#), [borderWidth](#), [strokeDashArray](#),

[opacity](#), and [shadow](#).

- The [visible](#) property of the node enables or disables the visibility of the node.

The following code illustrates how to customize the appearance of the shape.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeDashArray: '5,5'
    },
    borderWidth: 2,
    borderColor: 'red',
    // Text(label) added to the node
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeDashArray: '5,5'
  },
  borderWidth: 2,
  borderColor: 'red',
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Customize the style of main node on multi-selection.

The style of the main node can be customized by using the className `e-diagram-first-selection-indicator`.

Use the following CSS to customize the style of main node on multiple selection.

```
`css
```

```
.e-diagram-first-selection-indicator{
stroke-width: 5px;
stroke: red;
stroke-dasharray: 1,1;
```

```
}  
,
```

Gradient

The [gradient](#) property of the node allows you to define and apply the gradient effect to that node.

The gradient stop property defines the color and a position, where the previous color transition ends and a new color transition starts.

The gradient stop's opacity property defines the transparency level of the region.

There are two types of gradients as follows:

- Linear gradient
- Radial gradient

Linear gradient

- [LinearGradient](#) defines a smooth transition between a set of colors (so-called stops) on a line.
- A linear gradient's x1, y1, x2, y2 properties are used to define the position (relative to the node) of the rectangular region that needs to be painted.

INDEX.JSX

```
import * as React from "react";  
import * as ReactDOM from "react-dom";  
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";  
// A node is created and stored in nodes array.  
let linearGradient;  
linearGradient = {  
  //Start point of linear gradient  
  x1: 0,  
  y1: 0,  
  //End point of linear gradient  
  x2: 50,  
  y2: 50,  
  //Sets an array of stop objects  
  stops: [{  
    color: 'white',  
    offset: 0  
  },  
  {  
    color: '#6BA5D7',  
    offset: 100  
  }  
],  
  type: 'Linear'  
};  
let node = [{  
  // Position of the node  
  offsetX: 250,  
  offsetY: 250,  
  // Size of the node  
  width: 100,
```

```

        height: 100,
        style: {
            gradient: linearGradient
        }
        // Text(label) added to the node
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    GradientModel,
    LinearGradientModel,
    RadialGradientModel,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let linearGradient: GradientModel | LinearGradientModel |
RadialGradientModel;
linearGradient = {
    //Start point of linear gradient
    x1: 0,
    y1: 0,
    //End point of linear gradient
    x2: 50,
    y2: 50,
    //Sets an array of stop objects
    stops: [{
        color: 'white',
        offset: 0
    },
    {
        color: '#6BA5D7',
        offset: 100
    }
    ],
    type: 'Linear'
};
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,

```

```

    style: {
      gradient: linearGradient
    }
    // Text(label) added to the node
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={ '100%' }
        height={ '600px' }
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Radial gradient

- [RadialGradient](#) defines a smooth transition between stops on a circle.
- A radial gradient's cx, cy, fx, fy properties are used to define the position (relative to the node) of the outermost or the innermost circle of the radial gradient.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let radialGradient;
radialGradient = {
  //Center point of outer circle
  cx: 50,
  cy: 50,
  //Center point of inner circle
  fx: 25,
  fy: 25,
  //Radius of a radial gradient
  r: 50,
  //Sets an array of stop objects
  stops: [{
    color: 'white',
    offset: 0
  },
  {
    color: '#6BA5D7',
    offset: 100
  }
  ],
  type: 'Radial'
};
let node = [{
  // Position of the node
  offsetX: 250,

```

```

        offsetY: 250,
        // Size of the node
        width: 100,
        height: 100,
        style: {
            gradient: radialGradient
        }
        // Text(label) added to the node
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    GradientModel,
    LinearGradientModel,
    RadialGradientModel,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
let radialGradient: GradientModel | LinearGradientModel |
RadialGradientModel;
radialGradient = {
    //Center point of outer circle
    cx: 50,
    cy: 50,
    //Center point of inner circle
    fx: 25,
    fy: 25,
    //Radius of a radial gradient
    r: 50,
    //Sets an array of stop objects
    stops: [{
        color: 'white',
        offset: 0
    },
    {
        color: '#6BA5D7',
        offset: 100
    }
    ],
    type: 'Radial'
};
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,

```

```

        offsetY: 250,
        // Size of the node
        width: 100,
        height: 100,
        style: {
            gradient: radialGradient
        }
        // Text(label) added to the node
    }];
    // initialize Diagram component
    function App() {
        return (
            <DiagramComponent
                id="container"
                width={'100%'}
                height={'600px'}
                nodes={node}
            />
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

Shadow

Diagram provides support to add [shadow](#) effect to a node that is disabled, by default. It can be enabled with the

constraints property of the node. The following code illustrates how to drop shadow.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, NodeConstraints } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    constraints: NodeConstraints.Default | NodeConstraints.Shadow,
    // Text(label) added to the node
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```

```
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  constraints: NodeConstraints.Default | NodeConstraints.Shadow,
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Customizing shadow

The angle, distance, and opacity of the shadow can be customized with the shadow property of the node. The following code example illustrates how to customize shadow.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, NodeConstraints } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
```



```

        offsetY: 250,
        // Size of the node
        width: 100,
        height: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        },
        constraints: NodeConstraints.Default | NodeConstraints.Shadow,
        shadow: {
            angle: 50,
            opacity: 0.8,
            distance: 9
        }
        // Text(label) added to the node
    }
];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    constraints: NodeConstraints.Default | NodeConstraints.Shadow,
    shadow: {
        angle: 50,
        opacity: 0.8,
        distance: 9
    }
    // Text(label) added to the node
}
];

```

```
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Icon

Diagram provides support to describe the state of the node. i.e., the node is expanded or collapsed state.

Note: Icon can be created only when the node has outEdges.

- To explore the properties of expand and collapse icon, refer to [expandIcon](#) and [collapseIcon](#).
- The expandIcon's and collapseIcon's shape properties allow to define the shape of the icon.

The following code example illustrates how to create an icon of various shapes.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        content: 'Node1'
    }],
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    expandIcon: {
        shape: 'ArrowDown',
        width: 10,
        height: 10
    },
    collapseIcon: {
        shape: 'ArrowUp',
        width: 10,
        height: 10
    }
},
```

```

        id: 'Init',
        width: 140,
        height: 50,
        offsetX: 300,
        offsetY: 140,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        },
        annotations: [{
            content: 'Node2'
        }],
    }
];
let connectors = [{
    // Unique name for the connector
    id: "connector1",
    // Source and Target node's name to which connector needs to be
    // connected.
    sourceID: "Start",
    targetID: "Init",
    type: 'Orthogonal'
}];
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={nodes} connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let nodes: NodeModel[] = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        content: 'Node1'
    }],
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    expandIcon: {
        shape: 'ArrowDown',
        width: 10,

```

```

        height: 10
      },
      collapseIcon: {
        shape: 'ArrowUp',
        width: 10,
        height: 10
      }
    },
    {
      id: 'Init',
      width: 140,
      height: 50,
      offsetX: 300,
      offsetY: 140,
      style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
      },
      annotations: [{
        content: 'Node2'
      }],
    }
  ];
let connectors: ConnectorModel[] = [{
  // Unique name for the connector
  id: "connector1",
  // Source and Target node's name to which connector needs to be
  // connected.
  sourceID: "Start",
  targetID: "Init",
  type: 'Orthogonal'
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors = {connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Customizing expand icon

- Set the borderColor, borderWidth, and background color for an expandIcon using borderColor, borderWidth, and fill properties.
- Set a size for an expandIcon by using width and height properties.
- The expand icon can be aligned relative to the node boundaries. It has margin, offset, horizontalAlignment, and verticalAlignment settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.

- The [iconColor](#) property can be used to set the strokeColor of the Icon.

Customizing collapse icon

- Set the [borderColor](#),

[borderWidth](#), background color for an collapseIcon using [borderColor](#), [borderWidth](#), and [fill](#) properties.

- Set a size for collapseIcon by using [width](#) and

[height](#) properties.

- Like expand icon, collapse icon also can be aligned relative to the node boundaries. It has margin, offset, horizontalAlignment, and verticalAlignment settings. It is quite tricky, when all four alignments are used together but gives you more control over alignment.
- The [iconColor](#) property can be used to set the strokeColor of the Icon.

Interaction

Diagram provides support to drag, resize, or rotate the node interactively. For more information about editing a node at runtime, refer to [Edit Nodes](#).

Constraints

The constraints property of the node allows you to enable/disable certain features. For more information about node constraints, refer to [Node Constraints](#).

Custom properties

The [addInfo](#) property of the node allows to maintain additional information to the node.

Stack order

The nodes z-order property specifies the stack order of the node. A node with greater stack order is always in front of a node with a lower stack order.

Data flow

Node has the InEdges and OutEdges read-only property. In this property, you can find what are all the connectors that are connected to the node, and then you can find these connectors by using the [getObject](#) method in the diagram.

```
`ts
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
let node: NodeModel[] = [{
```

```
  id: 'node1',
```

```
  // Position of the node
```

```
  offsetX: 450,
```

```
  offsetY: 100,
```

```
  // Size of the node
```

```
width: 80,
height: 50,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white'
},
},
{
  id: 'node2',
  // Position of the node
  offsetX: 350,
  offsetY: 200,
  // Size of the node
  width: 80,
  height: 50,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
},
{
  id: 'node3',
  // Position of the node
  offsetX: 450,
  offsetY: 200,
  // Size of the node
  width: 80,
  height: 50,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
},
},
```

```
{
  id: 'node4',
  // Position of the node
  offsetX: 550,
  offsetY: 200,
  // Size of the node
  width: 80,
  height: 50,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
};

let connector: ConnectorModel[] = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  type: 'Orthogonal'
},
{
  id: 'connector2',
  sourceID: 'node1',
  targetID: 'node3',
  type: 'Orthogonal'
},
{
  id: 'connector3',
  sourceID: 'node1',
  targetID: 'node4',
  type: 'Orthogonal'
}
];

function App() {
  return (
```

```

<DiagramComponent
id="container"
width={'100%'}
height={'600px'}
nodes={node}
connectors = {connector}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

See Also

- [How to add annotations to the node](#)
- [How to add ports to the node](#)
- [How to enable/disable the behavior of the node](#)
- [How to add nodes to the symbol palette](#)
- [How to edit the node visual interface](#)
- [How to create diagram nodes using drawing tools](#)

Shapes in React Diagram component

Diagram provides support to add different kind of nodes. They are as follows:

- Text node
- Image node
- HTML node
- Native node
- Basic shapes
- Flow shapes

<!-- markdownlint-disable MD033 -->

<!-- markdownlint-disable MD010 -->

Text

Texts can be added to the diagram as [text](#) nodes. The shape property of the node allows you to set the type of node and for text nodes, it should be set as **text**. In addition, define the content object that is used to define the text to be added and style is used to customize the appearance of that text. The following code illustrates how to create a text node..

INDEX.JSX

```
import * as React from "react";
```



```

import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type of the shape as text
    shape: {
        type: 'Text',
        content: 'Text Element'
    },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: {
        strokeColor: 'none',
        fill: 'none',
        color: 'black',
        textAlign: 'Center'
    }
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type of the shape as text
    shape: {
        type: 'Text',
        content: 'Text Element'
    },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: {

```

```

        strokeColor: 'none',
        fill: 'none',
        color: 'black',
        textAlign: 'Center'
    }
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        // Add node
        nodes={node}
        // render initialized Diagram
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Image

Diagram allows to add images as [image](#) nodes. The shape property of node allows you to set the type of node and for image nodes, it should be set as **image**. In addition, the source property of shape enables you to set the image source.

The following code illustrates how an image node is created.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  // sets the type of the shape as image
  shape: {
    type: 'Image',
    source:
    'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png'
  },
  //Customizes the appearances such as text, font, fill, and stroke.
  style: {
    fill: 'none'
  }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}

```

```

    // Add node
    nodes={node}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  // sets the type of the shape as image
  shape: {
    type: 'Image',
    source:
'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png'
  },
  //Customizes the appearances such as text, font, fill, and stroke.
  style: {
    fill: 'none'
  }
}
]];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Base64 Encoded Image Into The Image Node:

The following code illustrates how add Base64 image into image node.

INDEX.JSX

[illegible]

```
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  // sets the type of the shape as image
  shape: {
    type: 'Image',
    source:
      'data:image/gif;base64,R0lGODlhPQBEAPeoAJosM//AwO/AwHVYZ/z595kzAP/s7P+goOXMv
      8+fhw/v739/f+8PD98fH/8mJl+fn/9ZWb8/PzWlWv///6wWGbImAPgTEMImIN9gUFCEm/gDALULD
      N8PAD6atYdCTX9gUNKlj8wZAKUsAOzZz+UMA0sJAP/Z2ccMDA8PD/95eX5NWvsJCOVNQPtFX/8zM
      8+QePLl38MGBR8JCP+zs9myn/8GBqwpAP/GxgwJCPny78lzlYlgjAJ8vAP9fX/+MjMUcAN8zM/9wc
      M8ZGcATEL+QePdZwf/29uc/P9cmJu9MTDImIN+/r7+/vz8/P8VNQGNugV8AAF9fX8swMNgtAFldO
      ICAGpNSUnNWSMQ5MBAQEJE3QPIGAM9AQMQGcG9vb6MhJsEdGM8vLx8fH98AANIWAMuQeL8fABkTE
      PPQ0OM5OSYdGfL5jo+Pj/+pqcsTE78wMFNGQLYmID4dGPvd3UBAQJmTkP+8vH9QUK+vr8ZWSHpzc
      JMmILdwcLOGCHRQUHxwck9PT9DQ00/v70w5MLypoG8wKOuwsP/g4P/Q0IcwKEswKMl8aJ9fX2xjd
      OtGRs/Pz+Dg4GImIP8gIH0sKEAwKKmTiKZ8aB/f39Wsl+Lft8dgUE9PT5x5aHBwcP+AgP+WltdgY
      MyZfyYwz78AAAAAAD///8AAP9mZv///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      CH5BAEAAKqALAAAAA9AEQAAj/AFEJHEiwoMGDCBMqXMiWocAbBww4nEhxoYkUpzJGrMixogkfG
      UNq1NixJEIDB0SqHGmyJSojM1bKZOmyop0gM3Oe2liTISKMOoPy7GnwY9CjIYcSRYm0aVKSLmE6n
      fq05QycVLPuhDrxBlCtYJUqNAq2bNWEBj6ZXRUyxZyDrtqwnXvkhACDV+euTeJm1Ki7A73qNWtFi
      F+/gA95Gly2CJLDhwEHMOUAAuOpLYDEgBxZ4GRTlC1fDnpkM+fOqD6DDj1aZpITp0dtGCDhr+fVu
      Cu3zlq49ijaokTZTo27uG7Gjn2P+hI8+PDPERoUB318bWbfAJ5sUNFcUGRTYUqV/3ogfXp1rWlMc
      6awJjiAAAd2fm4ogXjz56aypOoIde4OE5u/F9x199dlXnnGiHZWEYbGpsAEA3QXYnHwEfliKAgsWg
      J8LPeiUXGwedCAKABACCN+EAlpYIIYaFlcDhytd5lsgAJbo3onOpajiihl092KHGaUXGwWjUBChj
      SPiWJu00/LYIm4v1tXfE6J4gCSJEZ7YgRYUNrkji9P55sF/ogxw5ZkSqIDaZBV6aSGYq/lGZplnd
      kckZ98xoICbTcIJGQAZcNmmdUc210hs35nCyJ58fgmIKX5RQGOZowxaZwYA+JaoKQwswGijBV4C6
      SiTUmpphMspJx9unX4KaimjDv9aaXOEBteBqmuuxgEhOLX6Kqx+yXqqBANsgCtit4FWQAEkrNbpq
      7HSOmtwag5w57GrmlJBASEU18ADjUYb3ADTinIttsGsb1oJffa63bduimuqKB1keqWUhoCSK374w
      bujvOSu4QG6UvxBRydcPksav++Ca6G8A6Pr1x2kVMYHwsVxUALDq/krnrhPSOzXG11UTIoffqGR7
      Goi2MAxbv602kEG56I7CSlRsEFKfVYovDJoIRTg7sugNRDgqCJzJgcKE0ywc0ELm6KBCCJo8DIPF
```

```
eCWNGcyqNFE06ToAfV0HBRgxsvLThHn1oddQMrXj5DyAQgjEHSAJMWZwS3HPxT/QMbabI/iBCliM
LEJKX2EEkomBAUCxRi42VDADxyTYDVogV+wSCHqmKxEKCDAYFDFj4OmwbY7bDGdBhtrnTQYOigeC
hUmc1K3QTnAUfEgGFgAWt88hKA6aCRIXhxnQ1yg3BCayK44EWdkUQcBBYEQChFXfCB776aQsG0BI
lQgQgE8qO26X1h8cEUep8ngRBnOy74E9QgRgEAC8SvOfQkh7FDBDmS43PmGoIiKUUEGkMEC/PJHg
xw0xH74yx/3XnaYRjgMB8obxQW6kL9QYEJ0FIFgByfIL7/IQAlvQwEpnAC7DtLNJCKUoO/w45c44
GwCXiAFB/OXAATQryUxdN4LfFiwgjCNYg+kYMIEFkCKDs6PKAIJouyGWMS1FSKJOMRB/BoIxYJIU
XFUxNwoIkEKPAGCBZSQHQ1A2EWDfDEUVLyADj5AchSIQW6gu10bE/JG2VnCZGfo4R4d0sdQoBAHh
PjhIB94v/wRoRKQWGRHgrhGSQJxCS+0pCZbEhAAOw=='
    },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: {
      fill: 'none'
    }
  }
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Note: Deploy your HTML file in the web application and export the diagram (image node) or else the image node will not be exported in the Chrome and Firefox due to security issues. Refer to the following link.

Link 1: <http://asked.online/draw-images-on-canvas-locally-using-chrome/2546077/>

Link 2: <http://stackoverflow.com/questions/4761711/local-image-in-canvas-in-chrome>

Image alignment

Stretch and align the image content anywhere but within the node boundary.

The scale property of the node allows to stretch the image as you desired (either to maintain proportion or to stretch). By default, the [scale](#) property of the node is set as **meet**.

The following code illustrates how to scale or stretch the content of the image node.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
```

```

        width: 100,
        height: 100,
        //sets the type of the shape as Image
        shape: {
            type: 'Image',
            source:
'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png',
            scale: 'None'
        },
        //Customizes the appearances such as text, font, fill, and stroke.
        style: {
            fill: 'none'
        }
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    NodeConstraints,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //sets the type of the shape as Image
    shape: {
        type: 'Image',
        source:
'https://ej2.syncfusion.com/demos/src/diagram/employees/image16.png',
        scale: 'None'
    },
    //Customizes the appearances such as text, font, fill, and stroke.
    style: {
        fill: 'none'
    }
}];
// initialize Diagram component
function App() {
    return (

```

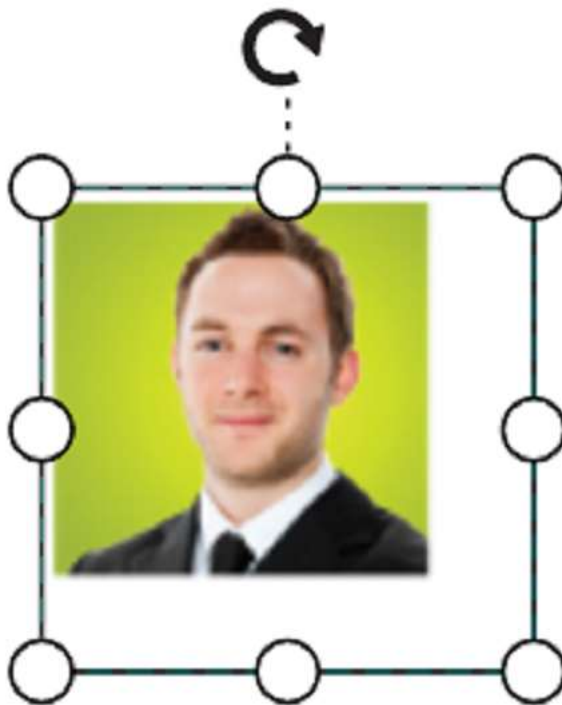
```
<DiagramComponent
  id="container"
  width={'100%'}
  height={'600px'}
  // Add node
  nodes={node}
  // render initialized Diagram
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

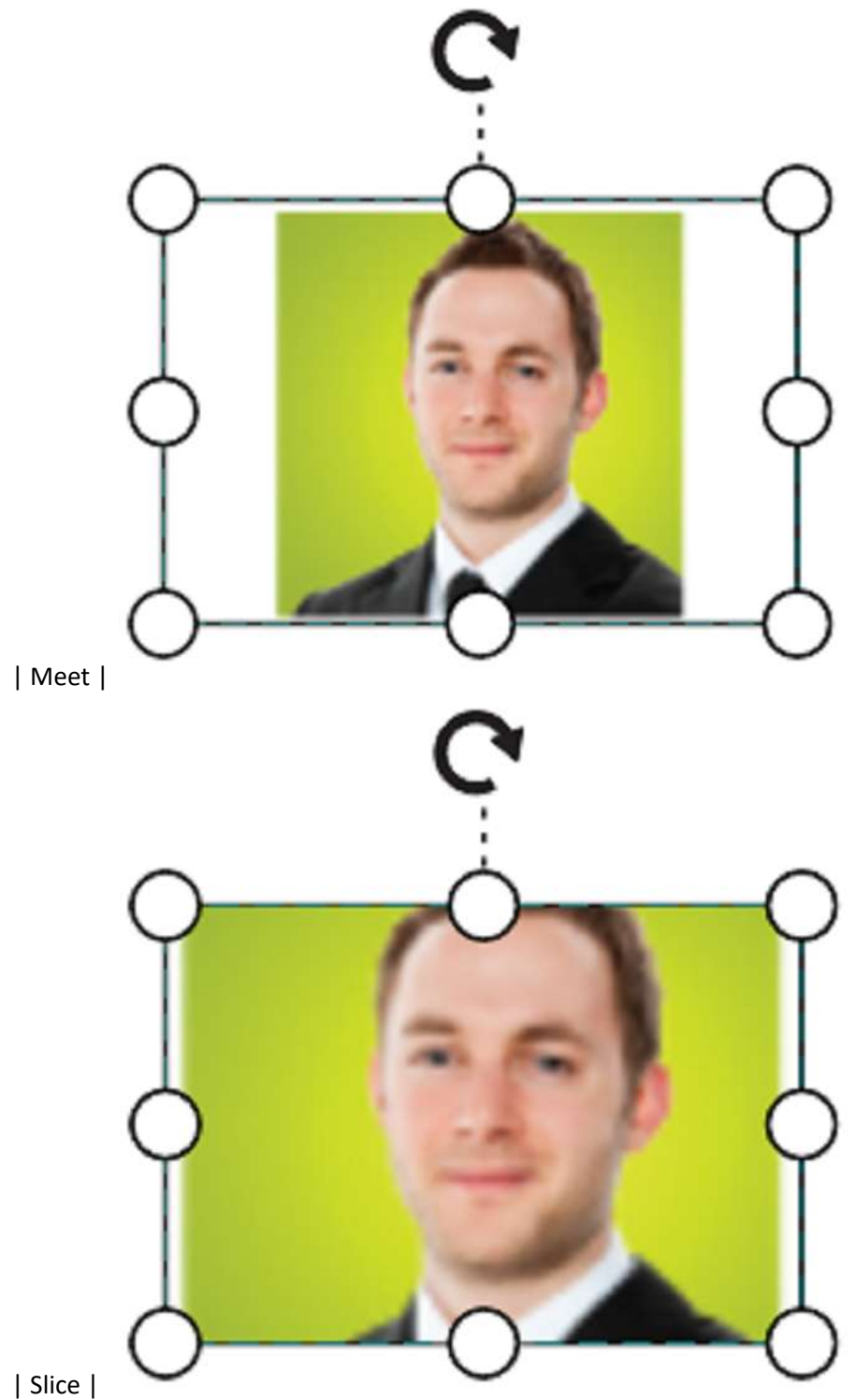
The following table illustrates all the possible scale options for the image node.

| Values | Images |

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

| None |







| Stretch |

HTML

Html elements can be embedded in the diagram through [Html](#) type node. The shape property of node allows you to set the type of node and to create a HTML node it should be set as `HTML`. The following code illustrates how an Html node is created.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
```

```
width: 100,
height: 100,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white'
},
//sets the type of the shape as HTML
shape: {
  type: 'HTML',
  content: '<div style="background:#6BA5D7;height:100%;width:100%;"><button type="button"
  style="width:100px"> Button</button></div>',
}
});
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
    id="container"
    width={'100%'}
    height={'600px'}
    // Add node
    nodes={node}
    // render initialized Diagram
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Note: HTML node cannot be exported to image format, like JPEG, PNG, and BMP. It is by design, while exporting the diagram is drawn in a canvas. Further, this canvas is exported into image formats. Currently, drawing in a canvas equivalent from all possible HTML is not feasible. Hence, this limitation.

HTML Node With Template

Html elements can be embedded in the diagram using [Html](#) type node. The shape property of the node allows you to set the type of node. The following code shows how an Html node is created with a template.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
//A node is created and stored in nodes array.
let node: NodeModel[] = [{
  //Id of the node
  id: "Node",
  //Position of the node
  offsetX: 250,
  offsetY: 250,
  //Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as HTML
  shape: {
    type: 'HTML'
  }
}];
function diagramTemplate(props) {
  if (props.id === "node") {
    <input type="button" id="button" value={props.id}>
  }
}
function App() {
```

```

return (
  <DiagramComponent
    id="container"
    width={'100%'}
    height={'600px'}
    // Add node
    nodes={node}
    nodeTemplate={diagramTemplate.bind(this)}
  />
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Native

Diagram provides support to embed SVG element into a node. The shape property of node allows you to set the type of node. To create a [native](#) node, it should be set as **native**. The following code illustrates how a native node is created.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";

// A node is created and stored in nodes array.
let node: NodeModel []= [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,

```

height: 100,

//sets the type of the shape as Native

shape: {

type: 'Native',

```
content: '<g xmlns="http://www.w3.org/2000/svg"><g transform="translate(1 1)"><g>  <path
style="fill:#61443C;" d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-1.109c-2.01,1.33-4.472,
1.791-6.827,1.28 C62.726,435.13,62.354,435.072,61.979,435.057z"/><path
style="fill:#61443C;" d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-49.749-66.304 c-5.784-
1.581-11.753-2.385-17.749-2.389c-2.425-0.028-4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-
23.296 c0.377-47.729-34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-2.731,23.296c2.405-0.312,4.829-
0.455,7.253-0.427  C472.572,434.123,502.783,464.869,502.469,502.471z"/>      </g>  <path
style="fill:#8B685A;" d="M476.869,502.471h7.536c-0.191-32.558,22.574-60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-1.28c1.707-0.085,3.413-0.171,5.12-
0.171  c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-1.288,3.414-
3.341,3.879-5.685  c7.68-39.706,39.605-70.228,79.616-76.117c4.325-0.616,8.687-0.929,13.056-
0.939c13.281-0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.345-0.598,4.337-
2.142,5.502-4.264  c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-0.9,15.666-2.731,23.296
c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389C456.696,444.61,477.033,471.713,476.869,502.471
L476.869,502.471z"/>      <path style="fill:#66993E;" d="M502.469,7.537c0,0-6.997,264.96-
192.512,252.245c-20.217-1.549-40.166-5.59-59.392-12.032  c-1.365-0.341-2.731-0.853-4.096-
1.28c0,0-0.597-2.219-1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>      <path style="fill:#9ACA5C;"
d="M476.784,10.183c-1.28,26.197-16.213,238.165-166.827,249.6  c-20.217-1.549-40.166-5.59-59.392-
12.032c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z"/>      <path style="fill:#66993E;"
d="M206.192,246.727c-0.768,3.925-1.365,6.144-1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28  c-
21.505,7.427-44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-1.792-
0.085-2.731-0.085-2.731  C222.747,86.129,211.653,216.689,206.192,246.727z"/>      <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-1.365,6.144c-1.365,0.427-
2.731,0.939-4.096,1.28  c-13.351,4.412-27.142,7.359-
41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/></g>  <g>      <path
d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-6.858-1.849-9.298l-8.533-8.533  c-
3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671z"/>      <path
d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-12.012l-
8.533-8.533  c-3.341-3.281-8.701-3.256-12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>
      <path d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-
0.054,12.012l8.533,8.533  c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-
12.012L296.169,454.771z"/>      <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-
0.054c3.311-3.311,3.335-8.671,0.054-12.012l-8.533-8.533  c-3.341-3.281-8.701-3.256-12.012,0.054c-
```

```
3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>          <path
d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-3.341,3.256-8.701-0.054-
12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/>          <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-3.341,3.256-8.701-0.054-
12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/>          <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-3.341,3.256-8.701-0.054-
12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/>          <path d="M164.636,454.771l-
8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065
c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771l164.636,454.771z"/>          <path
d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171l232.903,429.171z"/>          <path
d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-3.341,3.256-8.701-0.054-
12.012 c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/>          <path d="M70.77,463.304l-
8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271s3.1,5.28,6.065,6.065 c2.965,0.785,6.122-
0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304l70.77,463.304z"/>          <path
d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012 C130.67,442.981,125.31,442.957,121.97,446.238l121.97,446.238z"/>
          <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-4.433,0.899-6.033,2.5l-
8.533,8.533 c-2.178,2.151-3.037,5.304-
2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-0.073,8.262-2.251l8.533-8.533
c1.601-1.6,2.5-3.77,2.5-6.033C204.802,424.408,203.903,422.237,202.302,420.638l202.302,420.638z"/>
          <path d="M210.836,463.304c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-
0.054,12.012l8.533,8.533 c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-
6.065c0.785-2.965-0.082-6.122-2.27-8.271l210.836,463.304z"/>          <path
d="M343.836,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065 c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771l343.836,454.771z"/>          <path
d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-8.671,0.054-12.012l-8.533-8.533 c-
3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l429.17,483.904z"/>
          <path d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012 s-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>          <path
d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.5l8.533-8.533c3.281-3.341,3.256-8.701-0.054-
12.012 s-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>          <path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62.581-8.828,88.583-
26.796 c94.225-65.725,99.567-227.462,99.75-234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-
1.679-4.141-2.52-6.558-2.308 C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
```

```

38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881 c-4.581-7.691-9.996-14.854-
16.147-21.358c8.023-38.158,0.241-77.939-21.57-110.261C160.753,95.829,98.828,68.458,9.228,61.196
c-2.417-0.214-4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665 c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-
5.666-26.723-8.595-40.759-8.6c-49.449,0.497-91.788,35.567-101.483,84.058 c-5.094-1.093-10.29-
1.641-15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-2.499c1.6-1.6,2.499-
3.771,2.499-6.035 c0.249-41.103-31.914-75.112-72.967-77.154c0.65-4.78,0.975-9.598,0.975-
14.421c0.914-45.674-28.469-86.455-72.083-100.045 c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.138z M87.519,238.092
c-55.35-38.567-67.358-129.25-69.833-158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-162.473-140.577c-2.82-
1.165-6.048-0.736-8.466,1.125 s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.776,161.392,144.874
C150.795,260.908,115.29,257.451,87.519,238.092z M279.969,114.046c37.6-53.788,109.708-
86.113,214.408-96.138 c-2.65,35.375-17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-
142.305,13.433 c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892 c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z M262.694,368.017c15.097-
26.883,43.468-43.587,74.3-43.746 c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-
0.857,14.351-2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661 c3.899-29.333,28.777-51.334,58.367-
51.617c4-0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467 c3.858-2.519,6.531-
6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-70.192c16.045-
0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,368.017L262.694,368.
017z"/></g></g>'

```

```

},

```

```

});

```

```

// initialize Diagram component

```

```

function App() {

```

```

  return (

```

```

    <DiagramComponent

```

```

      id="container"

```

```

      width={'100%'}

```

```

      height={'600px'}

```

```

    // Add node

```

```

    nodes={node}

```



```

/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Note: Like HTML node, the native node also cannot be exported to image format. Fill color of native node can be overridden by the inline style or fill of the SVG element specified in the template.

SVG content alignment

Stretch and align the svg content anywhere but within the node boundary.

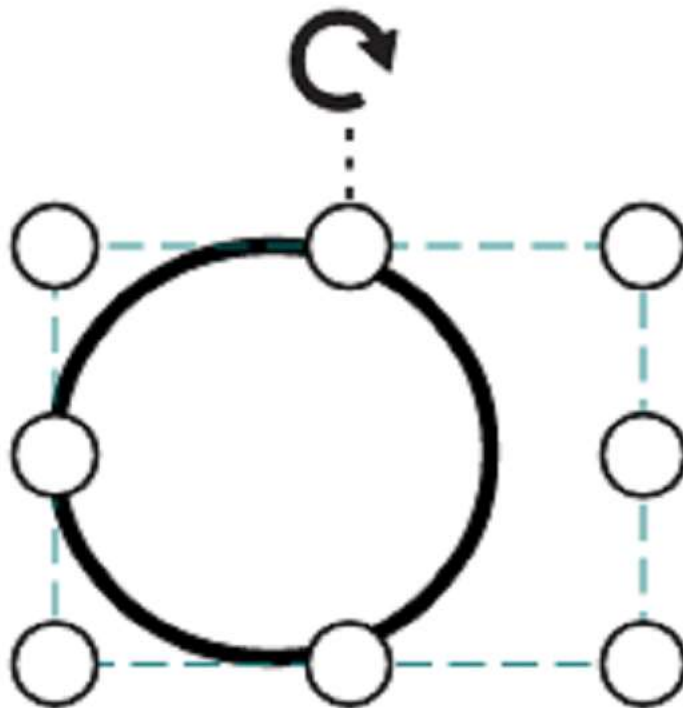
The scale property of the node allows to stretch the svg content as you desired (either to maintain proportion or to stretch). By default, the `scale` property of node is set as **meet**. The following code illustrates how to scale or stretch the content of the node.

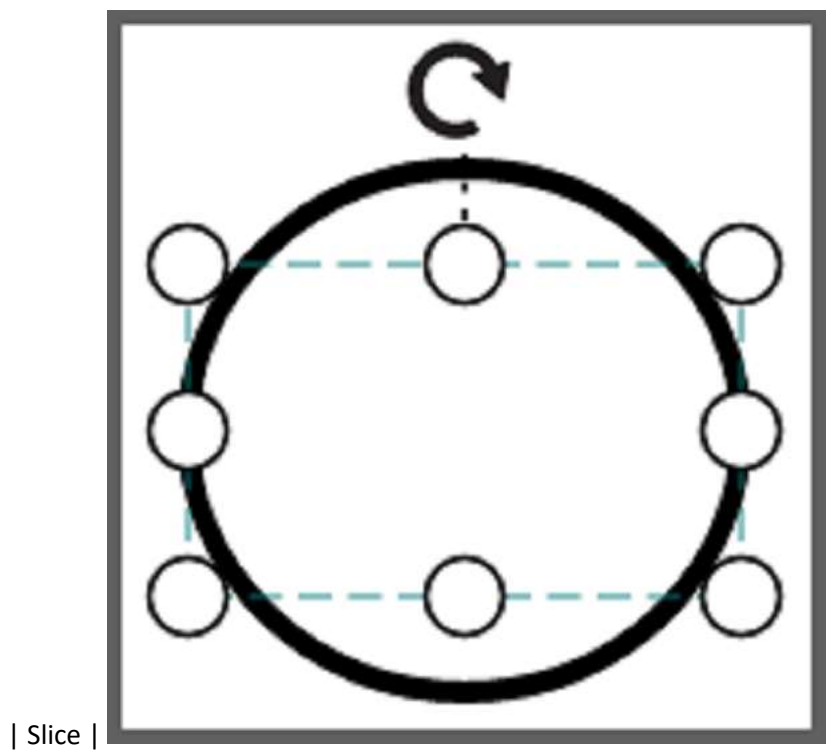
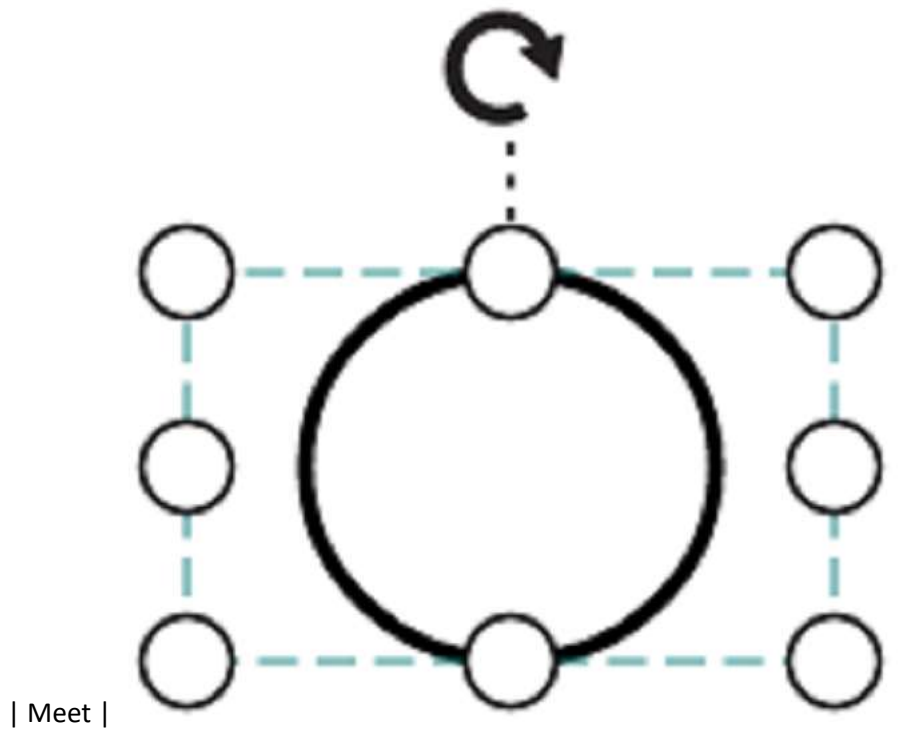
The following tables illustrates all the possible scale options for the node.

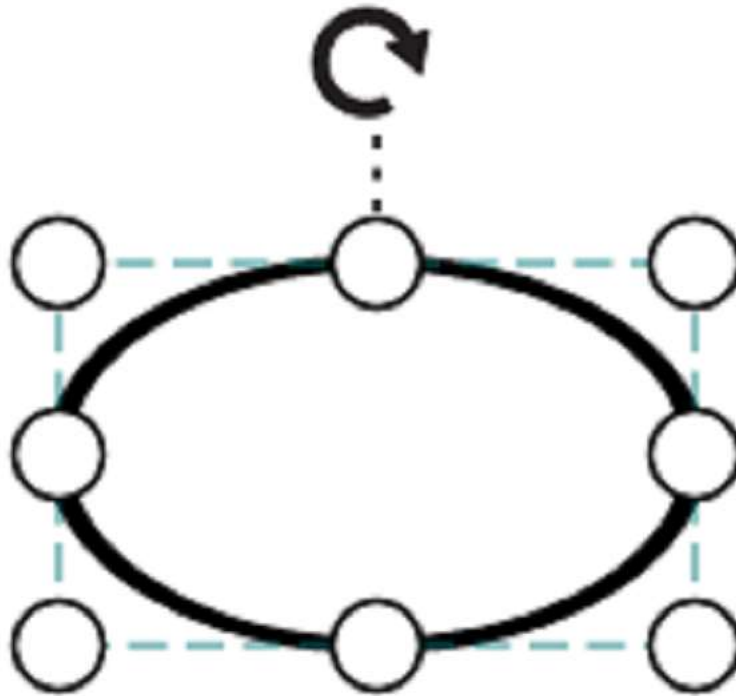
Values	Images
None	

None	
------	--

None	
------	--







| Stretch |

Basic shapes

- The [Basic](#) shapes are common shapes that are used to represent the geometrical information visually. To create basic shapes, the type of the shape should be set as **basic**. Its shape property can be set with any one of the built-in shape.
- To render a rounded rectangle, you need to set the type as basic and shape as rectangle. Set the [cornerRadius](#) property to specify the radius of rounded rectangle.

The following code example illustrates how to create a basic shape.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as Basic
  shape: {
    type: 'Basic',
    shape: 'Rectangle',
    cornerRadius: 10
  },
  style: {
```

```

        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as Basic
  shape: {
    type: 'Basic',
    shape: 'Rectangle',
    cornerRadius: 10
  },
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    />
  );
}

```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Note: By default, the **shape** property of the node is set as **basic**.

Default property for shape is Rectangle.

Note: When the **shape** is not set for a basic shape, it is considered as a **rectangle**.

The list of basic shapes are as follows.



Path

The **Path** node is a commonly used basic shape that allows visually to represent the geometrical information. To create a path node, specify the shape as **path**. The path property of node allows you to define the path to be drawn. The following code illustrates how a path node is created.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as Path
  shape: {
    type: 'Path',
    data: 'M35.2441,25 L22.7161,49.9937 L22.7161,0.00657536
L35.2441,25 z M22.7167,25 L-0.00131226,25 M35.2441,49.6337 L35.2441,0.368951
M35.2441,25 L49.9981,25'
  },
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  // Add node
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as Path
  shape: {
    type: 'Path',
    data: 'M35.2441,25 L22.7161,49.9937 L22.7161,0.00657536 L35.2441,25
z M22.7167,25 L-0.00131226,25 M35.2441,49.6337 L35.2441,0.368951 M35.2441,25
L49.9981,25'
  },
  },
  ];
  // initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Flow Shapes

The [flow](#) shapes are used to represent the process flow. It is used for analyzing, designing, and managing for documentation process. To create a flow shape, specify the shape type as **flow**. Flow shapes and by default, it is considered as **process**. The following code example illustrates how to create a flow shape.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,

```

```

    // Size of the node
    width: 100,
    height: 100,
    //sets the type of the shape as Flow
    shape: {
        type: 'Flow',
        shape: 'Document'
    },
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
  }];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  NodeConstraints,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //sets the type of the shape as Flow
  shape: {
    type: 'Flow',
    shape: 'Document'
  },
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}

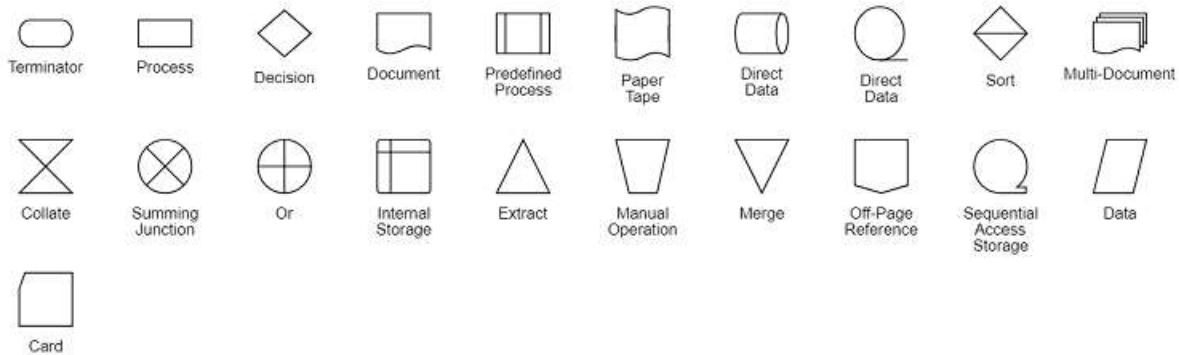
```

```

    height={'600px'}
    // Add node
    nodes={node}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The list of flow shapes are as follows.



Bpmn shapes in React Diagram component

BPMN shapes are used to represent the internal business procedure in a graphical notation and enable you to communicate the procedures in a standard manner. To create a BPMN shape, in the node property shape, type should be set as “bpmn” and its shape should be set as any one of the built-in shapes. The following code example illustrates how to create a simple business process.

Note: If you want to use BPMN shapes in React Diagram, you need to inject BpmnDiagrams in the diagram.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Event
  shape: {
    type: 'Bpmn',
    shape: 'Event',
    // set the event type as End
    event: {
      event: 'End'
    }
  },
},

```



```

    }];
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
        <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Event
    shape: {
        type: 'Bpmn',
        shape: 'Event',
        // set the event type as End
        event: {
            event: 'End'
        }
    },
}];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            // Add node
            nodes={node}
        >
            <Inject services={[BpmnDiagrams]} />
        </DiagramComponent>
    );
}

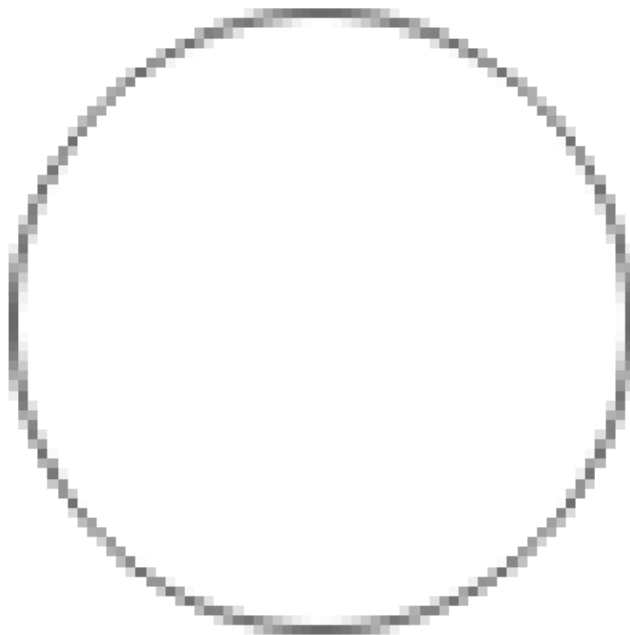
```

```
}  
const root = ReactDOM.createRoot(document.getElementById('diagram'));  
root.render(<App />);
```

Note : The default value for the property **shape** is “event”.

The list of BPMN shapes are as follows:

Shape	Image
-----	-----

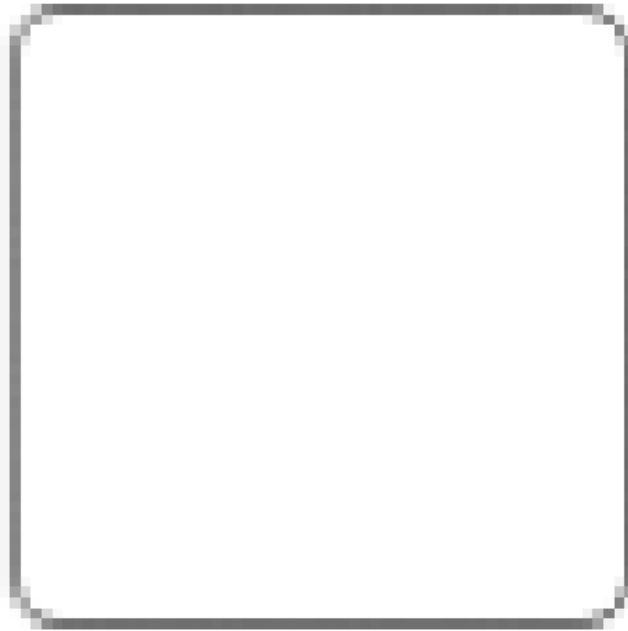


Event	
-------	--



| Gateway |

|



| Task |

|

| Message |

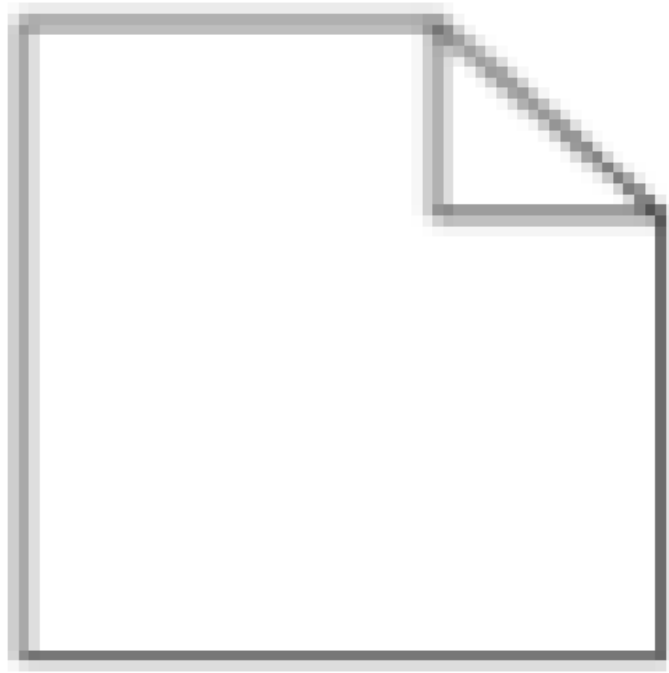


|



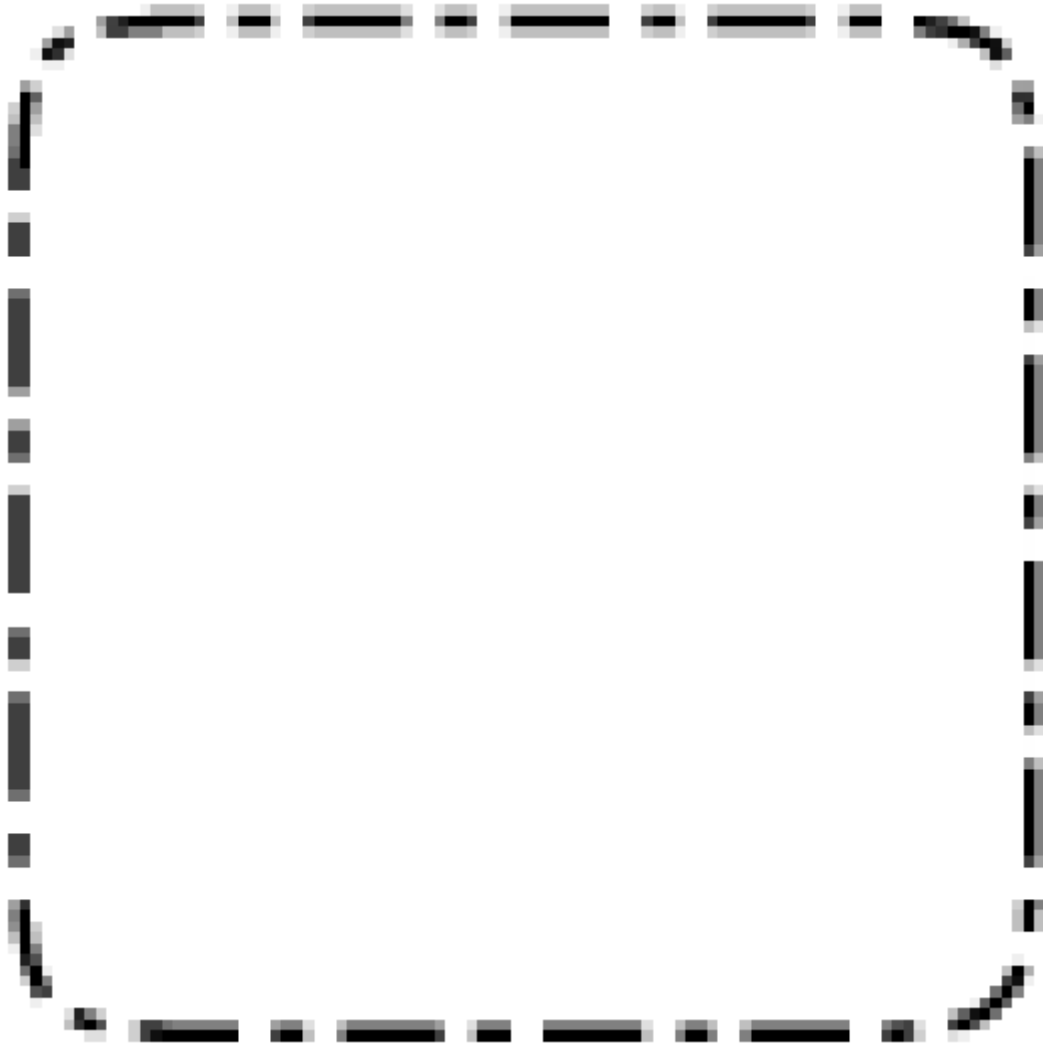
| DataSource |

|



| DataObject |

|



| Group |

The BPMN shapes and its types are explained as follows.

<!-- markdownlint-disable MD033 -->

Event

An [event](#) is notated with a circle and it represents an event in a business process. The type of events are as follows:

- Start
- End
- Intermediate

The [event](#) property of the node allows you to define the type of the event. The default value of the event is **start**. The following code example illustrates how to create a BPMN event.

INDEX.JSX

```
import * as React from "react";  
import * as ReactDOM from "react-dom";
```



```
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as event
  shape: {
    type: 'Bpmn',
    shape: 'Event',
    // Sets event as End and trigger as None
    event: {
      event: 'End',
      trigger: 'None'
    }
  },
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
    <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as event
```

```

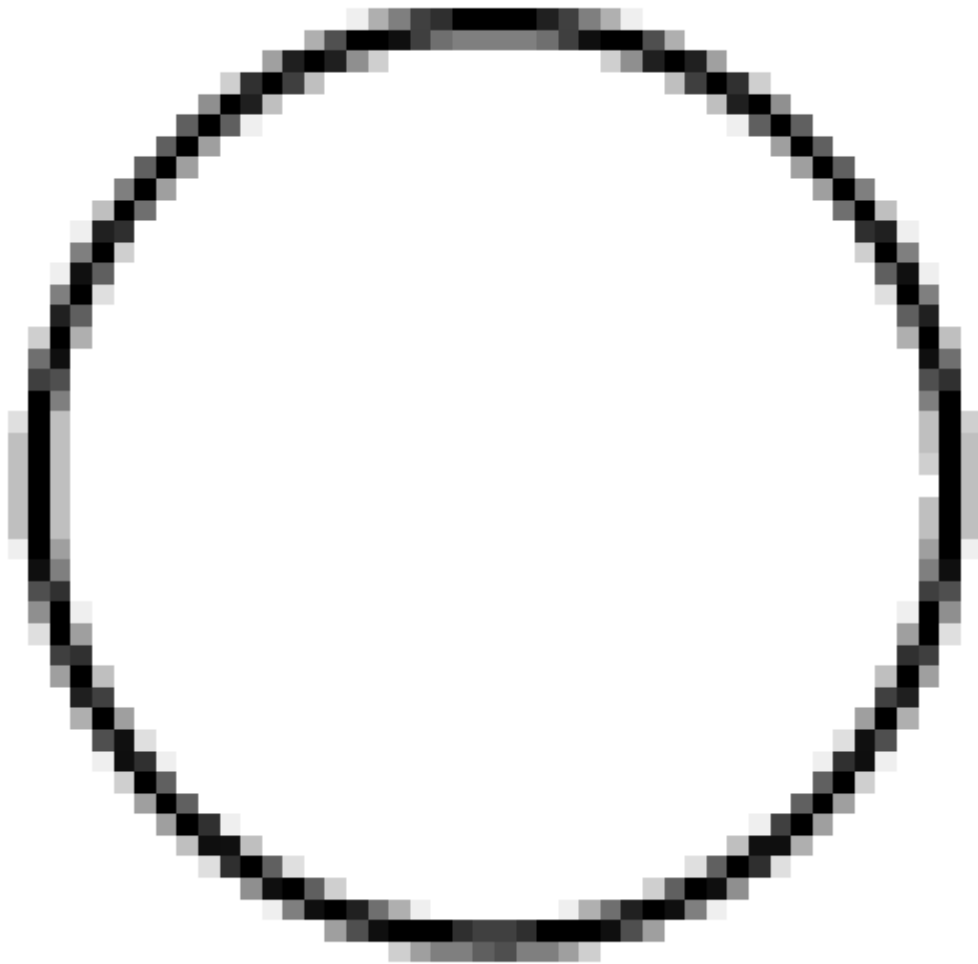
    shape: {
      type: 'Bpmn',
      shape: 'Event',
      // Sets event as End and trigger as None
      event: {
        event: 'End',
        trigger: 'None'
      }
    },
  ]];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Event triggers are notated as icons inside the circle and they represent the specific details of the process. The [trigger](#) property of the node allows you to set the type of trigger and by default, it is set as **none**. The following table illustrates the type of event triggers.

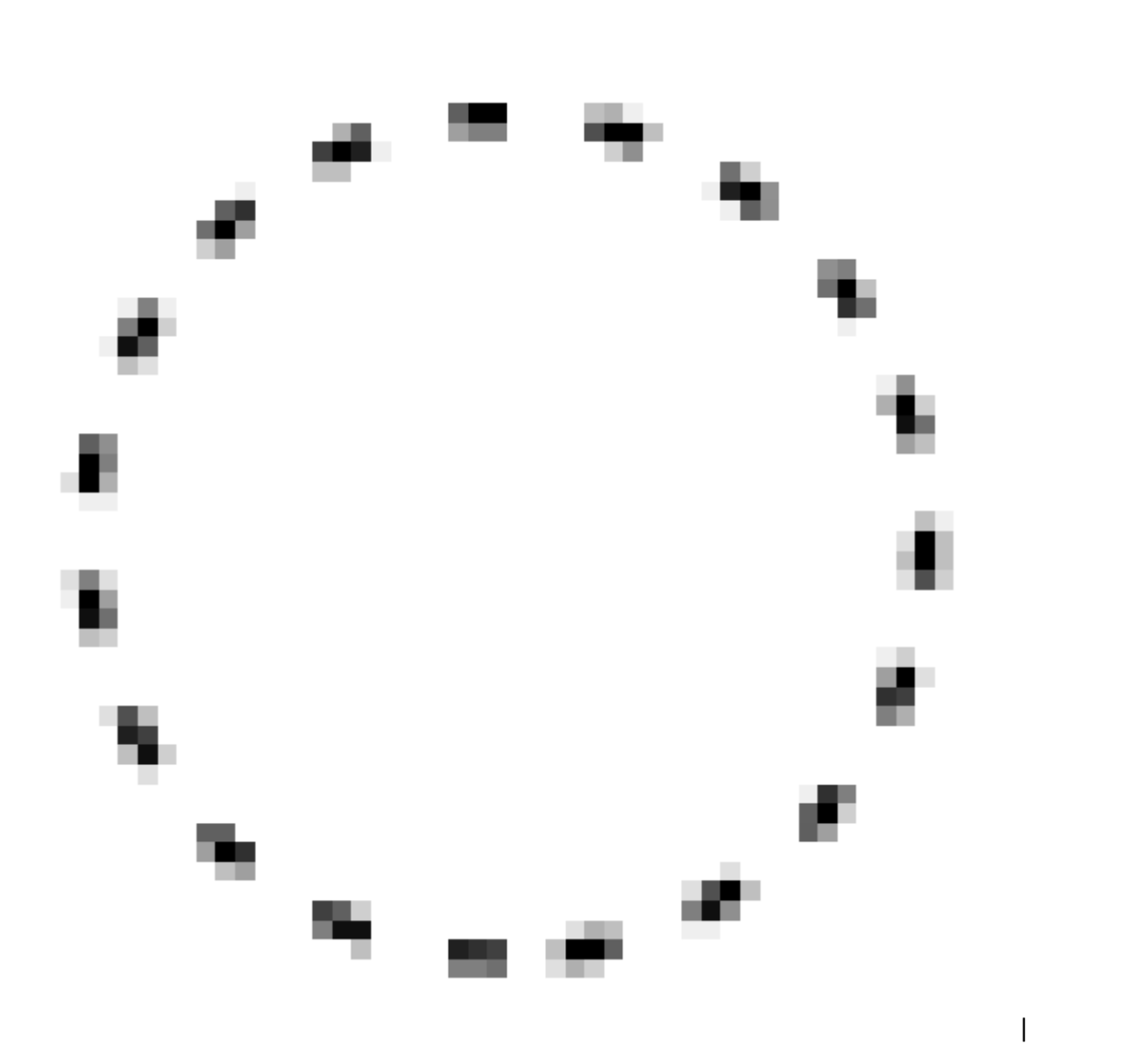
Triggers	Start	Non-Interrupting Start	Intermediate	Non-Interrupting Intermediate	Throwing Intermediate	End
----------	-------	------------------------	--------------	-------------------------------	-----------------------	-----

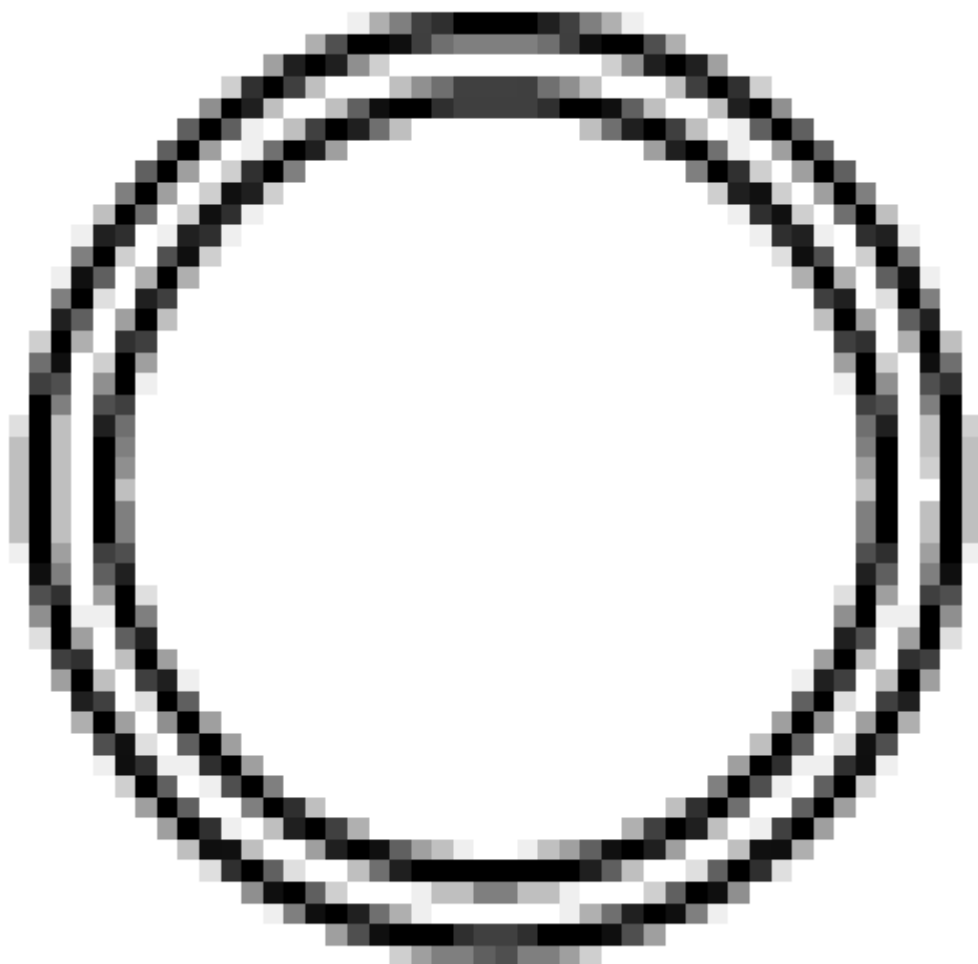
-----	-----	-----	-----	-----	-----	-----
-------	-------	-------	-------	-------	-------	-------



| None |

|

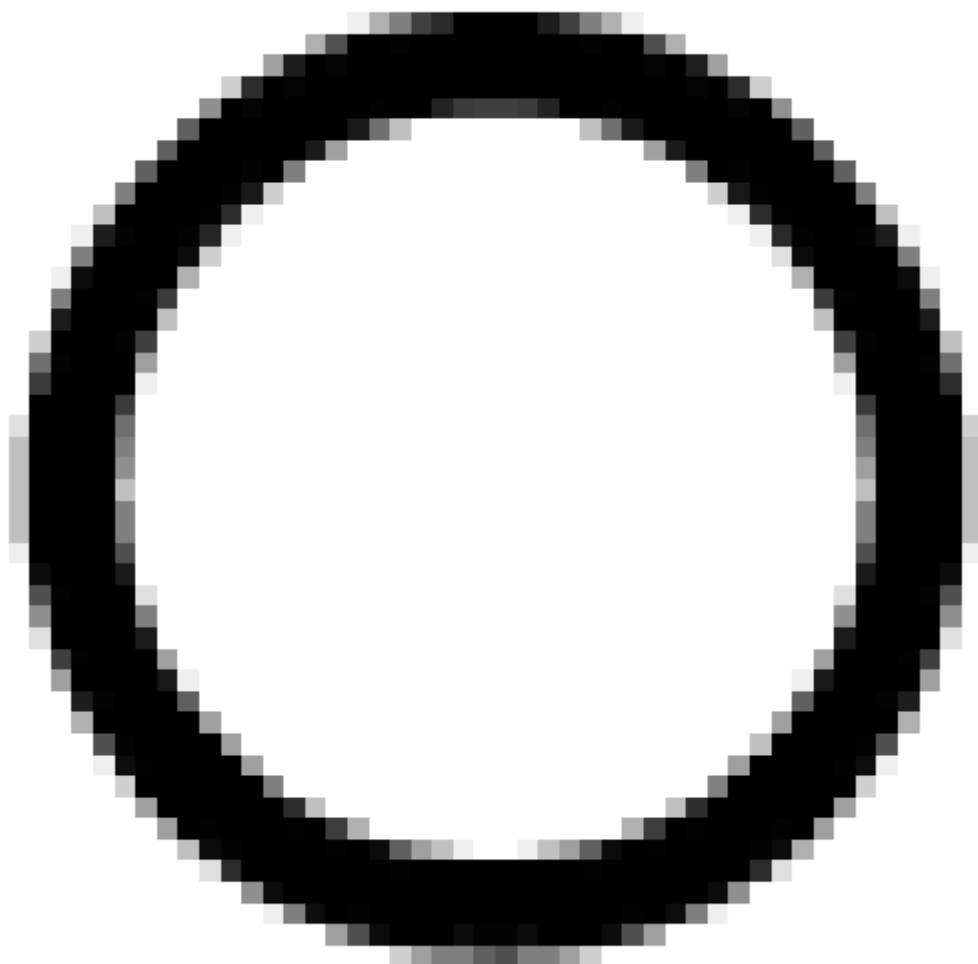




|



||



|

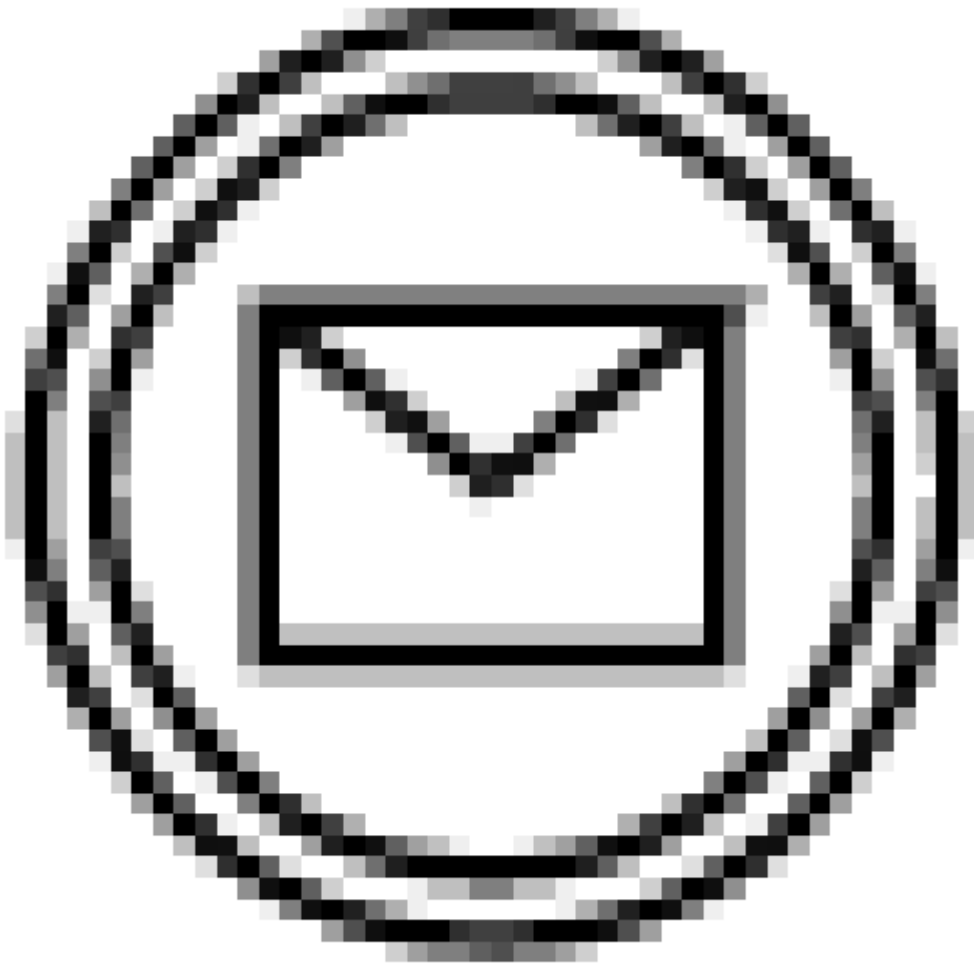
| Message |



|



|



|



|



|



|

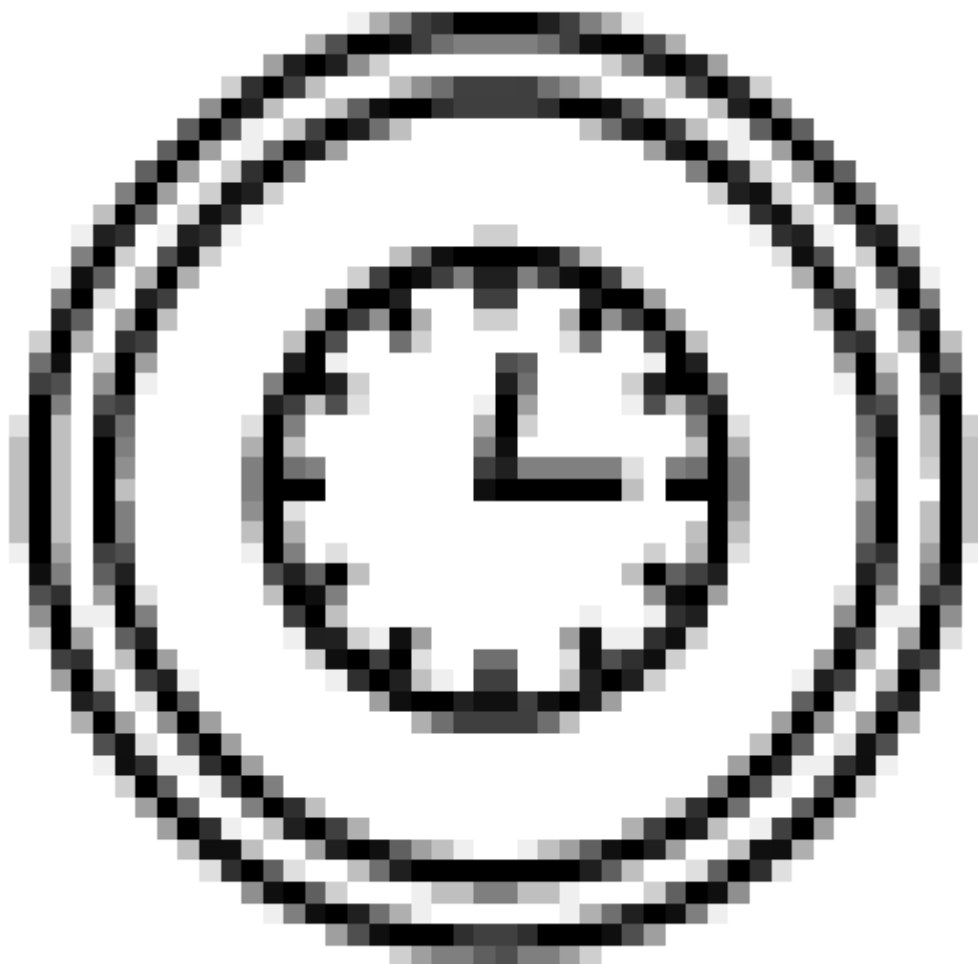


| Timer |

|



|



|

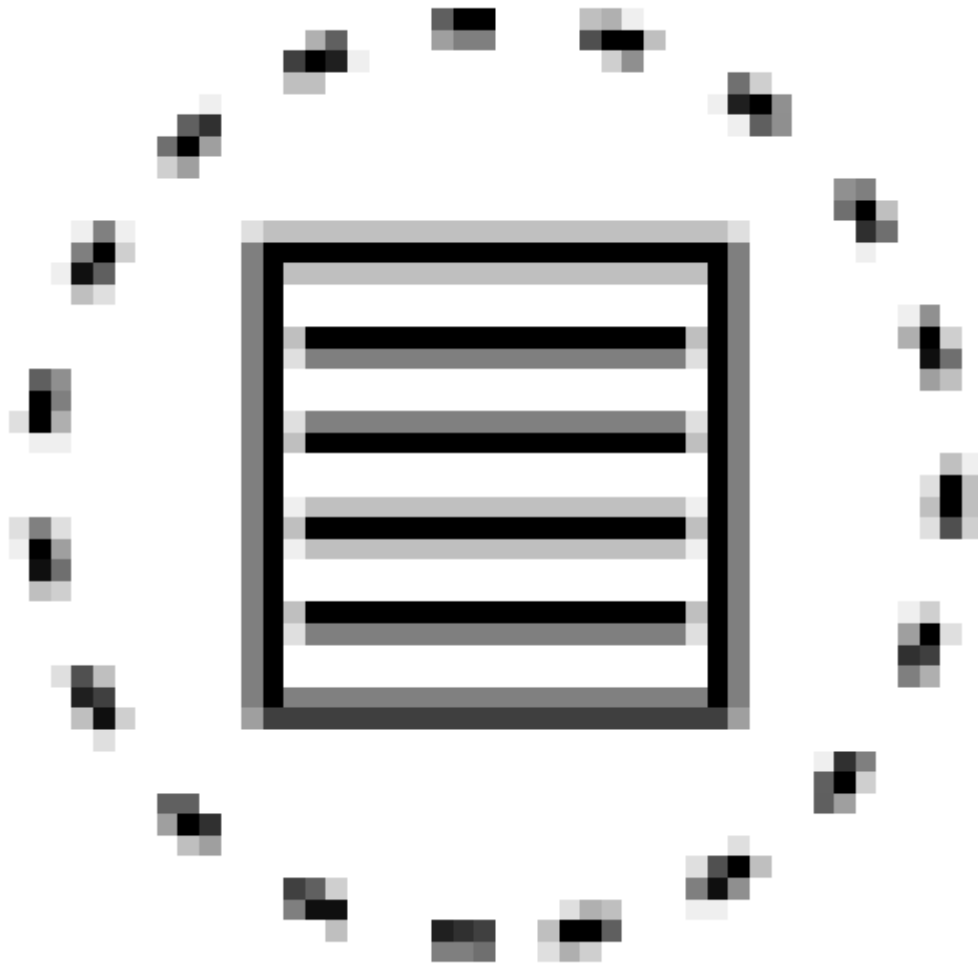


|||

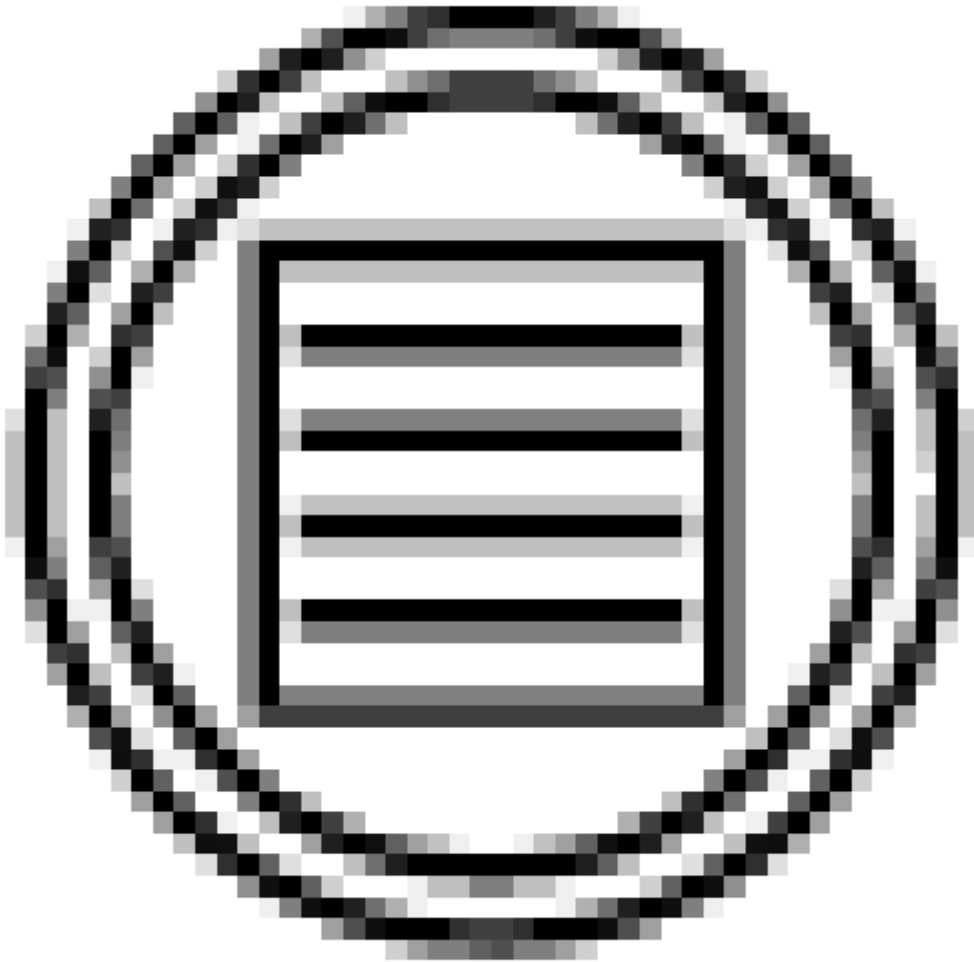
| Conditional |



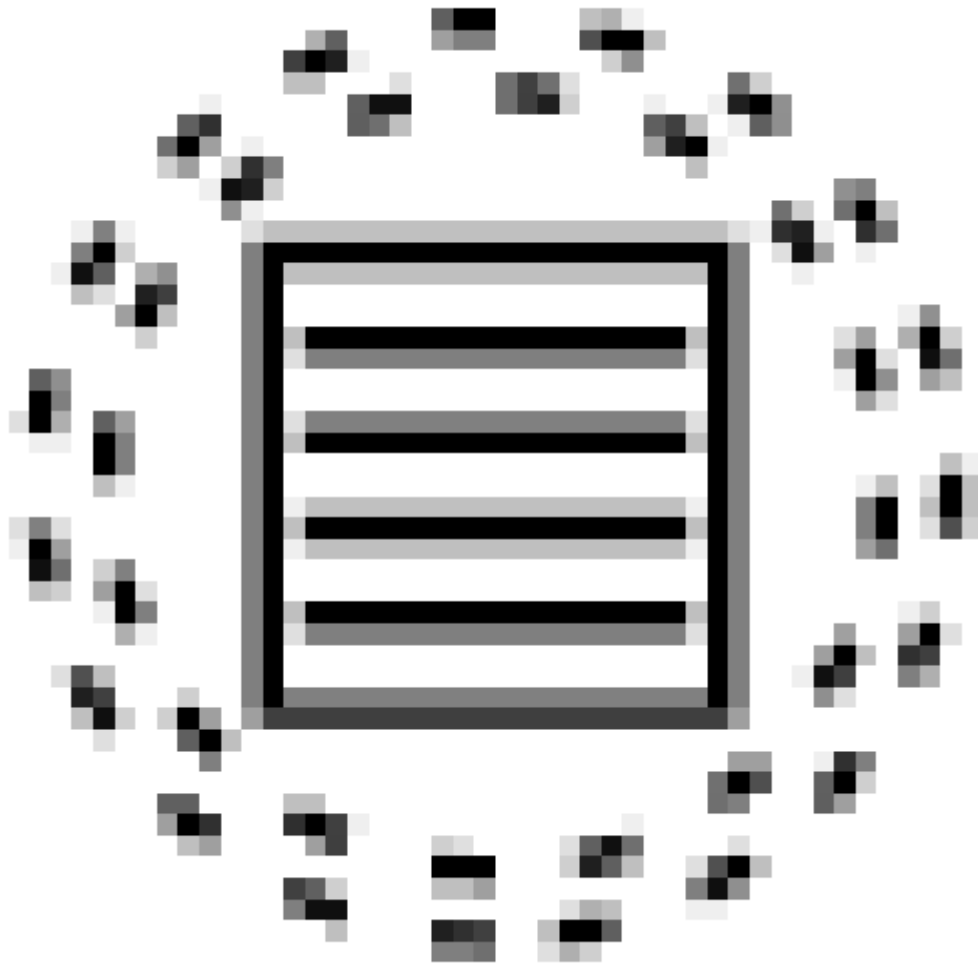
|



|



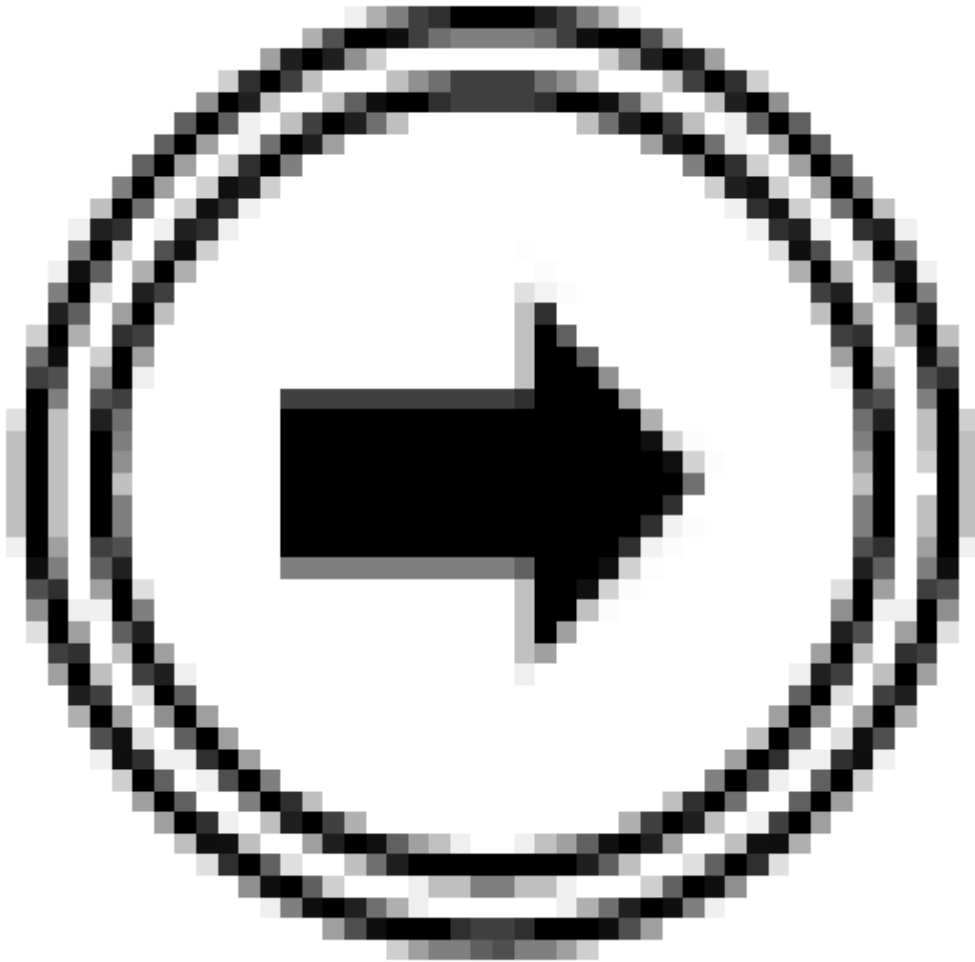
|



|||



| [Link](#) | | |



||

||



| Signal |

|



|



|



|



|



|



| Error |

|



|

|||



|

| Escalation |



|



|



|



|

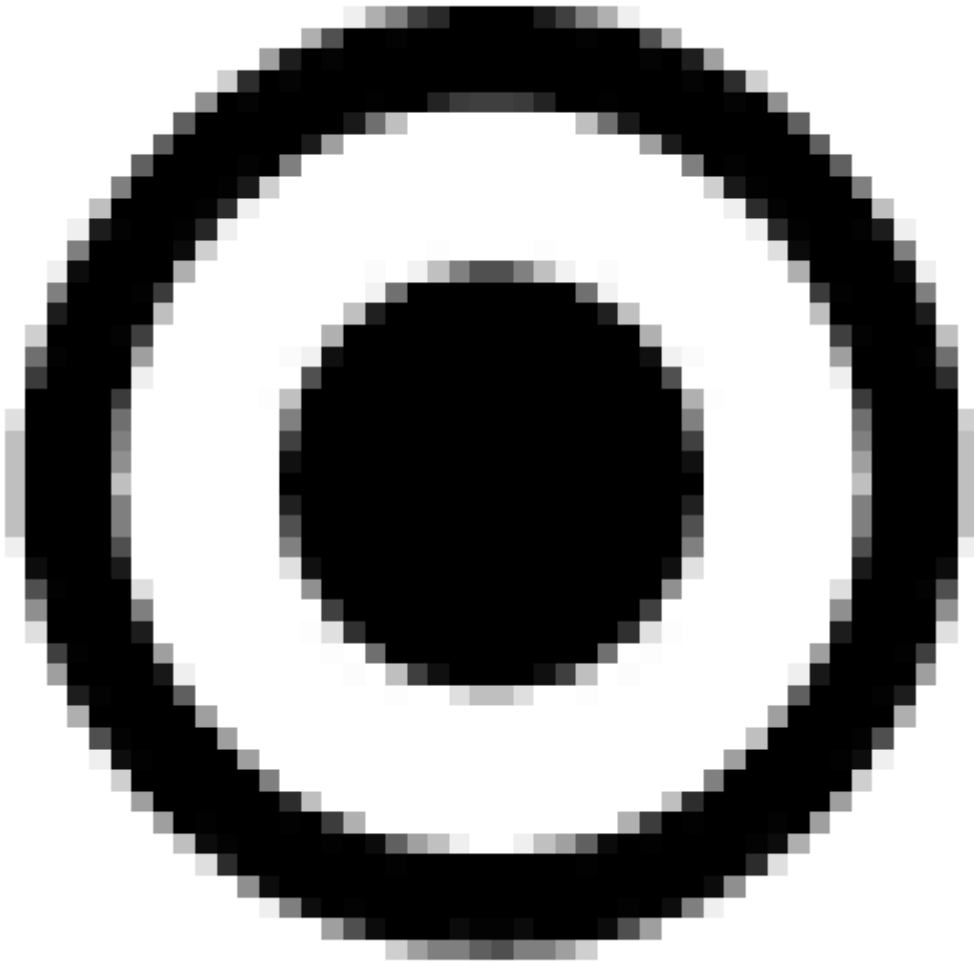


|



|

| Termination | | | | |



|

| Compensation |



||



||

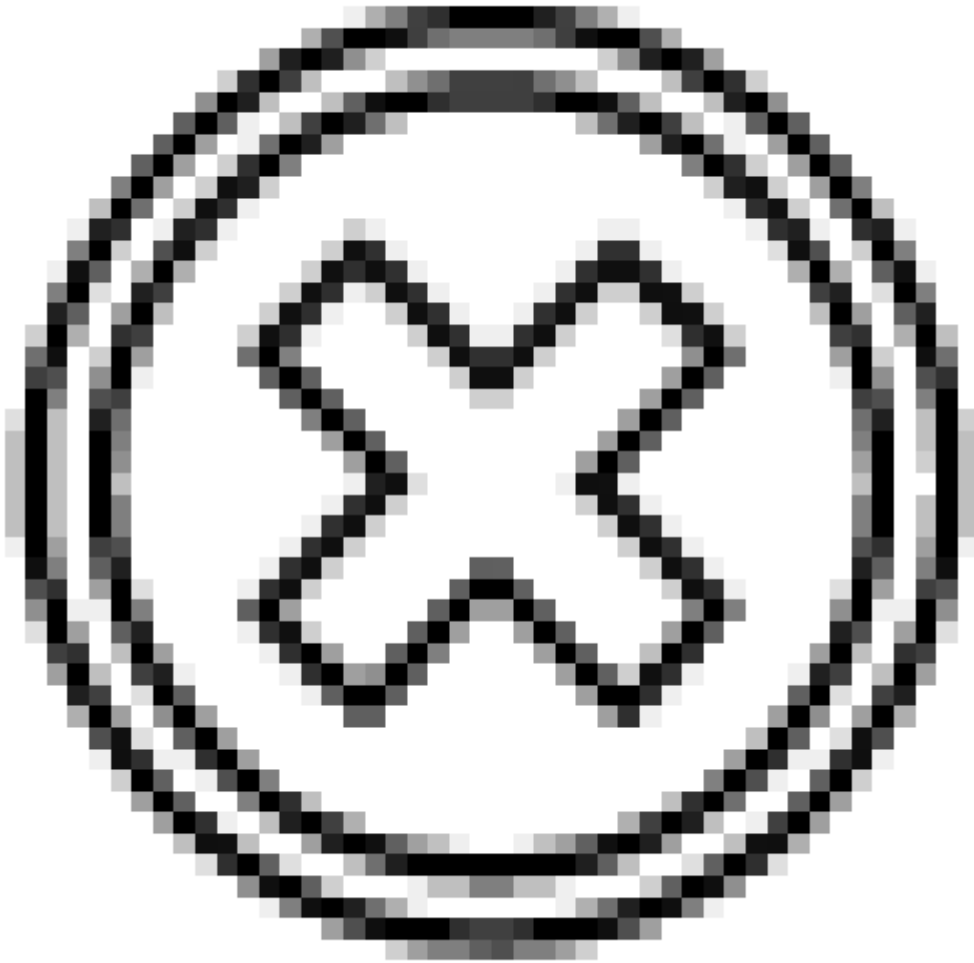


|



|

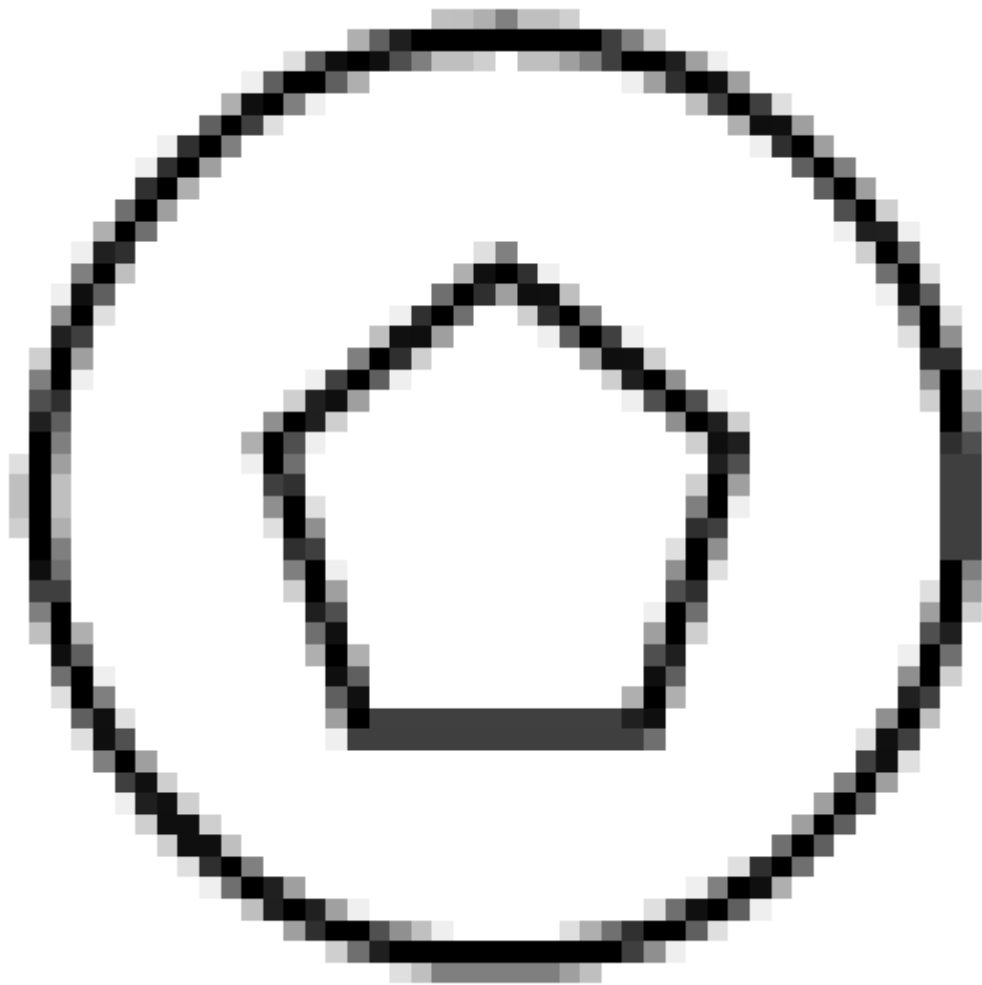
| Cancel | | |



| | |

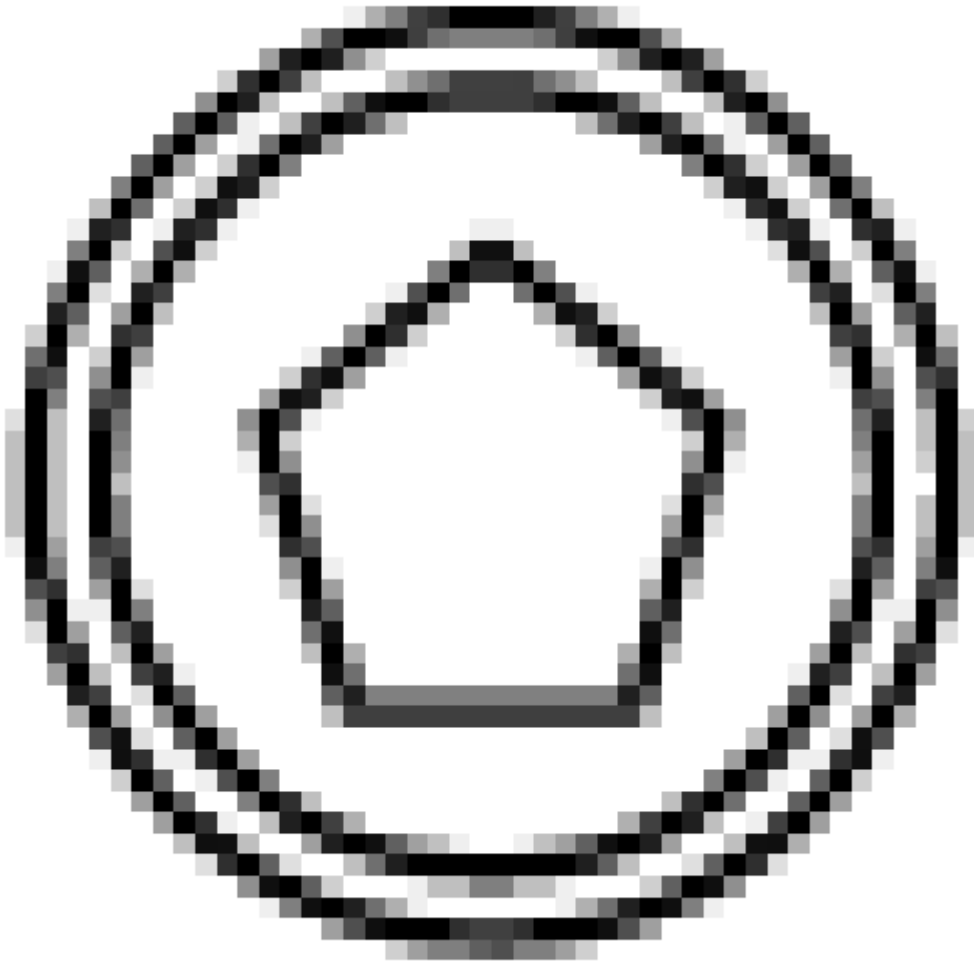


|



| Multiple |

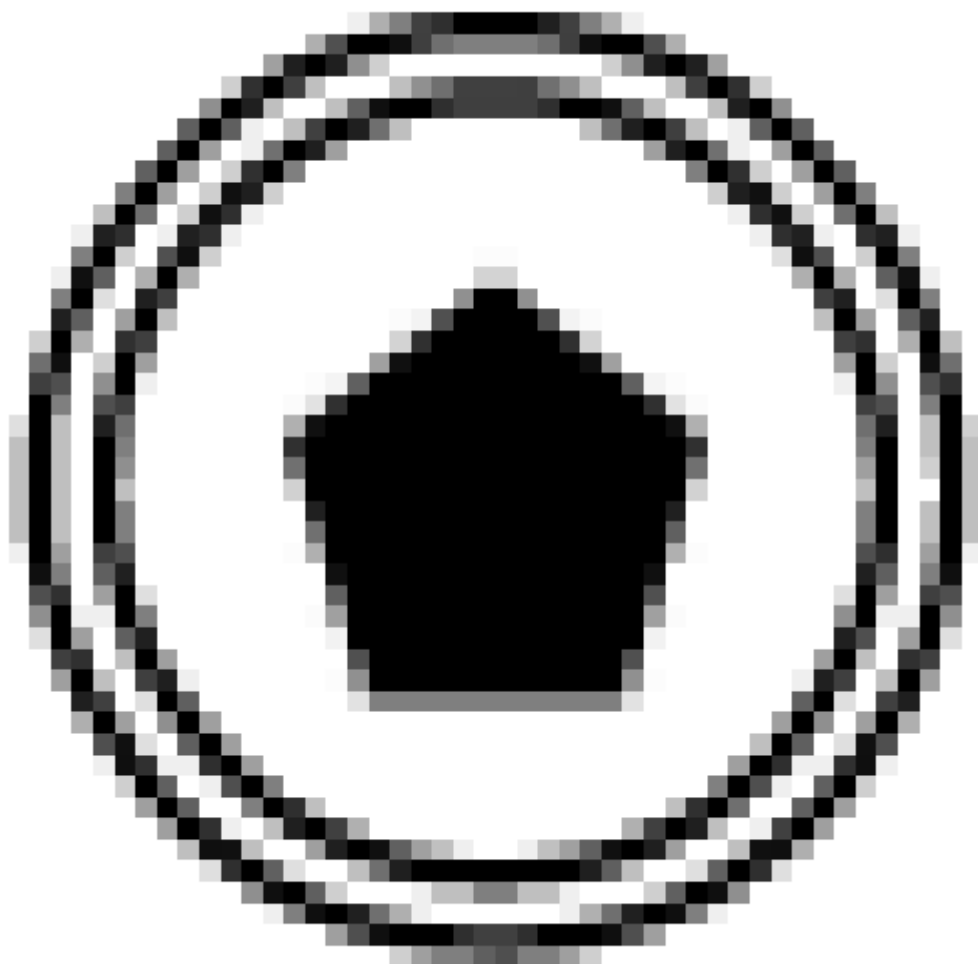




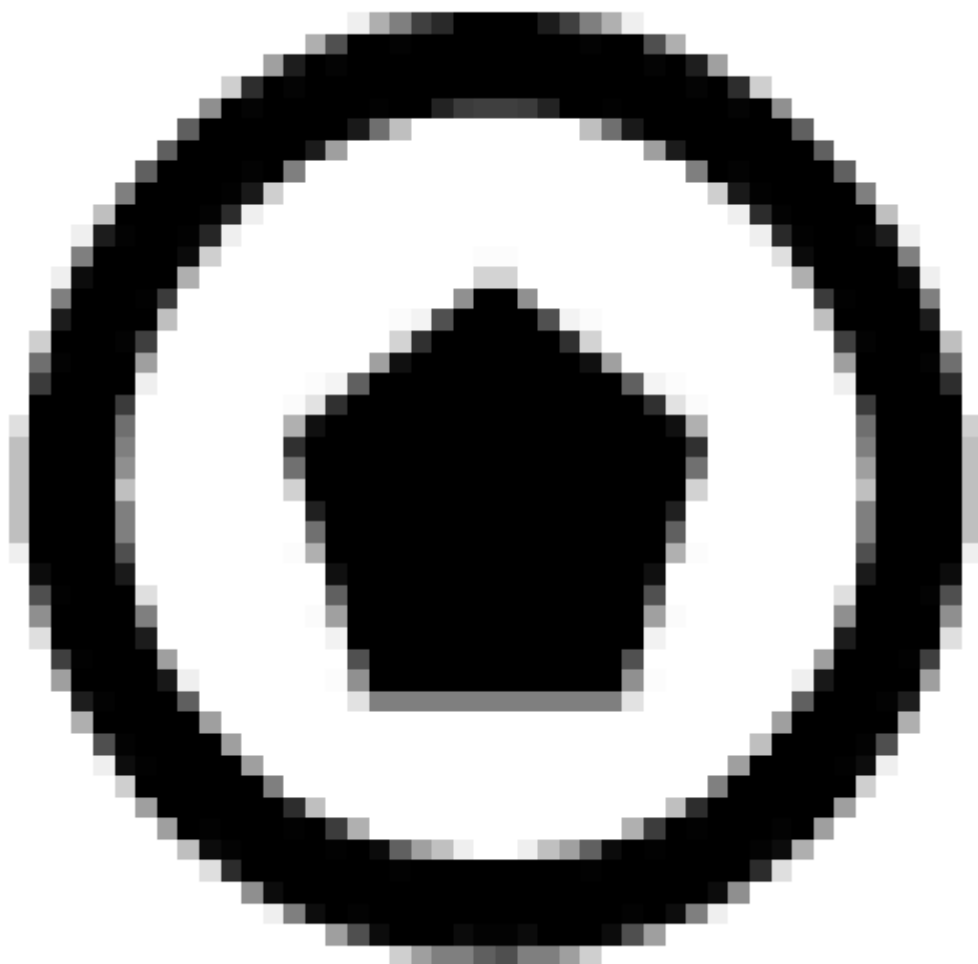
|



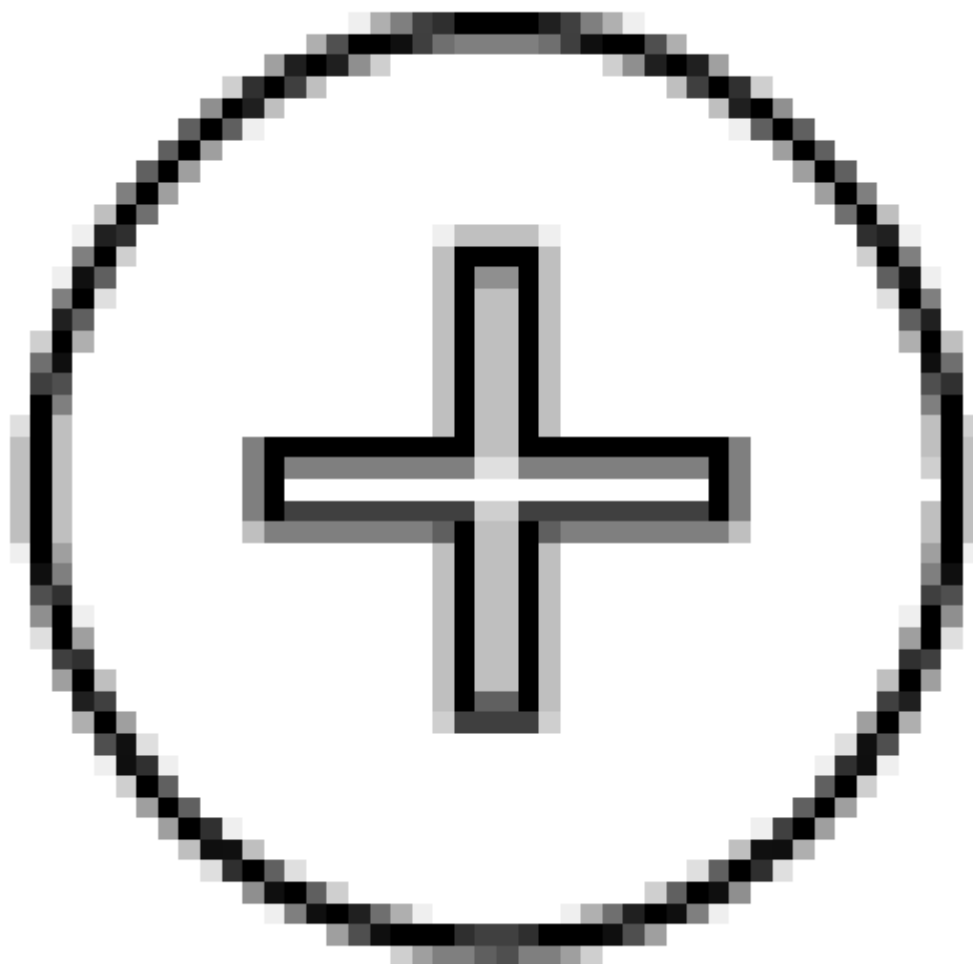
|



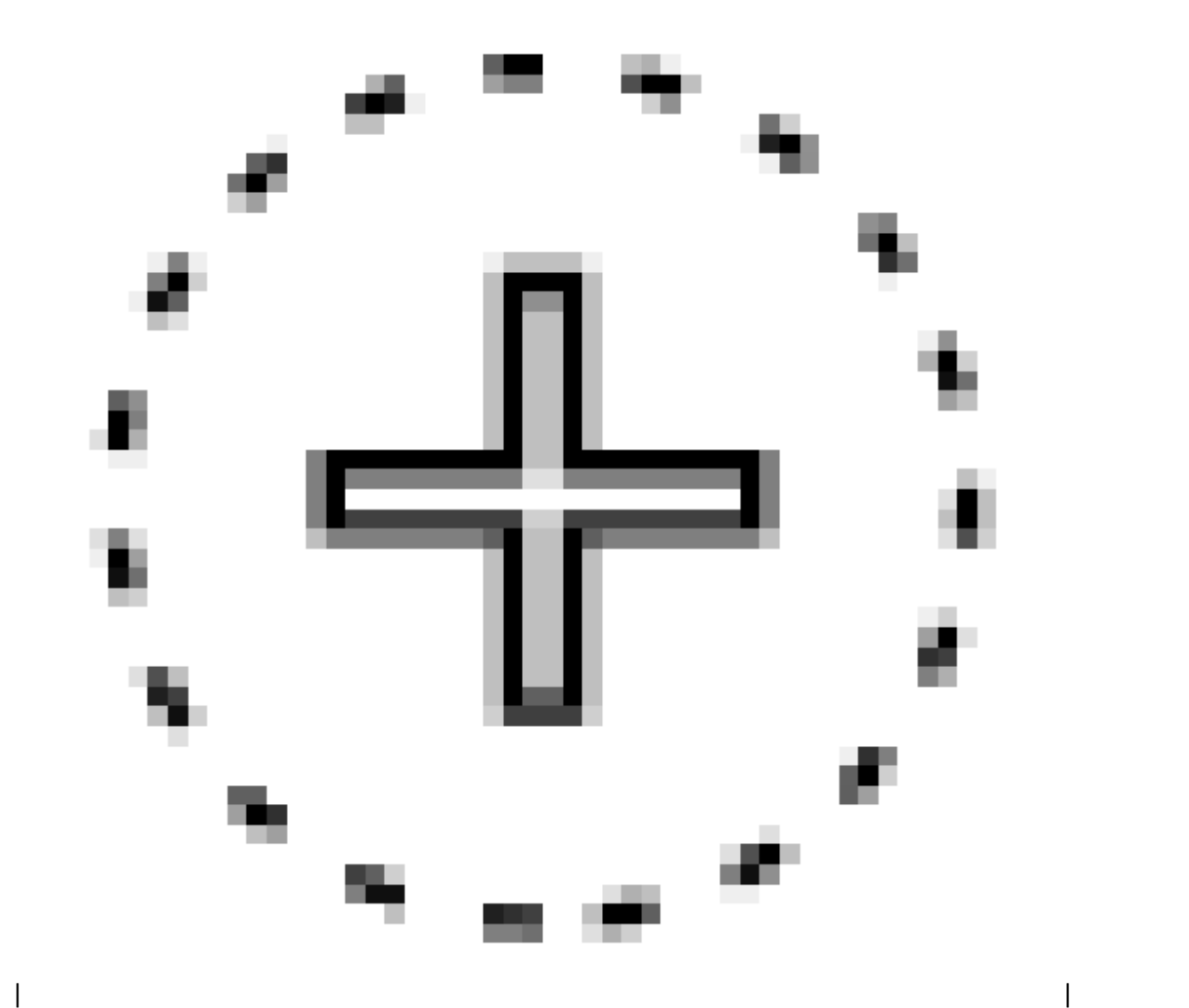
|



|



| Parallel |





|



|||

Gateway

Gateway is used to control the flow of a process and it is represented as a diamond shape. To create a gateway, the shape property of the node should be set as [gateway](#) and the gateway property can be set with any of the appropriate gateways. The following code example illustrates how to create a BPMN Gateway.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
```

```

        height: 100,
        //Sets type as Bpmn and shape as Gateway
        shape: {
            type: 'Bpmn',
            shape: 'Gateway',
            //Sets type of the gateway as None
            gateway: {
                type: 'None',
            }
        },
    },
    ]];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}>
        <Inject services={[BpmnDiagrams]} />
        </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Gateway
    shape: {
        type: 'Bpmn',
        shape: 'Gateway',
        //Sets type of the gateway as None
        gateway: {
            type: 'None',
        }
    },
},
];

```



```
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

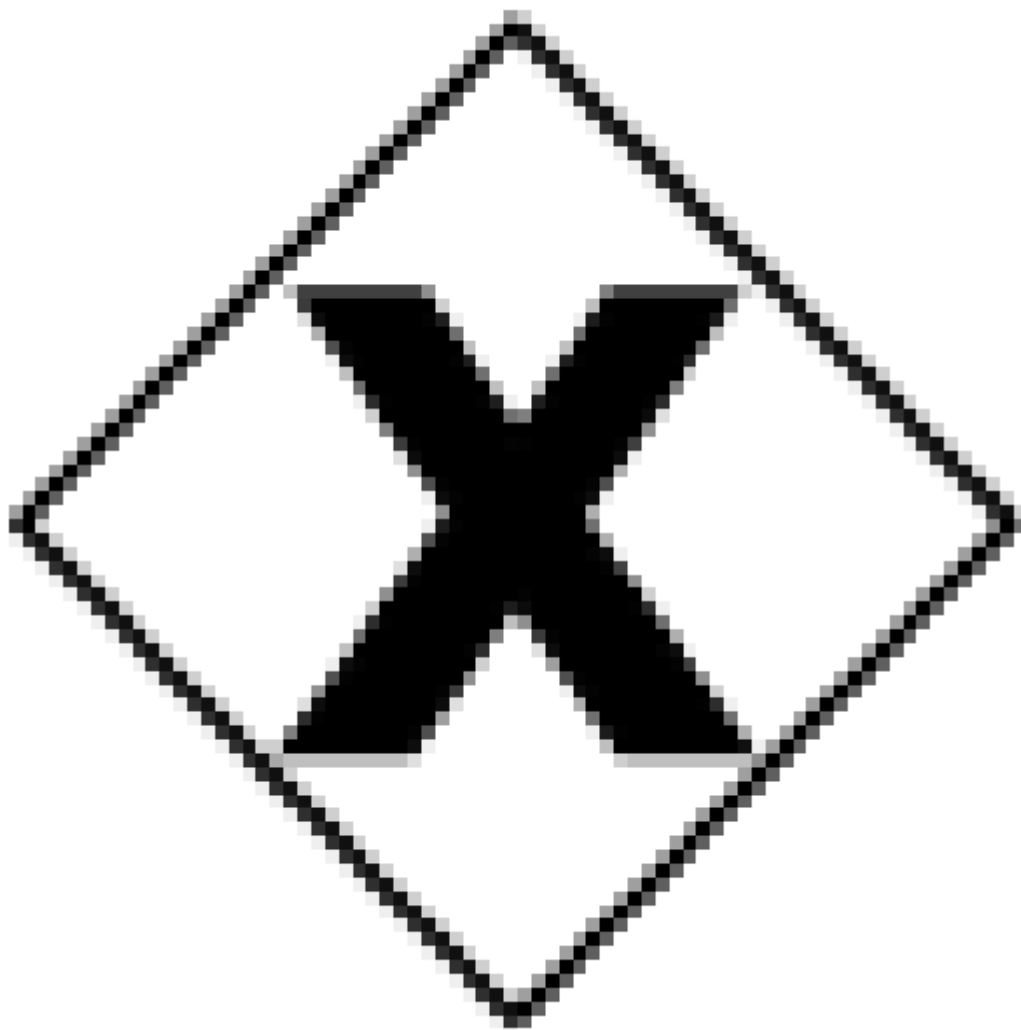
Note: By default, the **gateway** will be set as **none**.

There are several types of gateways as tabulated:

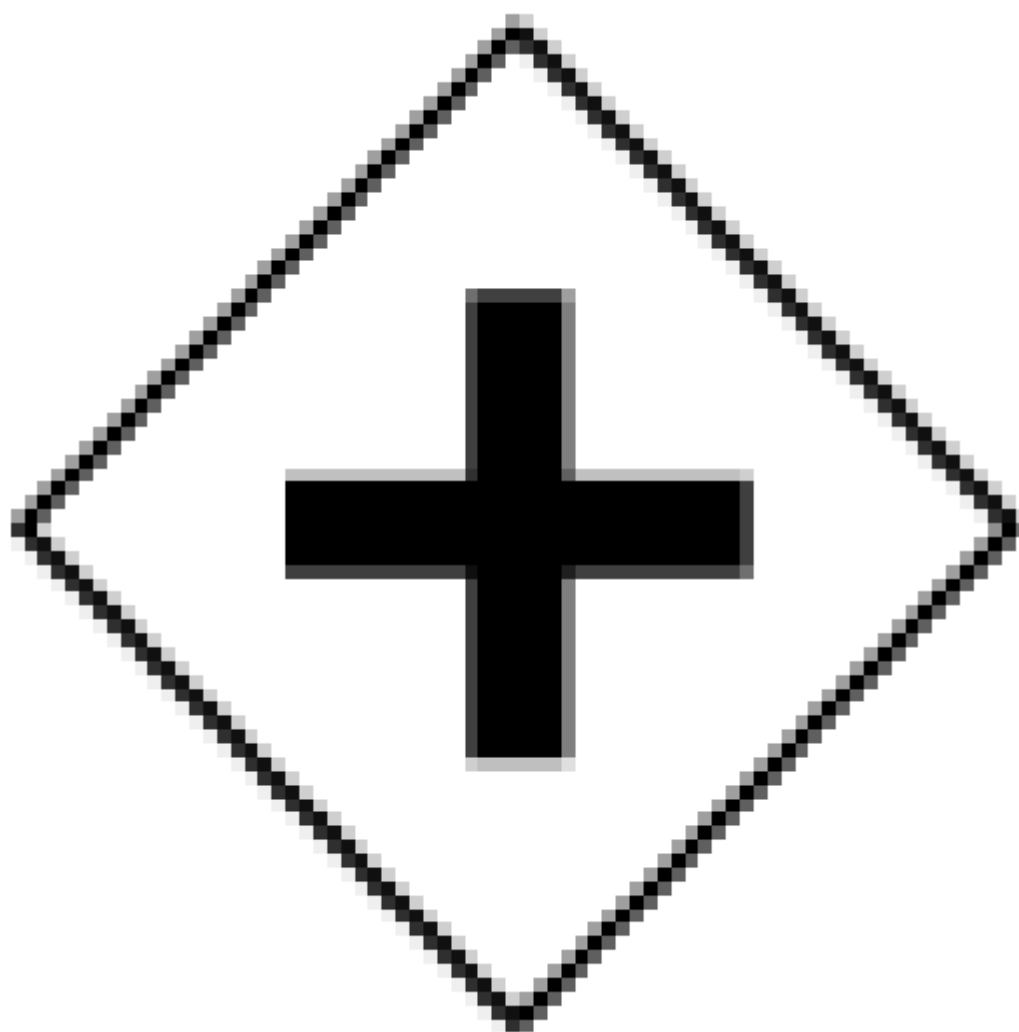
Shape	Image
-------	-------

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

| Exclusive |

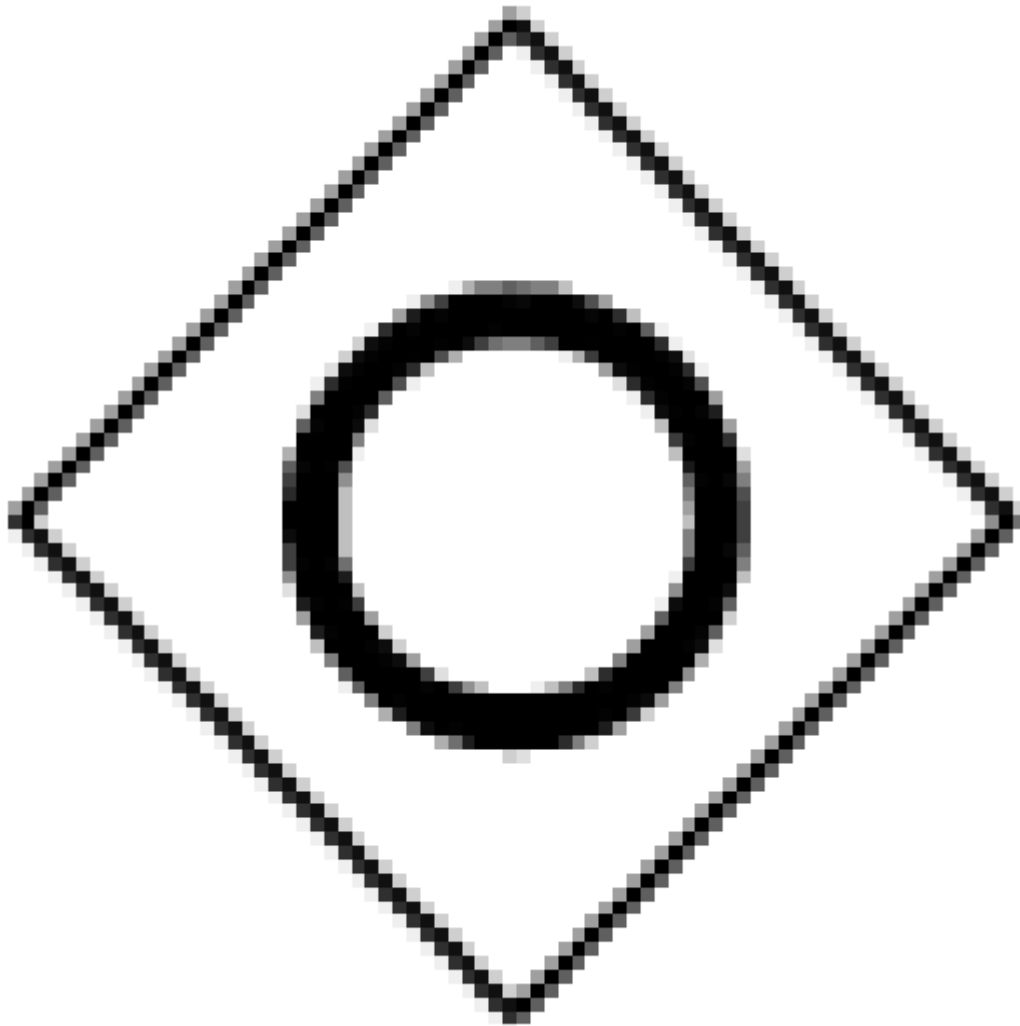


|



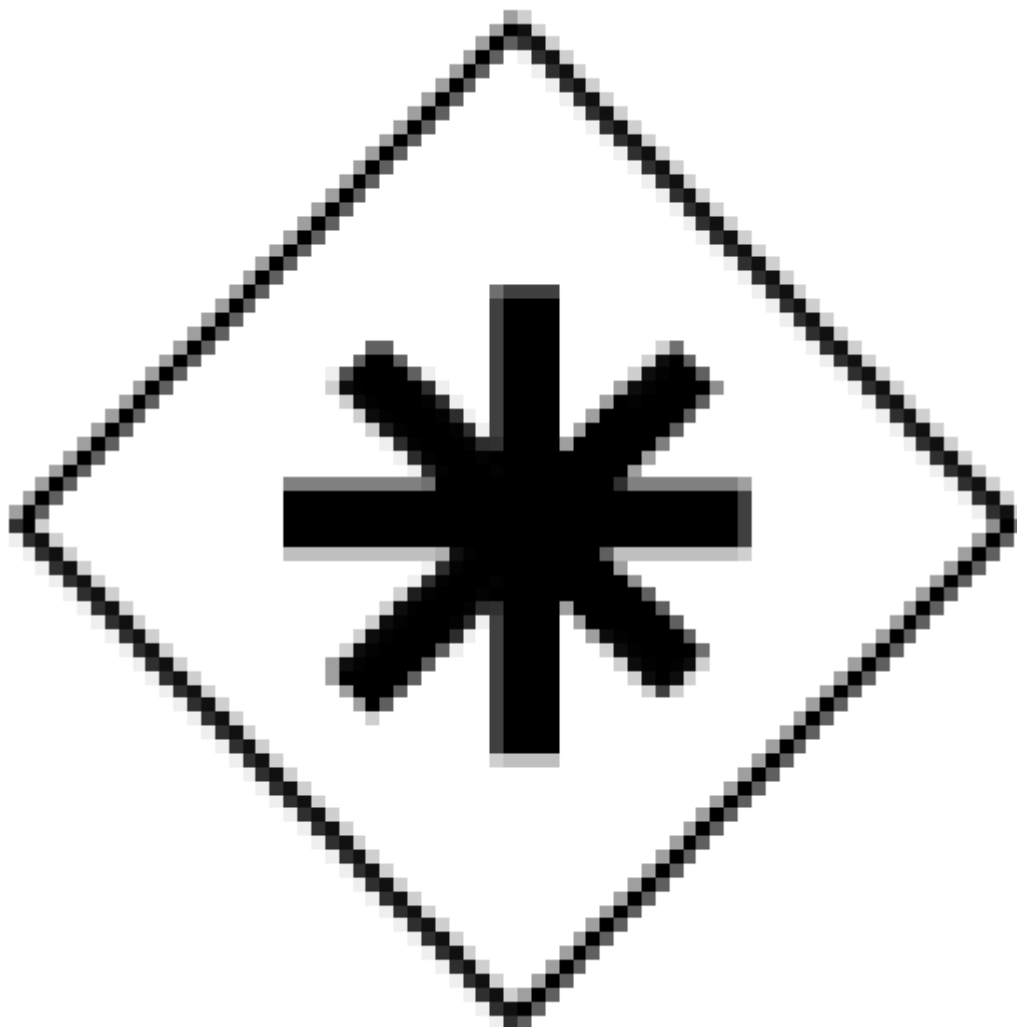
| Parallel |
|

| Inclusive |



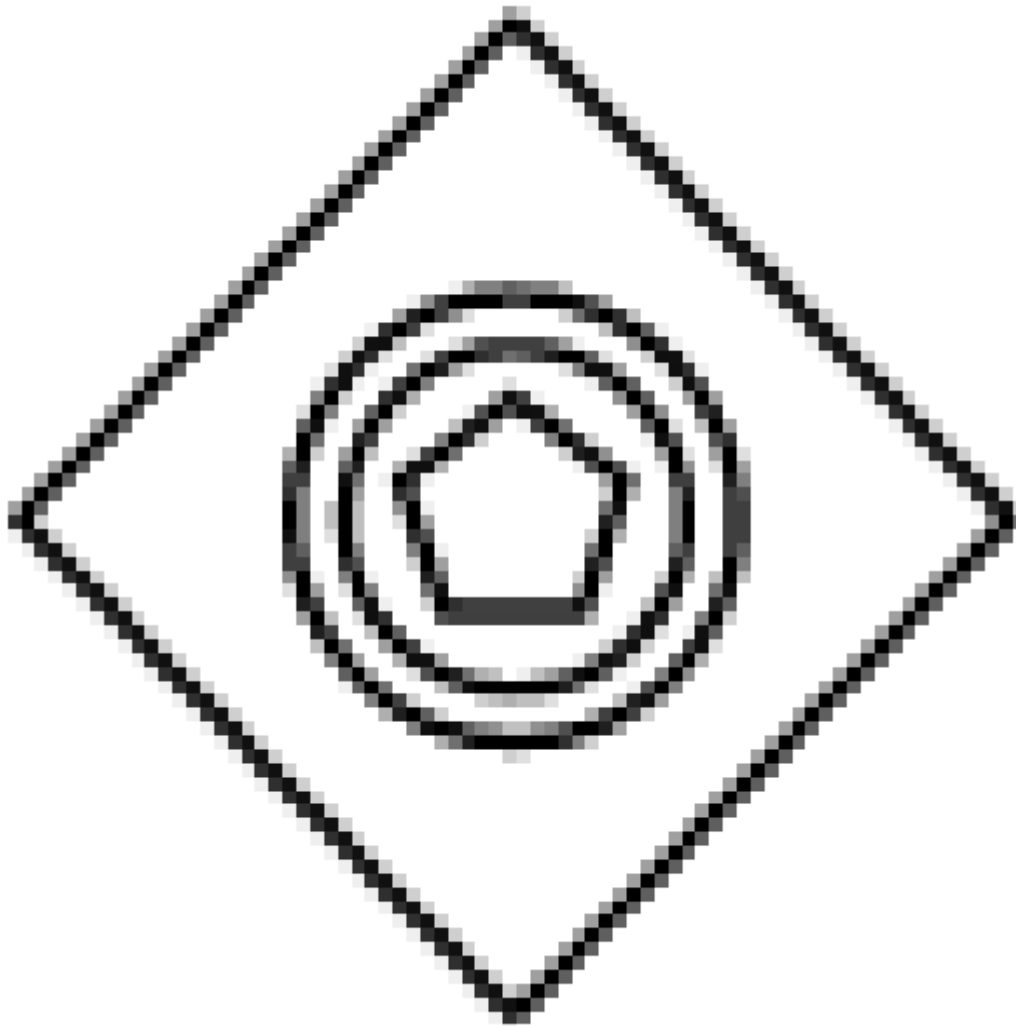
|

| Complex |



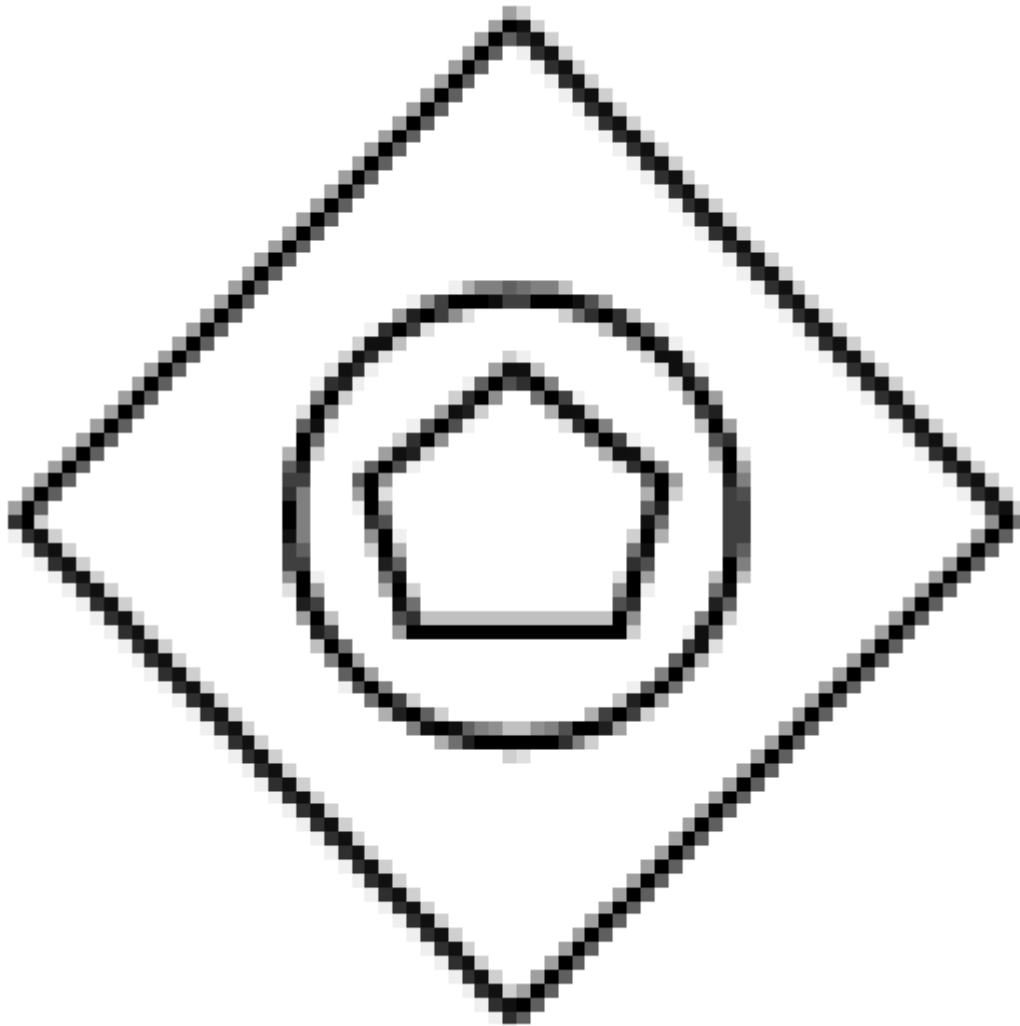
|

| EventBased |



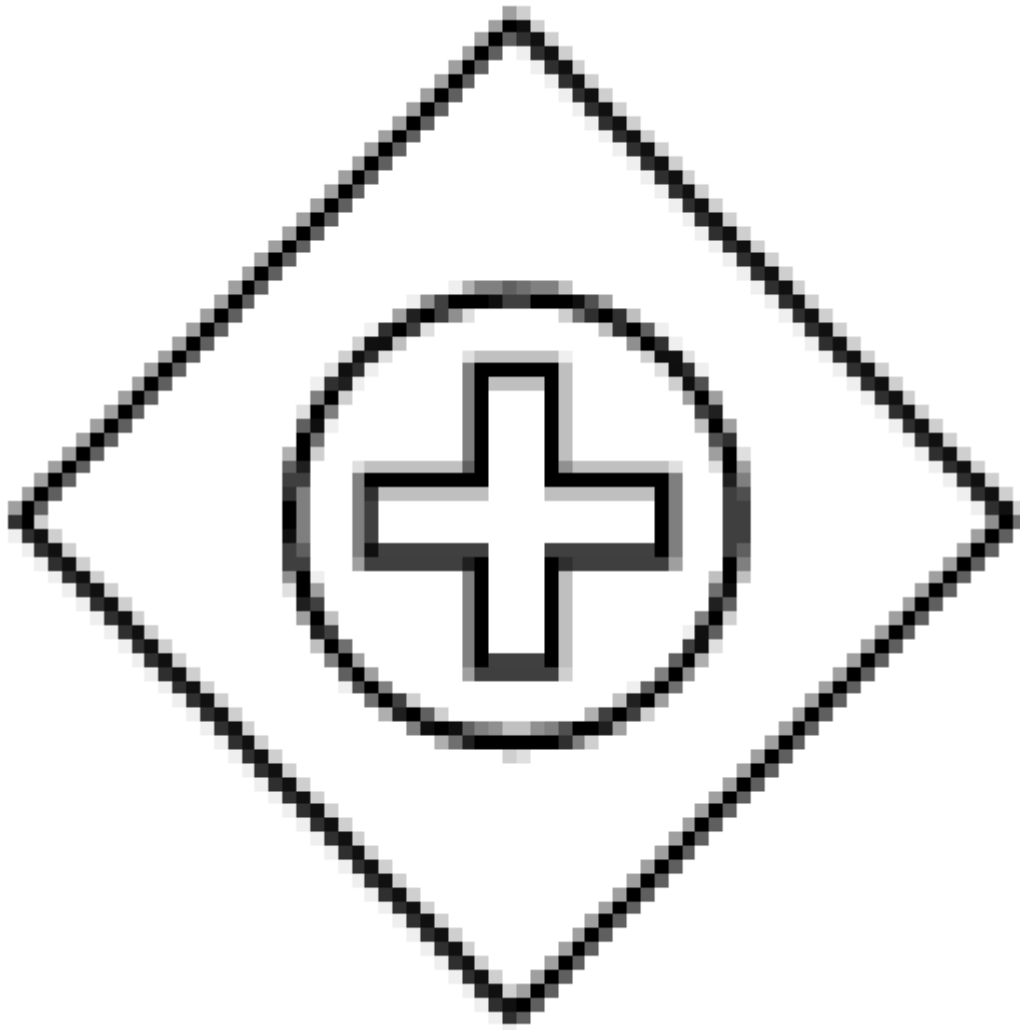
|

| ExclusiveEventBased |



|

| ParallelEventBased |



Activity

The [activity](#) is the task that is performed in a business process. It is represented by a rounded rectangle.

There are two types of activities. They are listed as follows:

- Task: Occurs within a process and it is not broken down to a finer level of detail.
- Subprocess: Occurs within a process and it is broken down to a finer level of detail.

To create a BPMN activity, set the shape as **activity**. You also need to set the type of the BPMN activity by using the activity property of the node. By default, the type of the activity is set as **task**. The following code example illustrates how to create an activity.

INDEX.JSX

```
import * as React from "react";  
import * as ReactDOM from "react-dom";
```



```

import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as Task
        activity: {
            activity: 'Task'
        },
    },
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
        <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity

```

```

    shape: {
      type: 'Bpmn',
      shape: 'Activity',
      //Sets activity as Task
      activity: {
        activity: 'Task'
      },
    },
  },
  });
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The different activities of BPMN process are listed as follows.

Tasks

The [task](#) property of the node allows you to define the type of task such as sending, receiving, user based task, etc. By

default, the type property of task is set as **none**. The following code illustrates how to create different types of

BPMN tasks.

The [type](#) property of tasks allows to represent these results as an event attached to the task..

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
  },
}];

```

```

        //Sets activity as Task
        activity: {
            activity: 'Task',
            //Sets the type of the task as Send
            task: {
                type: 'Send'
            }
        },
    },
    });
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}>
        <Inject services={[BpmnDiagrams]}/>
        </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as Task
        activity: {
            activity: 'Task',
            //Sets the type of the task as Send
            task: {
                type: 'Send'
            }
        }
    }
}];

```

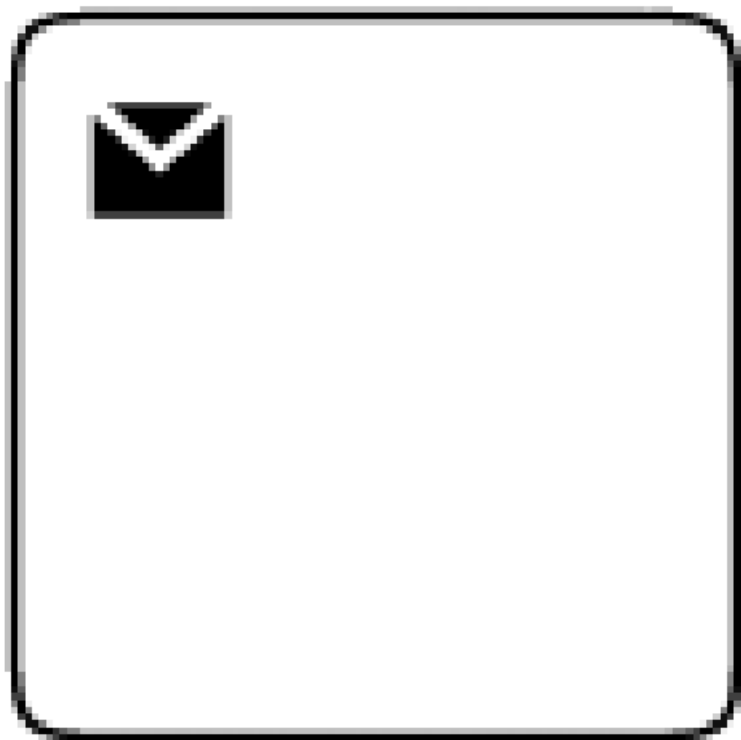
```
    },
  },
  });
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

The various types of BPMN tasks are tabulated as follows.

Shape	Image
-----	-----



Service	
---------	--



| Send |

|



| Receive |

|



| Instantiating Receive |

|





| Business Rule |

|



| User |

|



| Script |

Subprocess

A [sub-process](#) is a group of tasks, which is used to hide or reveal details of additional levels using the [collapsed](#) property.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess and collapsed of subprocess as
    true
    activity: {
      activity: 'SubProcess',
      subprocess: {
```

```

        collapsed: true
      }
    },
  ],
};
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess and collapsed of subprocess as true
    activity: {
      activity: 'SubProcess',
      subprocess: {
        collapsed: true
      }
    }
  }
},
];
// initialize diagram component
function App() {
  return (

```

```

<DiagramComponent
  id="container"
  width={'100%'}
  height={'600px'}
  // Add node
  nodes={node}
>
  <Inject services={[BpmnDiagrams]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The different types of subprocess are as follows:

- Event subprocess
- Transaction

Event subprocess

A subprocess is defined as an event subprocess, when it is triggered by an event. An event subprocess is placed within another subprocess which is not part of the normal flow of its parent process. You can set event to a subprocess with the [event](#) and [trigger](#) property of the subprocess. The [type](#) property of subprocess allows you to define the type of subprocess whether it should be event subprocess or transaction subprocess.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets the collapsed as true and type as Event
      subprocess: {
        collapsed: true,
        type: 'Event',
        //Sets event as Start and trigger as Message
        events: [{
          event: 'Start',

```

```

        trigger: 'Message'
      }
    ]
  },
  },
  },
  });
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
      <Inject services={[BpmnDiagrams]} />
      </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets the collapsed as true and type as Event
      subprocess: {
        collapsed: true,
        type: 'Event',
        //Sets event as Start and trigger as Message
        events: [{
          event: 'Start',
          trigger: 'Message'
        }
      ]
    }
  }
}];

```

```

    }
  },
},
});
// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Transaction subprocess

- [transaction](#) is a set of activities that logically belong together, in which all contained activities must complete their parts of the transaction; otherwise the process is undone. The execution result of a transaction is one of Successful Completion, Unsuccessful Completion (Cancel), and Hazard (Exception). The [events](#) property of subprocess allows to represent these results as an event attached to the subprocess.
- The event object allows you to define the type of event by which the subprocess will be triggered. The name of the event can be defined to identify the event at runtime.
- The event's offset property is used to set the fraction/ratio (relative to parent) that defines the position of the event shape.
- The trigger property defines the type of the event trigger.
- You can also use define ports and labels to subprocess events by using event's ports and labels properties.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {

```

```

        type: 'Bpmn', shape: 'Activity', activity: {
          //Sets activity as SubProcess
          activity: 'SubProcess',
          subProcess: {
            //Sets collapsed as true and type as Transition
            collapsed: true,
            type: 'Transaction',
            events: [{
              event: 'Intermediate',
              trigger: 'Cancel',
              offset: {
                x: 0.25,
                y: 1
              }
            },
            {
              event: 'Intermediate',
              trigger: 'Error',
              offset: {
                x: 0.25,
                y: 1
              }
            }
          ],
          //processes: ['start', 'end', 'nod1', 'nod']
        },
      ],
    },
  ],
};

// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
    <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";

```



```

// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn', shape: 'Activity', activity: {
      //Sets activity as SubProcess
      activity: 'SubProcess',
      subprocess: {
        //Sets collapsed as true and type as Transition
        collapsed: true,
        type: 'Transaction',
        events: [{
          event: 'Intermediate',
          trigger: 'Cancel',
          offset: {
            x: 0.25,
            y: 1
          }
        },
        {
          event: 'Intermediate',
          trigger: 'Error',
          offset: {
            x: 0.25,
            y: 1
          }
        }
      ],
      //processes: ['start', 'end', 'nod1', 'nod']
    } as BpmnSubProcessModel
  },
},
];
// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Process

Processes is an array collection that defines the children values for BPMN subprocess.

Loop

[Loop](#) is a task that is internally being looped. The loop property of task allows you to define the type of loop. The default value for `loop` is **none**.

You can define the loop property in subprocess BPMN shape as shown in the following code.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as Task
    activity: {
      activity: 'Task',
      //Sets loop of the task as Standard
      task: {
        loop: 'Standard'
      }
    }
  },
},
{
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets Activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets collapsed as true and loop as Standard
      subProcess: {
        collapsed: true,
        loop: 'Standard'
      }
    }
  },
},
];
```

```

    },
  }];
  // initialize diagram component
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
      // Add node
      nodes={node}>
        <Inject services={[BpmnDiagrams]} />
      </DiagramComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as Task
    activity: {
      activity: 'Task',
      //Sets loop of the task as Standard
      task: {
        loop: 'Standard'
      }
    }
  },
},
],
{
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node

```

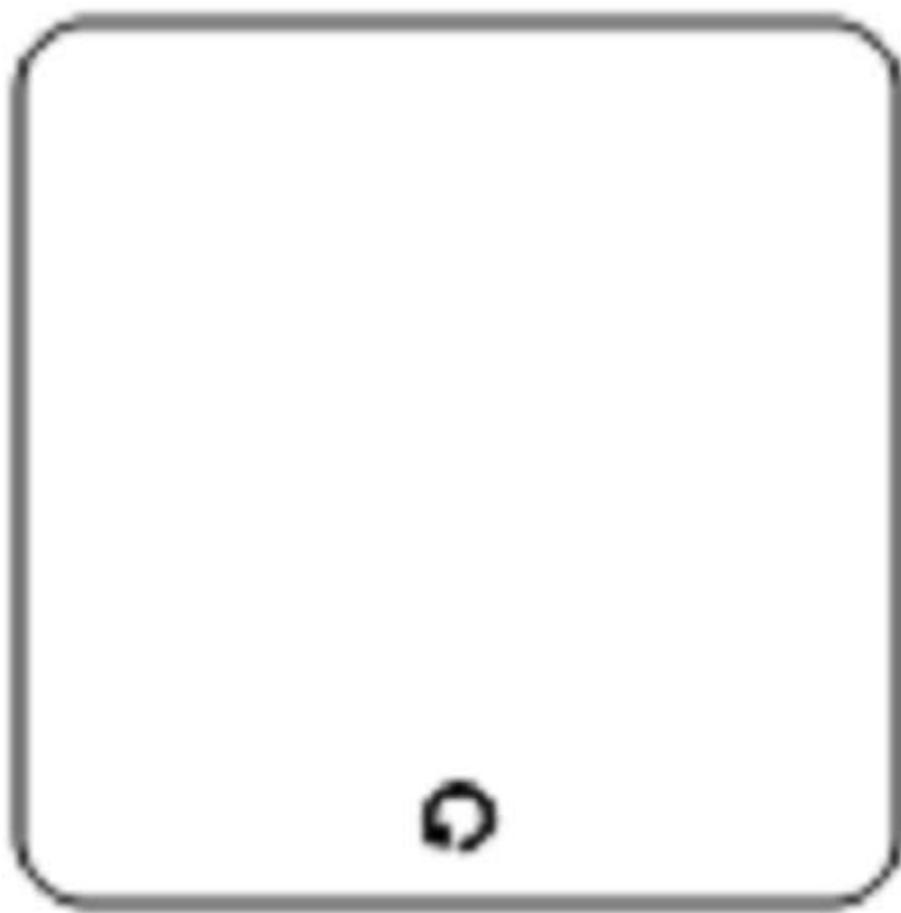
```

width: 100,
height: 100,
//Sets type as Bpmn and shape as activity
shape: {
  type: 'Bpmn',
  shape: 'Activity',
  //Sets Activity as SubProcess
  activity: {
    activity: 'SubProcess',
    //Sets collapsed as true and loop as Standard
    subprocess: {
      collapsed: true,
      loop: 'Standard'
    },
  },
},
}],
// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

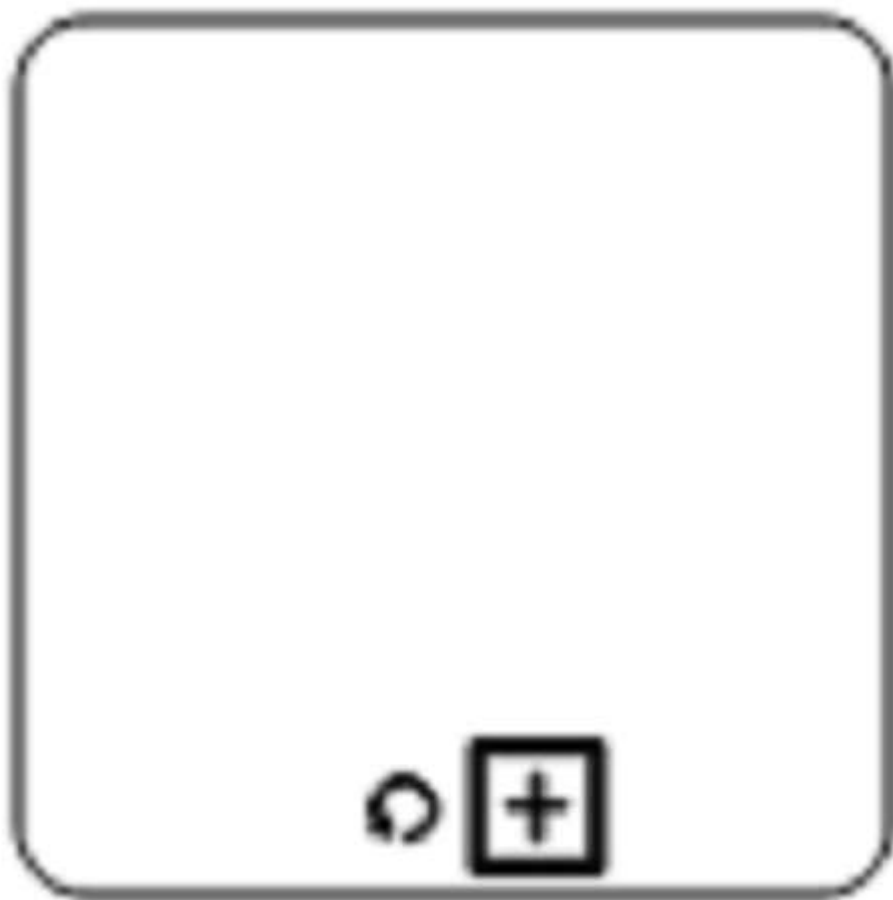
```

The following table contains various types of BPMN loops.

Loops	Task	Subprocess
-----	-----	-----



| Standard |



| SequenceMultiInstance |



|

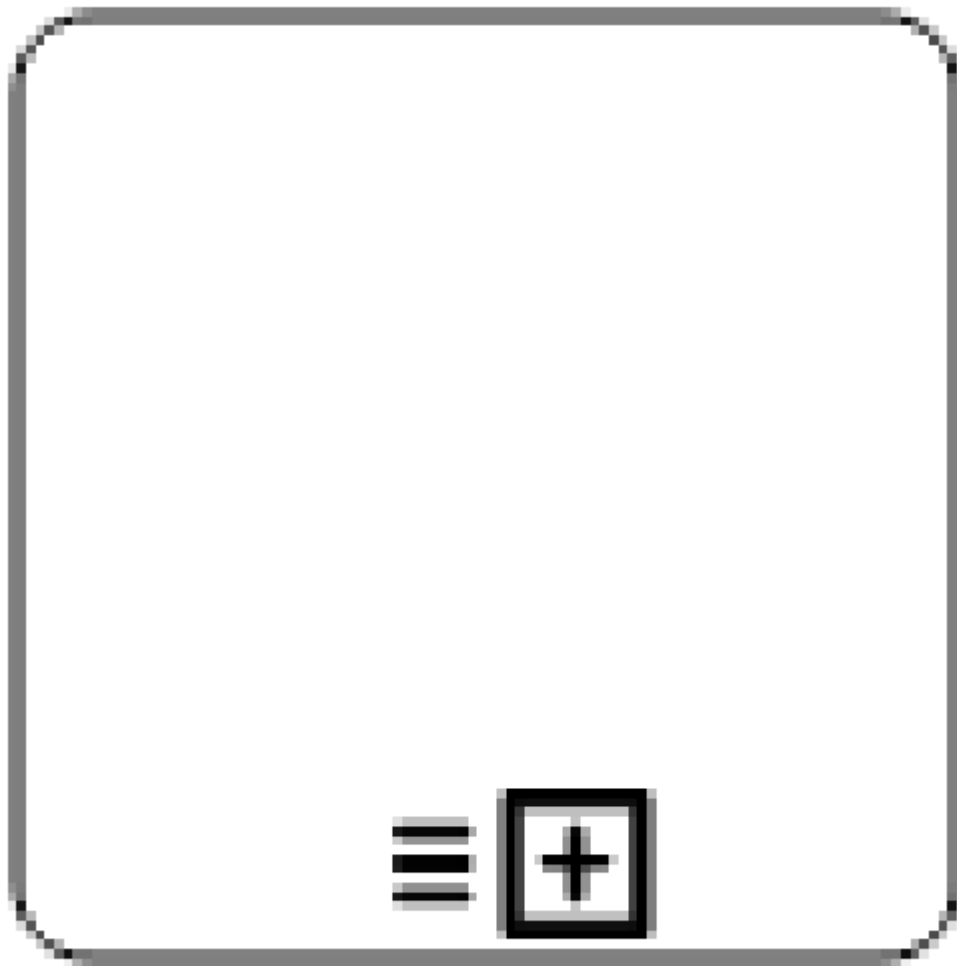


|

| ParallelMultiInstance |



|



Compensation

[Compensation](#) is triggered, when operation is partially failed and enabled it with the compensation property of the task and the subprocess.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
```

```

    shape: {
      type: 'Bpmn',
      shape: 'Activity',
      //Sets activity as Task
      activity: {
        activity: 'Task',
        //Sets compensation of the task as true
        task: {
          compensation: true,
        }
      },
    }
  }, {
    // Position of the node
    offsetX: 300,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
      type: 'Bpmn',
      shape: 'Activity',
      //Sets activity as SubProcess
      activity: {
        activity: 'SubProcess',
        //Set the collapsed as true and compensation as true
        subProcess: {
          collapsed: true,
          compensation: true,
        }
      },
    },
  },
];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
    <Inject services={[BpmnDiagrams]}/>
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,

```

```

    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
  } from "@syncfusion/ej2-react-diagrams";
  // A node is created and stored in nodes array.
  let node: NodeModel[] = [{
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
      type: 'Bpmn',
      shape: 'Activity',
      //Sets activity as Task
      activity: {
        activity: 'Task',
        //Sets compensation of the task as true
        task: {
          compensation: true,
        }
      },
    },
  }, {
    // Position of the node
    offsetX: 300,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
      type: 'Bpmn',
      shape: 'Activity',
      //Sets activity as SubProcess
      activity: {
        activity: 'SubProcess',
        //Set the collapsed as true and compensation as true
        subProcess: {
          collapsed: true,
          compensation: true,
        },
      },
    },
  },
  ]];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        // Add node
        nodes={node}

```

```

    >
    <Inject services={[BpmnDiagrams]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Call

A [call](#) activity is a global subprocess that is reused at various points of the business flow and set it with the call property of the task.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets the activity as task
    activity: {
      activity: 'Task',
      //Sets the call of the task as true
      task: {
        call: true
      }
    }
  },
},
];
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
    <Inject services={[BpmnDiagrams]} />
  </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";

```

```

import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets the activity as task
    activity: {
      activity: 'Task',
      //Sets the call of the task as true
      task: {
        call: true
      }
    }
  },
},
];
// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Adhoc

An adhoc subprocess is a group of tasks that are executed in any order or skipped in order to fulfill the end condition and set it with the [adhoc](#) property of subprocess.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
        //Sets activity as SubProcess
        activity: {
            activity: 'SubProcess',
            //Sets collapsed as true and adhoc as true
            subprocess: {
                collapsed: true,
                adhoc: true
            }
        }
    },
},
];
// initialize diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
        <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";

```

```
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as Activity
  shape: {
    type: 'Bpmn',
    shape: 'Activity',
    //Sets activity as SubProcess
    activity: {
      activity: 'SubProcess',
      //Sets collapsed as true and adhoc as true
      subProcess: {
        collapsed: true,
        adhoc: true
      }
    }
  },
},
];
// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Boundary

Boundary represents the type of task that is being processed. The [boundary](#) property of subprocess allows you to define the type of boundary. By default, it is set as **default**.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
```



```

        width: 100,
        height: 100,
        //Sets type as Bpmn and shape as Activity
        shape: {
            type: 'Bpmn',
            shape: 'Activity',
            //Sets activity as SubProcess
            activity: {
                activity: 'SubProcess',
                //Sets collapsed as true and boundary as Call
                subprocess: {
                    collapsed: true,
                    boundary: 'Call'
                }
            },
        },
    });
    // initialize diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
            // Add node
            nodes={node}>
                <Inject services={[BpmnDiagrams]} />
            </DiagramComponent>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Activity
    shape: {
        type: 'Bpmn',
        shape: 'Activity',
    },
}];

```

```

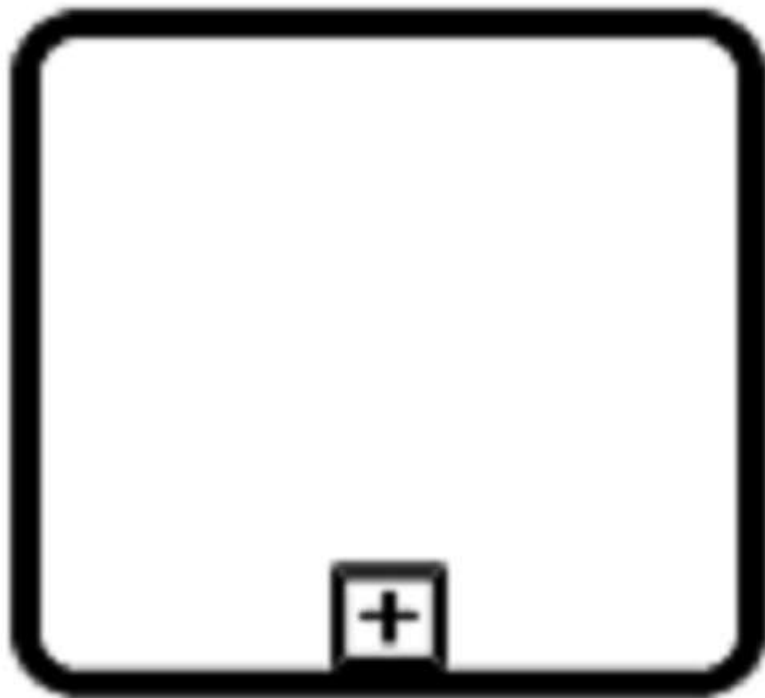
        //Sets activity as SubProcess
        activity: {
            activity: 'SubProcess',
            //Sets collapsed as true and boundary as Call
            subprocess: {
                collapsed: true,
                boundary: 'Call'
            }
        },
    ],
    });
    // initialize diagram component
    function App() {
        return (
            <DiagramComponent
                id="container"
                width={'100%'}
                height={'600px'}
                // Add node
                nodes={node}
            >
                <Inject services={[BpmnDiagrams]} />
            </DiagramComponent>
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

The following table contains various types of BPMN boundaries.

| Boundary | Image |

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



| Call |

|



| Event |

|



| Default |

Data

A data object represents information flowing through the process, such as data placed into the process, data resulting from the process, data that needs to be collected, or data that must be stored. To define a [data object](#), set the shape as **DataObject** and the type property defines whether data is an input or an output. You can create multiple instances of data object with the collection property of data.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as DataObject
  shape: {
    type: 'Bpmn',
    shape: 'DataObject',
    //Sets collection as true and type as Input
    dataObject: {
```

```

        collection: true,
        type: 'Input'
      }
    }
  }];
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}>
      <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

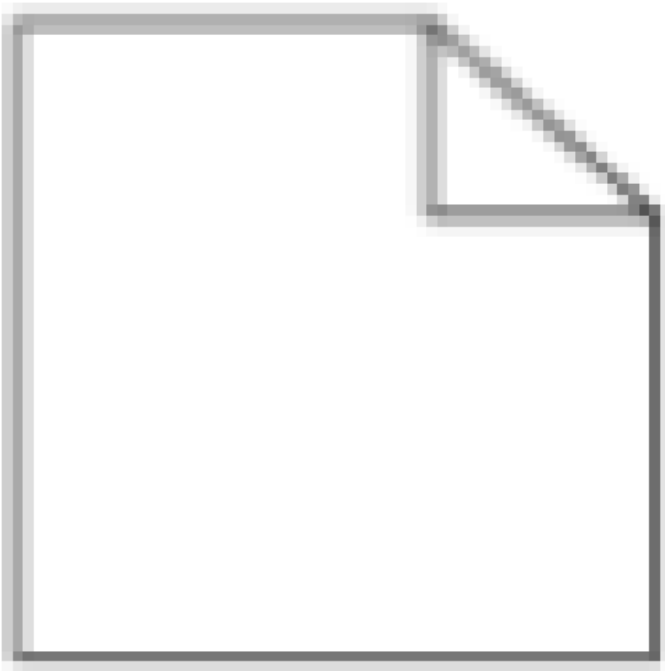
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnShapeModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as DataObject
  shape: {
    type: 'Bpmn',
    shape: 'DataObject',
    //Sets collection as true and type as Input
    dataObject: {
      collection: true,
      type: 'Input'
    }
  }
}];
// initialize diagram component
function App() {
  return (
    <DiagramComponent

```

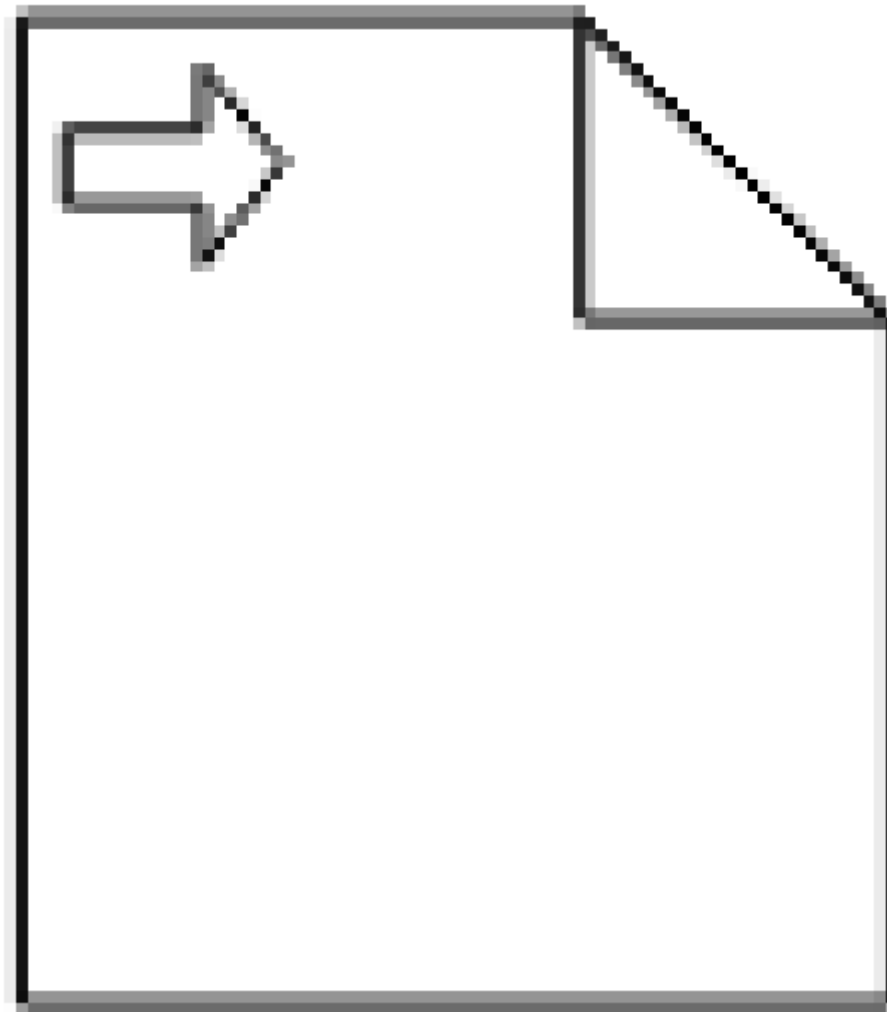
```
id="container"
width={ '100%' }
height={ '600px' }
// Add node
nodes={node}
>
  <Inject services={[BpmnDiagrams]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

The following table contains various representation of BPMN data object.

Boundary	Image
-----	-----

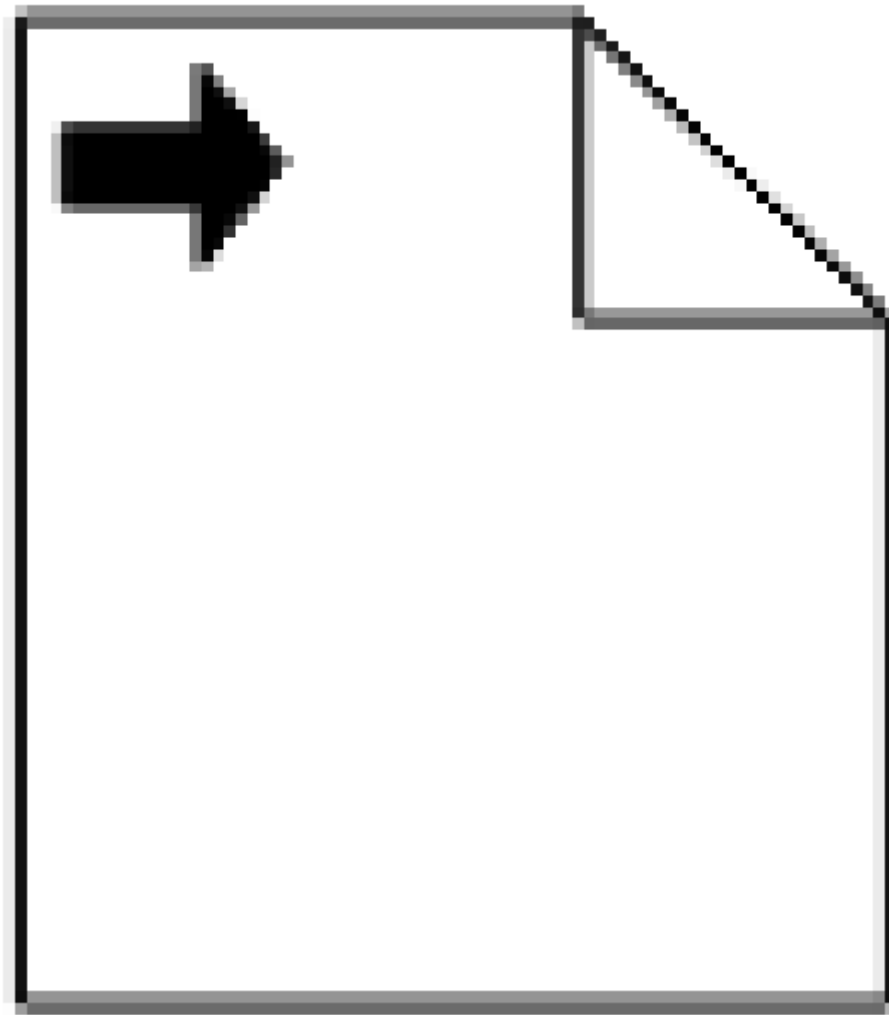


Collection Data Object



| Data Input |

|



| Data Output |

Datasource

Datasource is used to store or access data associated with a business process. To create a datasource, set the shape as **datasource**. The following code example illustrates how to create a datasource.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  //Sets type as Bpmn and shape as DataSource
```

```

        shape: {
            type: 'Bpmn',
            shape: 'DataSource',
        }
    }];
    // initialize diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
            // Add node
            nodes={node}>
            <Inject services={[BpmnDiagrams]} />
            </DiagramComponent>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as DataSource
    shape: {
        type: 'Bpmn',
        shape: 'DataSource',
    }
}];
// initialize diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            // Add node
            nodes={node}
        >

```

```

    <Inject services={[BpmnDiagrams]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Artifact

Artifact is used to show additional information about a process in order to make it easier to understand. There are two types of artifacts in BPMN.

- Text annotation
- Group

Text annotation

- A BPMN object can be associated with a text annotation which does not affect the flow but gives details about objects within a flow.
- A TextAnnotation points to or references another BPMN shape, which we call the `textAnnotationTarget` of the textAnnotation. When a target shape is moved or deleted, any TextAnnotations attached to the shape will be moved or deleted too. Thus, the TextAnnotations remain with their target shapes though you can reposition the TextAnnotation to any offset from its target. The `textAnnotationTarget` property of the `BpmnTextAnnotation` is used to connect an annotation element to the BPMN Node.
- The annotation element can be switched from a BPMN node to another BPMN node simply by dragging the source end of the annotation connector into the other BPMN node.
- By default, the TextAnnotation shape has a connection.
- The `textAnnotationDirection` property is used to set the shape direction of the text annotation.
- By default, the `textAnnotationDirection` is set to a Auto.
- To set the size for text annotation, use the `width` and `height` properties of the node.
- The `offsetX` and `offsetY` properties are used to set the distance between the BPMN node and the TextAnnotation.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let nodes = [
  {
    id: 'event1', style: { strokeWidth: 2 },
    height: 70, width: 70, offsetX: 400, offsetY: 200,
    shape: { type: 'Bpmn', shape: 'Event',
      event: { event: 'Start', trigger: 'None' },
    }
  },
];
//node with target

```

```

    {
      id: 'textNode1', width: 70, height: 70,
      offsetX: 400, offsetY: 400,
      annotations: [{content: 'textNode1'}],
      shape: {
        type: 'Bpmn', shape: 'TextAnnotation',
        textAnnotation: {
          textAnnotationDirection: 'Auto', textAnnotationTarget: 'event1'
        }
      },
    },
    //Node without target
    {
      id: 'textNode2', width: 70, height: 70,
      offsetX: 600, offsetY: 400,
      annotations: [{content: 'textNode1'}],
      shape: {
        type: 'Bpmn', shape: 'TextAnnotation',
        textAnnotation: {
          textAnnotationDirection: 'Auto', textAnnotationTarget: ''
        }
      },
    },
  ],
  // initialize diagram component
  function App() {
    return (
      <DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={nodes}>
        <Inject services={[BpmnDiagrams]} />
      </DiagramComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnShapeModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [
  {
    id: 'event1', style: { strokeWidth: 2 },
    height: 70, width: 70, offsetX: 400, offsetY: 200,
    shape: {
      type: 'Bpmn', shape: 'Event',

```

```

        event: { event: 'Start', trigger: 'None' },
      },
      //node with target
      {
        id: 'textNode1', width: 70, height: 70,
        offsetX:400,offsetY:400,
        annotations:[{content:'textNode1'}],
        shape: {
          type: 'Bpmn', shape: 'TextAnnotation',
          textAnnotation:{
            textAnnotationDirection:'Auto',textAnnotationTarget:'event1'}
        },
      },
      //Node without target
      {
        id: 'textNode2', width: 70, height: 70,
        offsetX:600,offsetY:400,
        annotations:[{content:'textNode1'}],
        shape: {
          type: 'Bpmn', shape: 'TextAnnotation',
          textAnnotation:{
            textAnnotationDirection:'Auto',textAnnotationTarget:''}
        },
      },
    ],
  ],
  // initialize diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        // Add node
        nodes={nodes}
      >
        <Inject services={[BpmnDiagrams]} />
      </DiagramComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Group

A group is used to frame a part of the diagram, shows that elements included in it are logically belong together and does not have any other semantics other than organizing elements. To create a group, the shape property of the node should be set as **group**. The following code example illustrates how to create a BPMN group.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{

```

```

        // Position of the node
        offsetX: 250,
        offsetY: 250,
        // Size of the node
        width: 100,
        height: 100,
        //Sets type as Bpmn and shape as Group
        shape: {
            type: 'Bpmn',
            shape: 'Group',
        }
    }
    }];
// initialize diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}>
            <Inject services={[BpmnDiagrams]} />
        </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    NodeModel,
    BpmnShape,
    BpmnSubProcessModel,
    BpmnShapeModel,
    BpmnDiagrams,
    BpmnActivityModel,
    BpmnFlowModel,
    BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    //Sets type as Bpmn and shape as Group
    shape: {
        type: 'Bpmn',
        shape: 'Group',
    }
}];
// initialize diagram component
function App() {

```

```

return (
  <DiagramComponent
    id="container"
    width={'100%'}
    height={'600px'}
    // Add node
    nodes={node}
  >
    <Inject services={[BpmnDiagrams]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

BPMN flows

[BPMN Flows](#) are lines that connects BPMN flow objects.

Association

[BPMN Association](#) flow is used to link flow objects with its corresponding text or artifact. An association is represented as a dotted graphical line with opened arrow. The types of association are as follows:

- Directional
- BiDirectional
- Default

The `association` property allows you to define the type of association. The following code example illustrates how to create an association.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
  //Sets type of the connector as Orthogonal
  type: 'Orthogonal',
  //Sets type as Bpmn, shape as Association and association as BiDirectional
  shape: {
    type: 'Bpmn',
    flow: 'Association',
    association: 'BiDirectional'
  }
}];

```

```

    },
  ]];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}>
    <Inject services={[BpmnDiagrams]}/>
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  ConnectorModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnShapeModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
  //Sets type of the connector as Orthogonal
  type: 'Orthogonal',
  //Sets type as Bpmn, shape as Association and association as
  BiDirectional
  shape: {
    type: 'Bpmn',
    flow: 'Association',
    association: 'BiDirectional'
  },
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent

```


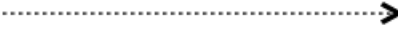



```

    id="container"
    width={ '100%' }
    height={ '600px' }
    // Add connector
    connectors={connector}
  >
    <Inject services={[BpmnDiagrams]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The following table demonstrates the visual representation of association flows.

Association	Image
Default	
Directional	
BiDirectional	

Note : The default value for the property `association` is **default**.

Sequence

A [sequence](#) flow shows the order in which the activities are performed in a BPMN process and is represented by a solid graphical line. The types of sequence are as follows:

- Normal
- Conditional
- Default

The sequence property allows you to define the type of sequence. The following code example illustrates how to create a sequence flow.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,

```

```

        y: 200
      },
      type: 'Orthogonal',
      //Sets type as Bpmn, flow as Sequence and sequence as Conditional
      shape: {
        type: 'Bpmn',
        flow: 'Sequence',
        sequence: 'Conditional'
      },
    }
  ]];
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}>
    <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  ConnectorModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnShapeModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
  type: 'Orthogonal',
  //Sets type as Bpmn, flow as Sequence and sequence as Conditional
  shape: {
    type: 'Bpmn',
    flow: 'Sequence',
    sequence: 'Conditional'
  }
}




```

```

    },
  }];
  // initialize diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        // Add connector
        connectors={connector}
      >
        <Inject services={[BpmnDiagrams]} />
      </DiagramComponent>
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

The following table contains various representation of sequence flows.

Sequence	Image
-----	-----
Default	
Conditional	
Normal	

Note: The default value for the property `sequence` is **normal**.

Message

A [message](#) flow shows the flow of messages between two participants and is represented by dashed line. The types of message are as follows:

- InitiatingMessage
- NonInitiatingMessage
- Default

The message property allows you to define the type of message. The following code example illustrates how to define a message flow.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, BpmnDiagrams } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.

```

```

let connector = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {
    x: 300,
    y: 200
  },
  type: 'Orthogonal',
  //Sets type as Bpmn, flow as Message and message as
  InitiatingMessage
  shape: {
    type: 'Bpmn',
    flow: 'Message',
    message: 'InitiatingMessage'
  },
}];
// initialize diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}>
    <Inject services={[BpmnDiagrams]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  NodeModel,
  ConnectorModel,
  BpmnShape,
  BpmnSubProcessModel,
  BpmnShapeModel,
  BpmnDiagrams,
  BpmnActivityModel,
  BpmnFlowModel,
  BpmnGatewayModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  // Position of the node
  sourcePoint: {
    x: 100,
    y: 200
  },
  targetPoint: {




```

```

        x: 300,
        y: 200
    },
    type: 'Orthogonal',
    //Sets type as Bpmn, flow as Message and message as InitiatingMessage
    shape: {
        type: 'Bpmn',
        flow: 'Message',
        message: 'InitiatingMessage'
    },
    },
    }];
// initialize diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={ '100%' }
            height={ '600px' }
            // Add connector
            connectors={connector}
        >
            <Inject services={[BpmnDiagrams]} />
        </DiagramComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The following table contains various representation of message flows.

Message	Image
Default	
InitiatingMessage	
NonInitiatingMessage	



Note: The default value for the property `message` is **default**.

UML diagram in React Diagram component

UML Class Diagram

A class diagram visually depicts the static structure of an application and is extensively employed in modeling object-oriented systems. It holds a unique position in UML diagrams, as it directly aligns with object-oriented languages. The diagram also facilitates the automatic generation of class diagram

shapes based on business logic, streamlining the translation from conceptual models to practical implementation.

Uml Class Diagram Shapes

The UML class diagram shapes are explained as follows.

Class

- A class defines a group of objects that share common specifications, features, constraints, and semantics. To create a class object, the classifier should be defined using the [class](#) notation. This notation serves as a foundational element in object-oriented programming, encapsulating the essential characteristics and behavior that objects belonging to the class will exhibit.
- Also, define the [name](#), [attributes](#), and [methods](#) of the class using the class property of node.
- The attribute's [name](#), [type](#), and [scope](#) properties allow you to define the name, data type, and visibility of the attribute.
- The method's [name](#), [parameters](#), [type](#), and [scope](#) properties allow you to define the name, parameter, return type, and visibility of the methods.
- The method parameters object properties allow you to define the name and type of the parameter.
- The following code example illustrates how to create a class.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
  id: 'Patient',
  shape: {
    type: 'UmlClassifier',
    classShape: {
      name: 'Patient',
      attributes: [
        createProperty('name', 'Name'),
        createProperty('title', 'String[*]'),
        createProperty('gender', 'Gender')
      ]
    },
    classifier: 'Class'
  },
  offsetX: 405,
  offsetY: 105
}];
//create class Property
function createProperty(name, type) {
  return { name: name, type: type };
}
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
}
```

```

}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'Patient',
  shape: {
    type: 'UmlClassifier',
    classShape: {
      name: 'Patient',
      attributes: [
        createProperty('name', 'Name'),
        createProperty('title', 'String[*]'),
        createProperty('gender', 'Gender')
      ]
    }
  },
  classifier: 'Class'
},
  {
    offsetX: 405,
    offsetY: 105
  }
];
//create class Property
function createProperty(name, type) {
  return { name: name, type: type };
}
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Interface

- An interface is a specific type of classifier that signifies a declaration of a cohesive set of public features and obligations. When creating an interface, involves defining the classifier property using the [interface](#) notation. This essential concept in object-oriented programming outlines a contract for classes to adhere to, specifying the required methods and behaviors without delving into the implementation details.
- Also, define the [name](#), [attributes](#), and [methods](#) of the interface using the interface property of the node.
- The attribute's name, type, and scope properties allow you to define the name, data type, and visibility of the attribute.
- The method's name, parameter, type, and scope properties allow you to define the name, parameter, return type, and visibility of the methods.
- The method parameter object properties of name and type allow you to define the name and type of the parameter.
- The following code example illustrates how to create an interface.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
  id: 'node',
  offsetX: 400,
  offsetY: 300,
  shape: {
    type: 'UmlClassifier',
    interfaceShape: {
      name: "Bank Account",
      property: [{
        name: "owner",
        type: "String[*]", style: {}
      },
      {
        name: "balance",
        type: "Dollars"
      }
    ],
    methods: [{
      name: "deposit", style: {},
      parameters: [{
        name: "amount",
        type: "Dollars",
        style: {}
      }
    ]
  }
},
  classifier: 'Interface'
}
];
// initialize Diagram component
function App() {
```



```

    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node',
  offsetX: 400,
  offsetY: 300,
  shape: {
    type: 'UmlClassifier',
    interfaceShape: {
      name: "Bank Account",
      property: [{
        name: "owner",
        type: "String[*]", style: {}
      },
      {
        name: "balance",
        type: "Dollars"
      }
    ],
    methods: [{
      name: "deposit", style: {},
      parameters: [{
        name: "amount",
        type: "Dollars",
        style: {}
      }
    ]
  }
    ],
    classifier: 'Interface'
  } as UmlClassifierShapeModel
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}

```

```

        // render initialized Diagram
    />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Enumeration

- To establish an enumeration, designate the classifier property of the node as [enumeration](#). Additionally, define the name and enumerate the members of the enumeration using the appropriate enumeration property of the node. This process encapsulates a set of distinct values within the enumeration, allowing for a clear representation of specific, and named constants within a system.
- You can set a name for the enumeration members collection using the name property of members collection.
- The following code example illustrates how to create an enumeration.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
    id: 'node',
    offsetX: 300,
    offsetY: 200,
    shape: {
        type: 'UmlClassifier',
        enumerationShape: {
            name: 'AccountType',
            members: [
                {
                    name: 'Checking Account', style: {}
                },
                {
                    name: 'Savings Account'
                },
                {
                    name: 'Credit Account'
                }
            ]
        },
        classifier: 'Enumeration'
    }
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node}/>);
}

```

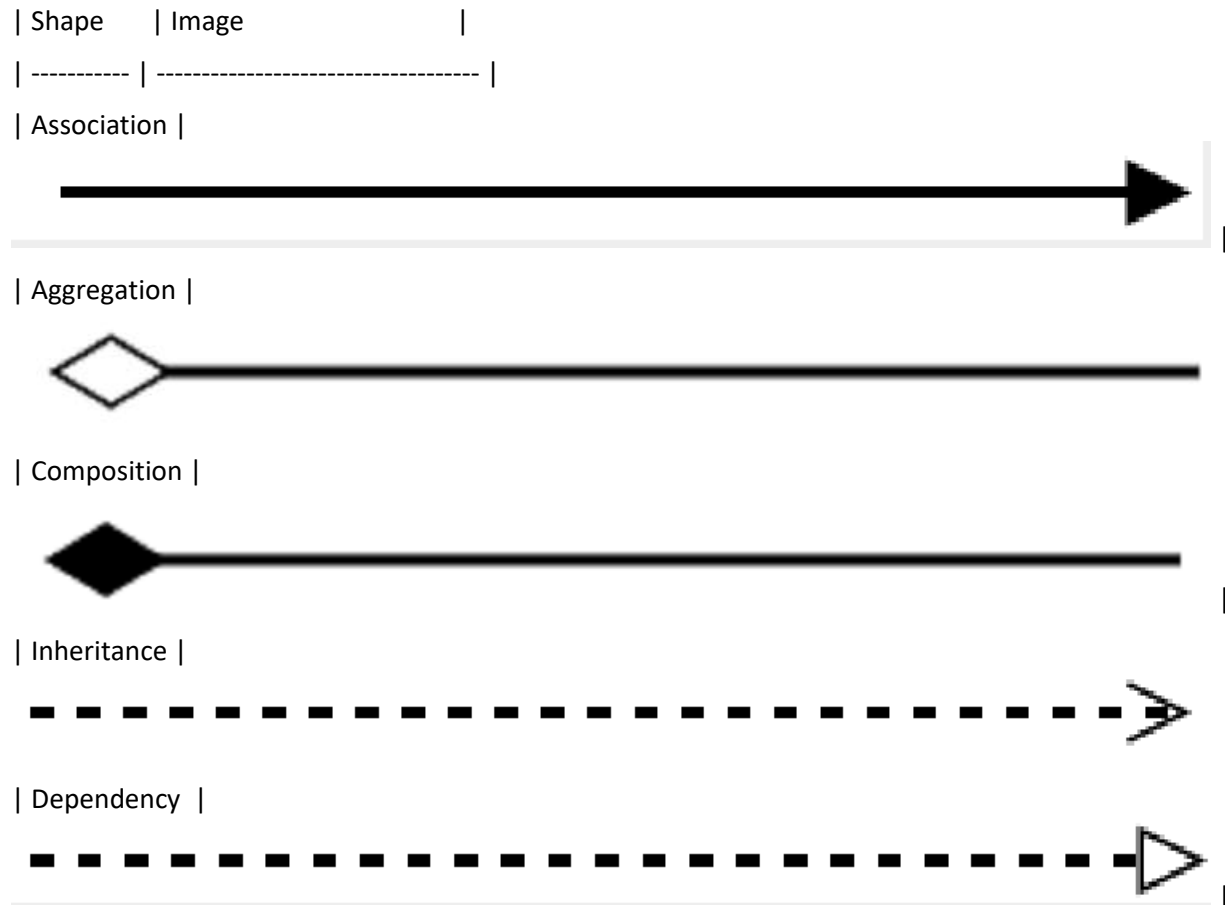
```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  UmlClassifierShapeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node',
  offsetX: 300,
  offsetY: 200,
  shape: {
    type: 'UmlClassifier',
    enumerationShape: {
      name: 'AccountType',
      members: [
        {
          name: 'Checking Account', style: {}
        },
        {
          name: 'Savings Account'
        },
        {
          name: 'Credit Account'
        }
      ]
    }
  },
  classifier: 'Enumeration'
} as UmlClassifierShapeModel
]];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

UML Class Relationships

- A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:



Association

Association is basically a set of links that connects elements of a UML model. The type of association is as follows.

1. Directional 2. BiDirectional

The association property allows you to define the type of association. The default value of association is "Directional". The following code example illustrates how to create an association.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
```

```

        targetPoint: { x: 300, y: 300 },
        type: "Straight",
        shape: {
            type: "UmlClassifier",
            relationship: "Association",
            //Define type of association
            association: "BiDirectional"
        }
    }
    });
    // initialize Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
            // Add connector
            connectors={connector}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
    id: "connector",
    //Define connector start and end points
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        relationship: "Association",
        //Define type of association
        association: "BiDirectional"
    }
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            // Add connector
            connectors={connector}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Aggregation

Aggregation is a binary association between a property and one or more composite objects that group together a set of instances. Aggregation is decorated with a hollow diamond. To create an aggregation shape, define the relationship as “aggregation”.

The following code example illustrates how to create an aggregation.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Aggregation"
  }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
  }
}];
```

```

        relationship: "Aggregation"
      }
    }];
    // initialize Diagram component
    function App() {
      return (
        <DiagramComponent
          id="container"
          width={'100%'}
          height={'600px'}
          // Add connector
          connectors={connector}
        />
      );
    }
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Composition

Composition is a “strong” form of “aggregation”. The composition is decorated with a black diamond. To create a composition shape, define the relationship property of the connector as “composition”.

The following code example illustrates how to create a composition.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Composition"
  }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";

```

```

import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Composition"
  }
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add connector
      connectors={connector}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Dependency

Dependency is a directed relationship, which is used to show that some UML elements need or depend on other model elements for specifications. Dependency is shown a dashed line with an opened arrow. To create a dependency, define the relationship property of the connector as “dependency”.

The following code example illustrates how to create a dependency.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",

```



```

        //Set an relationship for connector
        relationship: "Dependency"
    }
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add connector
        connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
    id: "connector",
    //Define connector start and end points
    sourcePoint: { x: 100, y: 100 },
    targetPoint: { x: 300, y: 300 },
    type: "Straight",
    shape: {
        type: "UmlClassifier",
        //Set an relationship for connector
        relationship: "Dependency"
    }
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            // Add connector
            connectors={connector}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Inheritance

Inheritance is also called a “generalization”. Inheritance is a binary taxonomic directed relationship between a more general classifier (superclass) and a more specific classifier (subclass). Inheritance is shown as a line with a hollow triangle.

To create an inheritance, define the relationship as “inheritance”.

The following code example illustrates how to create an inheritance.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Inheritance"
  }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    //Set an relationship for connector
    relationship: "Inheritance"
  }
}];
// initialize Diagram component
function App() {
```

```

return (
  <DiagramComponent
    id="container"
    width={'100%'}
    height={'600px'}
    // Add connector
    connectors={connector}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Multiplicity

Multiplicity is a definition of an inclusive interval of non-negative integers to specify the allowable number of instances of a described element. The type of multiplicity are as follows.

1.OneToOne 2.ManyToOne 3.OneToMany 4.ManyToMany By default the multiplicity will be considered as “OneToOne”. The multiplicity property in UML allows you to specify large number of elements or some collection of elements. The shape multiplicity’s source property is used to set the source label to the connector and the target property is used to set the target label to the connector. To set an optionality or cardinality for the connector source label, use the optional property. * The [lowerBounds](#) and [upperBounds](#) could be natural constants or constant expressions evaluated to a natural (non negative) number. The upper bound could also be specified as an asterisk ‘*’ which denotes an unlimited number of elements. The upper bound should be greater than or equal to the lower bound. * The following code example illustrates how to customize the multiplicity.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Dependency",
    multiplicity: {
      //Set multiplicity type
      type: "OneToMany",
      //Set source label to connector
      source: {
        optional: true,
        lowerBounds: 89,
        upperBounds: 67
      },
    },
    //Set target label to connector
    target: {
      optional: true,

```

```

        lowerBounds: 78,
        upperBounds: 90
      }
    }
  }
  });
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: "connector",
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 },
  type: "Straight",
  shape: {
    type: "UmlClassifier",
    relationship: "Dependency",
    multiplicity: {
      //Set multiplicity type
      type: "OneToMany",
      //Set source label to connector
      source: {
        optional: true,
        lowerBounds: 89,
        upperBounds: 67
      },
      //Set target label to connector
      target: {
        optional: true,
        lowerBounds: 78,
        upperBounds: 90
      }
    }
  }
}];
// initialize Diagram component
function App() {
  return (

```

```

    <DiagramComponent
      id="container"
      width={ '100%' }
      height={ '600px' }
      // Add connector
      connectors={connector}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

How to add UML child at runtime

In UML nodes, child elements such as members, methods and attributes can be added either programmatically or interactively.

Adding UML child through code

The [addChildToUmlNode](#) method is employed for dynamically adding a child to the UML node during runtime, providing flexibility in modifying the diagram structure programmatically.

The following code illustrates how to add methods to UML nodes in the diagram.

```

`ts
let node = diagram.selectedItems.nodes[0];

let methods = { name: 'getHistory', style: { color: "red", }, parameters: [{ name: 'Date', style: {} }], type:
'History' };

diagram.addChildToUmlNode(node, methods, 'Method');
`

```

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let nodes = [{
  id: 'node1',
  offsetX: 150,
  offsetY: 150,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: 'UmlClassifier',
    classShape: {
      attributes: [
        { name: 'accepted', type: 'Date', },
      ],
      methods: [{ name: 'getHistory', style: {}, parameters: [{ name:
'Date', style: {} }], type: 'History' }],
      name: 'Patient'
    },
  },
},

```

```

        classifier: 'Class'
      },
    ]];
document.getElementById('addMethod').onclick = function () {
  let node = diagramInstance.nodes[0];
  let methods = {
    name: 'getHistory',
    style: { color: 'red' },
    parameters: [{ name: 'Date', style: {} }],
    type: 'History',
  };
  diagramInstance.addChildToUmlNode(node, methods, 'Method');
};
// initialize Diagram component
function App() {
  return (
    <DiagramComponent id="container" ref={(diagram) =>
      (diagramInstance = diagram)} width={'100%'} height={'600px'}
      nodes={nodes}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  id: 'node1',
  offsetX: 150,
  offsetY: 150,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: 'UmlClassifier',
    classShape: {
      attributes: [
        { name: 'accepted', type: 'Date', },
      ],
      methods: [{ name: 'getHistory', style: {}, parameters: [{ name:
        'Date', style: {} }], type: 'History' }],
      name: 'Patient'
    },
    classifier: 'Class'
  },
}];
let addButton:any = document.getElementById('addMethod');
if (addButton) {

```

```

        addButton.addEventListener('click', function() {
            let node = diagramInstance.nodes[0];
            let methods = {
                name: 'getHistory',
                style: { color: 'red' },
                parameters: [{ name: 'Date', style: {} }],
                type: 'History',
            };
            diagramInstance.addChildToUmlNode(node, methods, 'Method');
        });
    }
    function App() {
        return (
            <DiagramComponent
                id="container"
                ref={(diagram) => (diagramInstance = diagram)}
                width={'100%'}
                height={'600px'}
                nodes={nodes}
            />
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

The following code illustrates how to add attributes to UML nodes in the diagram.

`ts

```

let node = diagram.selectedItems.nodes[0];
let attributes = { name: 'accepted', type: 'Date', style: { color: 'red', } };
diagram.addChildToUmlNode(node, attributes, "Attribute");

```

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let nodes = [{
    id: 'node1',
    offsetX: 150,
    offsetY: 150,
    style: {
        fill: '#26A0DA',
    },
    shape: {
        type: 'UmlClassifier',
        classShape: {
            attributes: [
                { name: 'accepted', type: 'Date', },
            ],
        },
    },
}];

```

```

        methods: [{ name: 'getHistory', style: {}, parameters: [{ name:
'Date', style: {} }], type: 'History' }],
        name: 'Patient'
    },
    classifier: 'Class'
},
]);
document.getElementById('addAttribute').onclick = function () {
    let node = diagramInstance.nodes[0];
    let attributes = { name: 'accepted', type: 'Date', style: { color:
'red', } };
    diagramInstance.addChildToUmlNode(node, attributes, "Attribute");
};
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'}
nodes={nodes}
/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
    id: 'node1',
    offsetX: 150,
    offsetY: 150,
    style: {
        fill: '#26A0DA',
    },
    shape: {
        type: 'UmlClassifier',
        classShape: {
            attributes: [
                { name: 'accepted', type: 'Date', },
            ],
            methods: [{ name: 'getHistory', style: {}, parameters: [{ name:
'Date', style: {} }], type: 'History' }],
            name: 'Patient'
        },
        classifier: 'Class'
    },
}];
let addButton:any = document.getElementById('addAttribute');
if (addButton) {

```



```

        addButton.addEventListener('click', function() {
            let node = diagramInstance.nodes[0];
            let attributes = { name: 'accepted', type: 'Date', style: { color:
"red", } };
            diagramInstance.addChildToUmlNode(node, attributes, "Attribute");
        });
    }
    function App() {
        return (
            <DiagramComponent
                id="container"
                ref={ (diagram) => (diagramInstance = diagram) }
                width={ '100%' }
                height={ '600px' }
                nodes={ nodes }
            />
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

The following code illustrates how to add members to UML nodes in the diagram.

```
`ts
```

```
let node = diagram.selectedItems.nodes[0];
```

```
let members = { name: 'Checking new', style: { color: 'red', }, isSeparator: true };
```

```
diagram.addChildToUmlNode(node, members, "Member");
```

```
,
```

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let nodes = [{
    id: 'node1',
    offsetX: 150,
    offsetY: 150,
    style: {
        fill: '#26A0DA',
    },
    shape: {
        type: 'UmlClassifier',
        enumerationShape: {
            name: 'AccountType',
            members: [
                {
                    name: 'Checking Account',
                },
            ],
        },
        classifier: 'Enumeration'
    },

```

```

    },
  ]];
document.getElementById('addMember').onclick = function () {
  let node = diagramInstance.nodes[0];
  let members = { name: 'Checking new', style: { color: "red", },
isSeparator: true };
  diagramInstance.addChildToUmlNode(node, members, "Member");
};
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'}
nodes={nodes}
/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let nodes: NodeModel[] = [{
  id: 'node1',
  offsetX: 150,
  offsetY: 150,
  style: {
    fill: '#26A0DA',
  },
  shape: {
    type: 'UmlClassifier',
    enumerationShape: {
      name: 'AccountType',
      members: [
        {
          name: 'Checking Account',
        },
      ],
    },
  },
  classifier: 'Enumeration'
},
];
let addButton:any = document.getElementById('addMember');
if (addButton) {
  addButton.addEventListener('click', function() {
    let node = diagramInstance.nodes[0];
    let members = { name: 'Checking new', style: { color: "red", },
isSeparator: true };
    diagramInstance.addChildToUmlNode(node, members, "Member");
  });
}

```

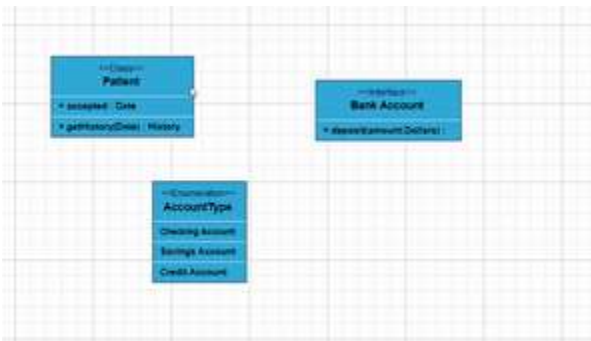
```

    });
  }
  function App() {
    return (
      <DiagramComponent
        id="container"
        ref={(diagram) => (diagramInstance = diagram)}
        width={'100%'}
        height={'600px'}
        nodes={nodes}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Adding UML child through user interaction

To include a child, select a node, move the mouse outside it, and position the pointer near the right side. A highlighter emerges between the two child elements. Click the highlighter to add a child type to the chosen UML node seamlessly. The following gif illustrates how to add a Child through user interaction.



Adding UML Nodes in Symbol palette

UML built-in shapes are efficiently rendered in a symbol palette. The `symbols` property is utilized to define UML symbols with the necessary classes and methods. By incorporating this feature, you can seamlessly augment the palette with a curated collection of predefined UML symbols, thereby enhancing the versatility of your UML diagramming application.

The following code example showcases the rendering of UML built-in shapes in a symbol palette.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, DiagramComponent } from "@syncfusion/ej2-react-diagrams";
//Initialize the basic shapes for the symbol palette
export function getUmlShapes() {
  let umlShapes = [
    {
      id: 'class',
      style: {
        fill: '#26A0DA',
      },
    },
  ],

```

```

        borderColor: 'white',
        shape: {
          type: 'UmlClassifier',
          classShape: {
            attributes: [
              { name: 'accepted', type: 'Date', style: { color:
"red", fontFamily: "Arial", textDecoration: 'Underline', italic: true
},isSeparator: true },
            ],
            methods: [{ name: 'getHistory', style: {}, parameters:
[ { name: 'Date', style: {} } ], type: 'History' } ],
            name: 'Patient'
          },
          classifier: 'Class'
        },
      },
    {
      id: 'Interface',
      style: {
        fill: '#26A0DA',
      }, borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        interfaceShape: {
          name: "Bank Account",
        },
        classifier: 'Interface'
      },
    },
    {
      id: 'Enumeration',
      style: {
        fill: '#26A0DA',
      }, borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        enumerationShape: {
          name: 'AccountType',
          members: [
            {
              name: 'Checking Account', style: {}
            },
          ],
        },
        classifier: 'Enumeration'
      },
    },
  ],
  ];
  return umlShapes;
}
//Initializes the symbol palette
let diagramInstance;
function App() {
  return (
    <div style={{ width: '100%' }}>
      <div id="palette-space" className="sb-mobile-palette">
        <SymbolPaletteComponent

```

```

        id="container"
        palettes=[
          {
            id: 'uml',
            expanded: true,
            symbols: getUmlShapes(),
            title: 'UML Shapes',
          },
        ]
        symbolHeight={80}
        symbolWidth={80}
        getNodeDefaults=(symbol) => {
          symbol.width = 100;
          symbol.height = 100;
        }
        //Sets the margin of the dragging helper relative to the
mouse cursor
        symbolMargin={{
          left: 12,
          right: 12,
          top: 12,
          bottom: 12,
        }}
        getSymbolInfo=(symbol) => {
          //Defines the symbol description
          return { fit: true, description: { text: symbol.id } };
        }
      />
    </div>
    <div id="diagram-space" className="sb-mobile-diagram">
      <DiagramComponent
        id="diagram"
        ref={(diagram) => (diagramInstance = diagram)}
        width={'100%'}
        height={'700px'}
      ></DiagramComponent>
    </div>
  </div>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  DiagramComponent,
  SymbolInfo,
  NodeConstraints,

```

```

SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
let diagramInstance:any;
export function getUmlShapes(): NodeModel[] {
  let umlShapes: NodeModel[] = [
    {
      id: 'class',
      style: {
        fill: '#26A0DA',
      },
      borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        classShape: {
          attributes: [
            { name: 'accepted', type: 'Date', style: { color: "red",
fontFamily: "Arial", textDecoration: 'Underline', italic: true
},isSeparator: true },
          ],
          methods: [{ name: 'getHistory', style: {}, parameters: [{
name: 'Date', style: {} }], type: 'History' }],
          name: 'Patient'
        },
        classifier: 'Class'
      },
    },
    {
      id: 'Interface',
      style: {
        fill: '#26A0DA',
      },
      borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        interfaceShape: {
          name: "Bank Account",
        },
        classifier: 'Interface'
      },
    },
    {
      id: 'Enumeration',
      style: {
        fill: '#26A0DA',
      },
      borderColor: 'white',
      shape: {
        type: 'UmlClassifier',
        enumerationShape: {
          name: 'AccountType',
          members: [
            {
              name: 'Checking Account', style: {}
            },
          ],
        },
        classifier: 'Enumeration'
      },
    },
  ],

```

```

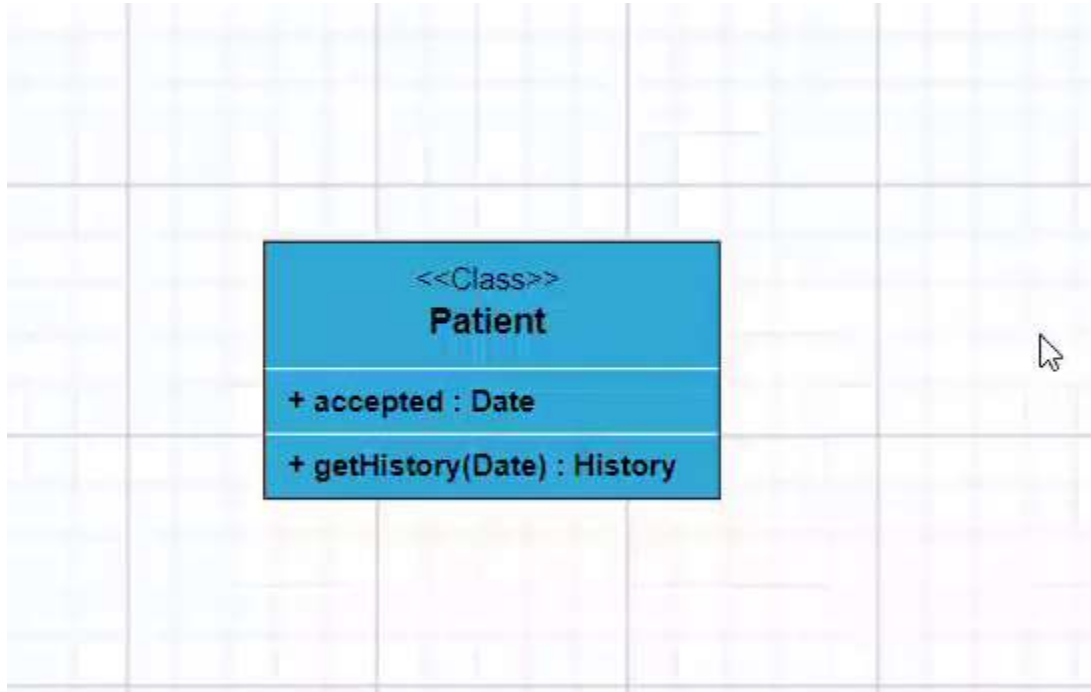
    },
    ];
    return umlShapes;
}
//Initializes the symbol palette
function App() {
    return (
        <div style={{ width: '100%' }}>
            <div id="palette-space" className="sb-mobile-palette">
                <SymbolPaletteComponent
                    id="container"
                    palettes={[
                        {
                            id: 'uml',
                            expanded: true,
                            symbols: getUmlShapes(),
                            title: 'UML Shapes',
                        },
                    ]}
                    symbolHeight={80}
                    symbolWidth={80}
//Sets the margin of the dragging helper relative to the mouse
                    symbolMargin={{
                        left: 12,
                        right: 12,
                        top: 12,
                        bottom: 12,
                    }}
                    getNodeDefaults={(symbol: NodeModel): void => {
                        symbol.width = 100;
                        symbol.height = 100;
                    }}
                    getSymbolInfo={(symbol: NodeModel): SymbolInfo => {
//Defines the symbol description
                        return { fit: true, description: { text: symbol.id, } };
                    }}
                />
            </div>
            <div id="diagram-space" className="sb-mobile-diagram">
                <DiagramComponent
                    id="diagram"
                    ref={(diagram) => (diagramInstance = diagram)}
                    width={'100%'}
                    height={'700px'}
                ></DiagramComponent>
            </div>
        </div>
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Editing

You can edit the name, attributes, and methods of the class diagram shapes just double clicking, similar to editing a node annotation.

The following image illustrates how the text editor looks in an edit mode.



UML Activity diagram

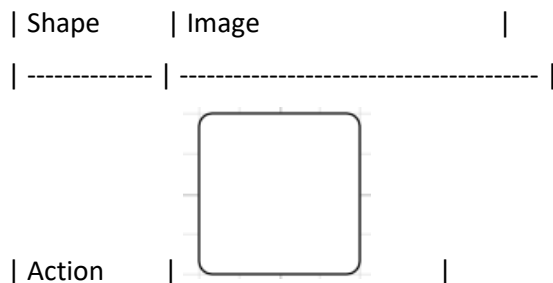
An Activity diagram functions as a visual flowchart, illustrating the progression from one activity to the next within a system. Each activity corresponds to a system operation, providing a clear depiction of the sequential flow in a dynamic process..

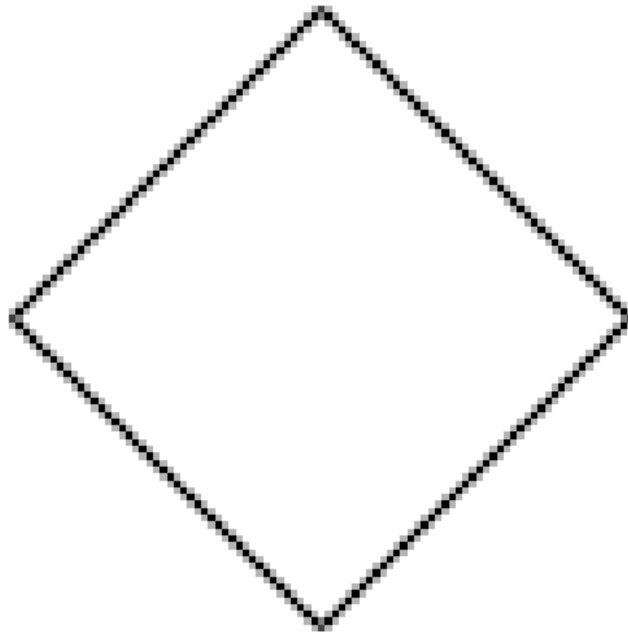
The purpose of an activity diagram can be described as follows.

1. Draw the activity flow of a system.
2. Describe the sequence from one activity to another.
3. Describe the parallel, branched, and concurrent flow of the system.

UML Activity diagram Shapes

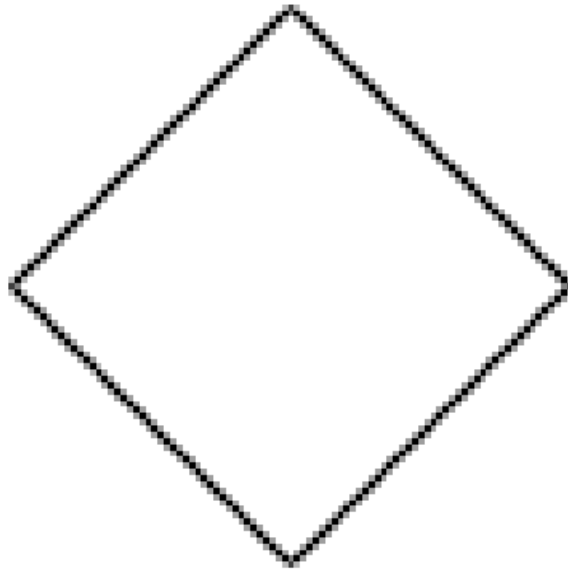
To create a UmlActivity, define the type as "UmlActivity" and the list of built-in shapes as demonstrated as follows and it should be set in the "shape" property.





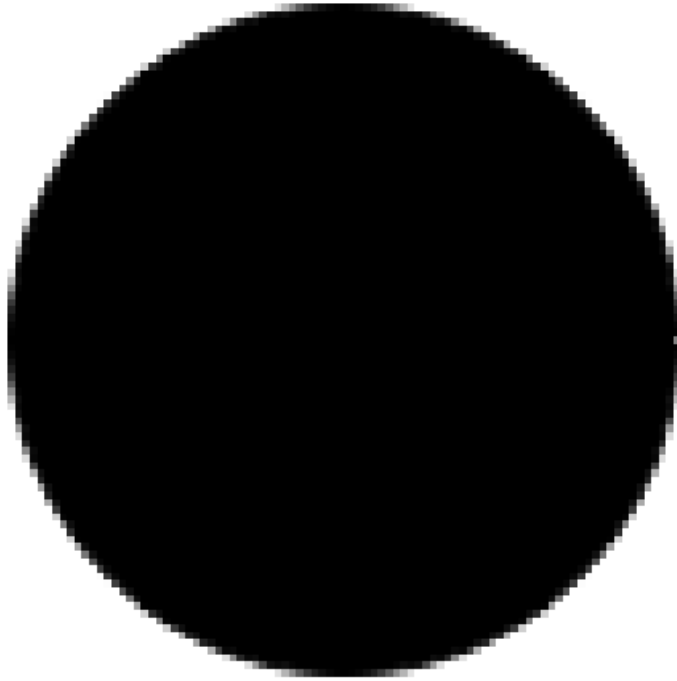
| Decision |

|



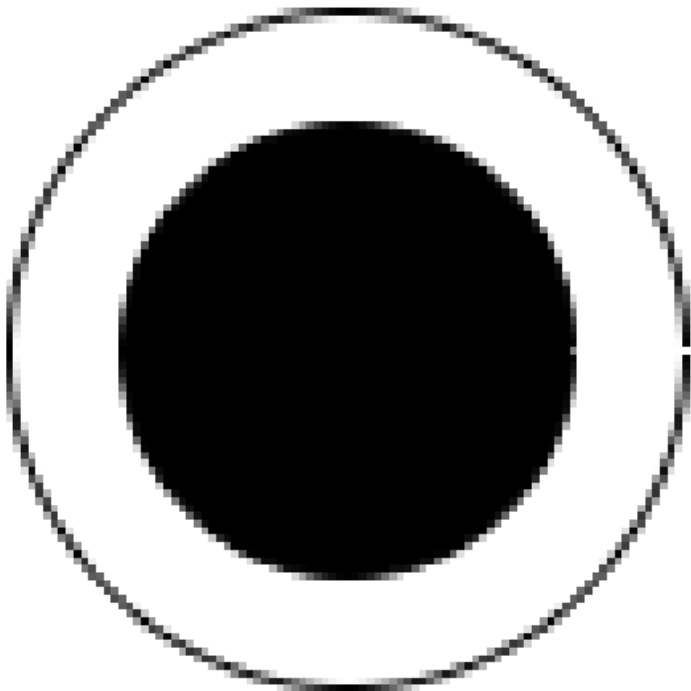
| MergeNode |

|



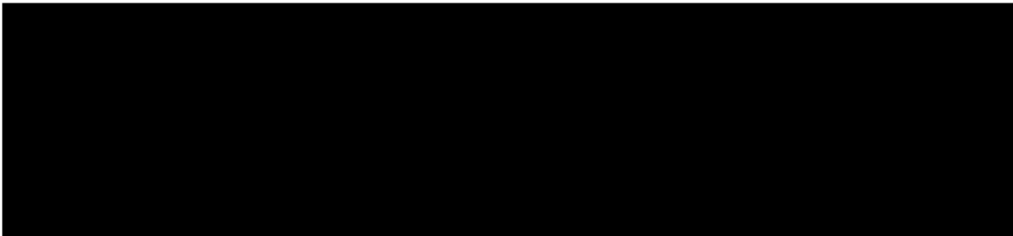
| InitialNode |

|



| FinalNode |
| ForkNode |

|

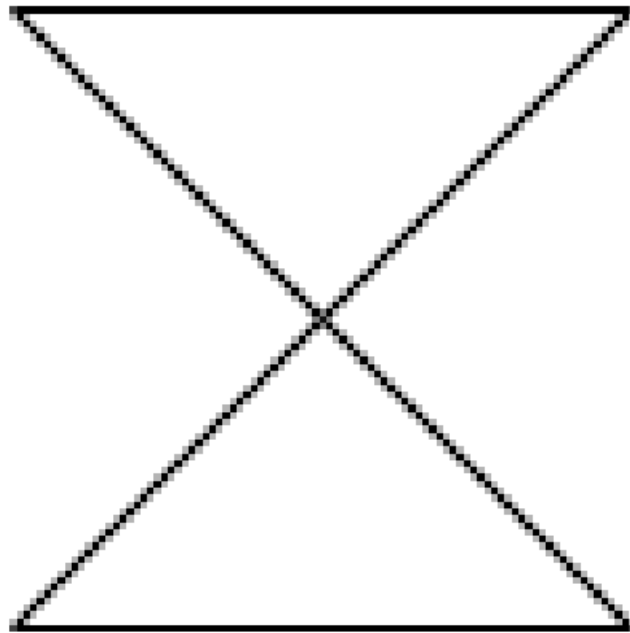


|



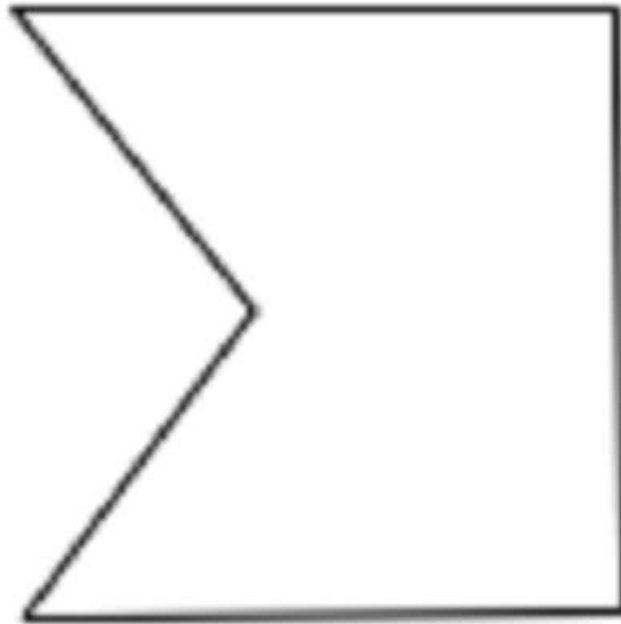
| JoinNode |

|



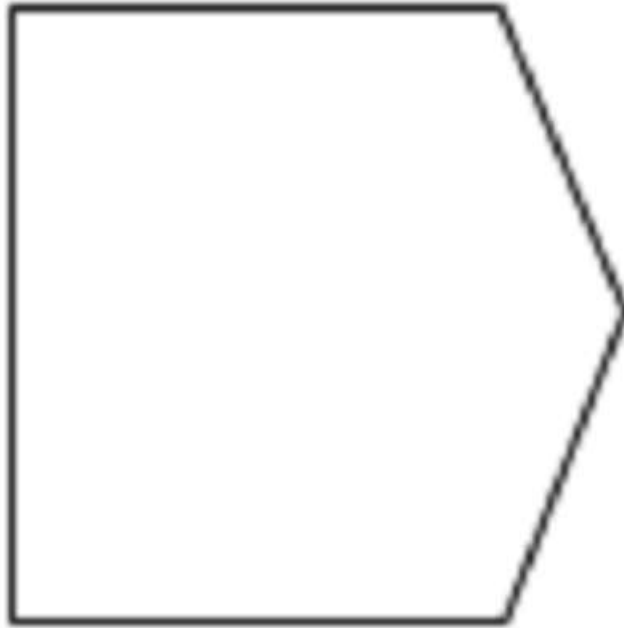
| TimeEvent |

|



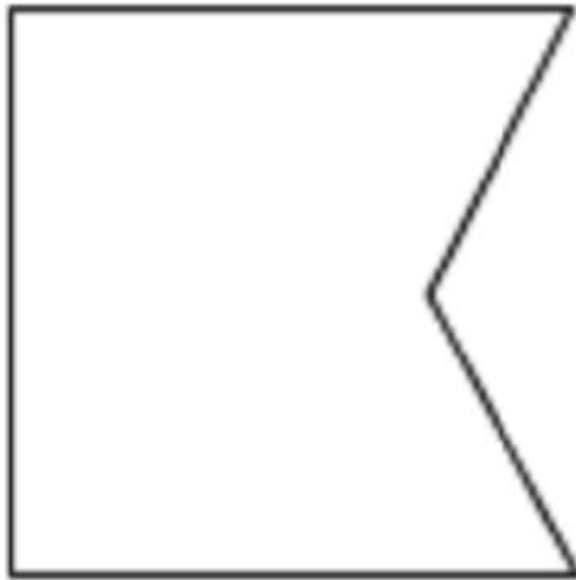
| AcceptingEvent |

|



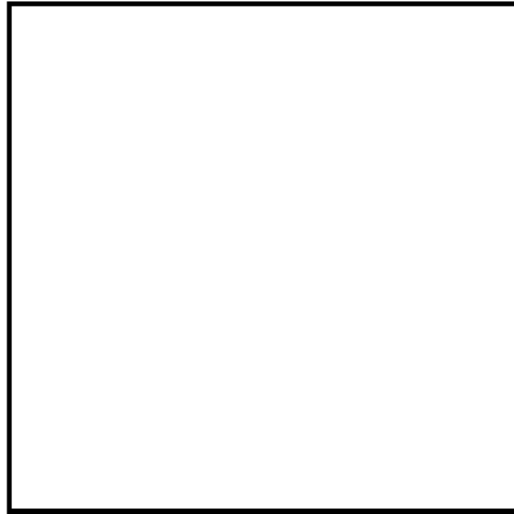
| SendSignal |

|



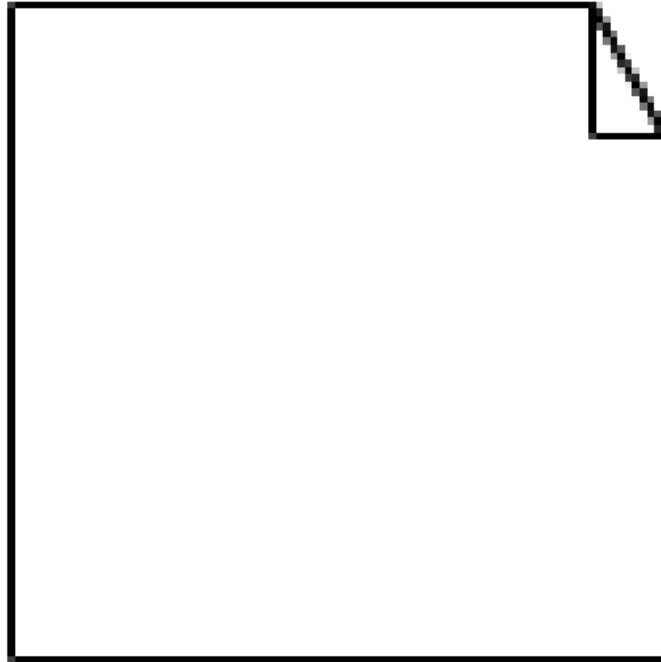
| ReceiveSignal |

|



| StructuredNode |

|



| Note

The following code illustrates how to create a UmlActivity shapes.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
  id: "UmlDiagram",
  //Set node size
  width: 100,
  height: 100,
  //position the node
  offsetX: 200,
  offsetY: 200,
  shape: {
    type: "UmlActivity",
    //Define UmlActivity shape
    shape: "Action"
  }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}>
    // Add node
```

```

    nodes={node}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: "UmlDiagram",
  //Set node size
  width: 100,
  height: 100,
  //position the node
  offsetX: 200,
  offsetY: 200,
  shape: {
    type: "UmlActivity",
    //Define UmlActivity shape
    shape: "Action"
  }
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Uml Activity connector

To create an Uml Activity connector, define the type as "UmlActivity" and flow as either "Exception" or "Control" or "Object".

The following code illustrates how to create a UmlActivity connector.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";

```

```

let diagramInstance;
// A node is created and stored in nodes array.
let connector = [{
  id: 'connector',
  type: 'Straight',
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 200, y: 200 },
  shape: { type: 'UmlActivity', flow: 'Exception' }
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add connector
    connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: 'connector',
  type: 'Straight',
  //Define connector start and end points
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 200, y: 200 },
  shape: { type: 'UmlActivity', flow: 'Exception' }
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add connector
      connectors={connector}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Connectors in React Diagram component

Connectors are objects used to create link between two points, nodes or ports to represent the relationships between them.

Create connector

Connector can be created by defining the source and target point of the connector. The path to be drawn can be defined with a collection of segments. To explore the properties of a [connector](#), refer to [Connector Properties](#).

Add connectors through connectors collection

The [sourcePoint](#) and [targetPoint](#) properties of connector allow you to define the end points of a connector.

The following code example illustrates how to add a connector through connector collection.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  // Name of the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  // Sets source and target points
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
```

```

Diagram,
DiagramComponent,
ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  // Name of the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  // Sets source and target points
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}
]];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Add connector at runtime

Connectors can be added at runtime by using public method, `diagram.add` and can be removed at runtime by using public method, `diagram.remove`.

The following code example illustrates how to add connector at runtime.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let connectors = [{
  id: 'connector1',

```

```

        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7',
            strokeWidth: 2
        },
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        },
        sourcePoint: {
            x: 100,
            y: 100
        },
        targetPoint: {
            x: 200,
            y: 200
        }
    }
    });
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} created={() =>
{
        // Adds to the Diagram
        diagramInstance.add(connectors[0]);
        // Remove from the diagram
        diagramInstance.remove(connectors[0]);
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let connectors: ConnectorModel = [{
    id: 'connector1',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'

```



```

    },
    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    }
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      created={() => {
        // Adds to the Diagram
        diagramInstance.add(connectors[0]);
        // Remove from the diagram
        diagramInstance.remove(connectors[0]);
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Connectors from palette

Connectors can be predefined and added to the symbol palette. You can drop those connectors into the diagram, when required.

For more information about adding connectors from symbol palette, refer to [Symbol Palette](#).

Draw connectors

Connectors can be interactively drawn by clicking and dragging on the diagram surface by using [drawingObject](#).

For more information about drawing connectors, refer to [Draw Connectors](#).

Update connector at runtime

Various connector properties such as [sourcePoint](#), [targetPoint](#), [style](#), [sourcePortID](#), [targetPortID](#), etc., can be updated at the runtime.

The following code example illustrates how to update a connector's source point, target point, styles properties at runtime.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";

```

```

let connectors = [{
  id: "connector1",
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
let diagramInstance;
function App() {
  return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'}
    connectors={connectors} created={() => {
      // Update the connector properties at the run time
      diagramInstance.connectors[0].style.strokeColor = '#6BA5D7';
      diagramInstance.connectors[0].style.fill = '#6BA5D7';
      diagramInstance.connectors[0].style.strokeWidth = 2;
      diagramInstance.connectors[0].targetDecorator.style.fill =
        '#6BA5D7';
      diagramInstance.connectors[0].targetDecorator.style.strokeColor =
        '#6BA5D7';
      diagramInstance.connectors[0].sourcePoint.x = 150;
      diagramInstance.connectors[0].targetPoint.x = 150;
      diagramInstance.dataBind();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
let diagramInstance: DiagramComponent;

```

```
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      connectors={connectors}
      created={() => {
        // Update the connector properties at the run time
        diagramInstance.connectors[0].style.strokeColor = '#6BA5D7';
        diagramInstance.connectors[0].style.fill = '#6BA5D7';
        diagramInstance.connectors[0].style.strokeWidth = 2;
        diagramInstance.connectors[0].targetDecorator.style.fill =
        '#6BA5D7';
        diagramInstance.connectors[0].targetDecorator.style.strokeColor =
        '#6BA5D7';
        diagramInstance.connectors[0].sourcePoint.x = 150;
        diagramInstance.connectors[0].targetPoint.x = 150;
        diagramInstance.dataBind();
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}
```

Connect nodes

- The [sourceID](#) and [targetID](#) properties allow to define the nodes to be connected.
- The [connectorSpacing](#) property allows you to define the distance between the source node and the connector. It is the minimum distance the connector will re-route or the new segment will create.
- The following code example illustrates how to connect two nodes.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, } from "@syncfusion/ej2-react-diagrams";
let nodes = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
  }
},
```

```

        shape: 'Terminator'
      }
    },
    {
      id: 'Init',
      width: 140,
      height: 50,
      offsetX: 300,
      offsetY: 300,
      shape: {
        type: 'Flow',
        shape: 'Process'
      },
      annotations: [{
        content: 'var i = 0;'
      }]
    }
  ];
  let connectors = [{
    // Name of the connector
    id: "connector1",
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    // ID of the source and target nodes
    sourceID: "Start",
    targetID: "Init",
    connectorSpacing: 7,
    type: 'Orthogonal'
  }];
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={nodes} connectors={connectors}
    // Defines the default properties for the node
    getNodeDefaults=(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    })/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel,
} from "@syncfusion/ej2-react-diagrams";
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
},
{
  id: 'Init',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 300,
  shape: {
    type: 'Flow',
    shape: 'Process'
  },
  annotations: [{
    content: 'var i = 0;'
  }]
}
];
let connectors: ConnectorModel[] = [{
  // Name of the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  // ID of the source and target nodes
  sourceID: "Start",
  targetID: "Init",
  connectorSpacing: 7,
  type: 'Orthogonal'
}
];

```

```

    }];
    function App() {
      return (
        <DiagramComponent id="container"
          width = {
            '100%'
          }
          height = {
            '600px'
          }
          nodes = {
            nodes
          }
          connectors = {
            connectors
          }
          // Defines the default properties for the node
          getNodeDefaults = {
            (node: NodeModel) => {
              node.height = 100;
              node.width = 100;
              node.style.fill = '#6BA5D7';
              node.style.strokeColor = 'white';
              return node;
            }
          }
        />
      )
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

- When you remove NodeConstraints [InConnect](#) from Default, the node accepts only an outgoing connection to dock in it. Similarly, when you remove NodeConstraints [OutConnect](#) from Default, the node accepts only an incoming connection to dock in it.
- When you remove both InConnect and OutConnect NodeConstraints from Default, the node restricts connector to establish connection in it.
- The following code illustrates how to disable InConnect constraints.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";

```

```

let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 100,
  //Disable InConnect constraints
  constraints: NodeConstraints.Default & ~NodeConstraints.InConnect,
}]
];

function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Connections with ports

The [sourcePortID](#) and [targetPortID](#) properties allow to create connections between some specific points of source/target nodes.

The following code example illustrates how to create port to port connections.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, PortVisibility } from "@syncfusion/ej2-react-diagrams";
let port1 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}

```

```

};
port1.shape = 'Circle';
port1.id = 'nodeportnew';
port1.visibility = PortVisibility.Visible;
port1.id = 'port1';
port1.offset = {
  x: 1,
  y: 0.5
};
let port2 = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
port2.offset = {
  x: 0,
  y: 0.5
};
port2.id = 'port2';
port2.visibility = PortVisibility.Visible;
port2.shape = 'Circle';
let nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  ports: [port1]
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  ports: [port2]
},
];
let connectors = [{
  id: "connector1",
  sourceID: 'node',
  targetID: 'node1',
  sourcePortID: 'port1',
  targetPortID: 'port2'
}];
function App() {
  return (<DiagramComponent id="container" width={900} height={900}
nodes={nodes} connectors={connectors} getNodeDefaults={(node) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```



```
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  ConnectorModel,
  NodeModel,
  BasicShapeModel,
  PointPortModel,
  Diagram,
  DiagramComponent,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
let port1: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
port1.shape = 'Circle';
port1.id = 'nodeportnew';
port1.visibility = PortVisibility.Visible;
port1.id = 'port1';
port1.offset = {
  x: 1,
  y: 0.5
};
let port2: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
port2.offset = {
  x: 0,
  y: 0.5
};
port2.id = 'port2';
port2.visibility = PortVisibility.Visible;
port2.shape = 'Circle';
let nodes: NodeModel[] = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  ports: [port1]
},
{
  id: 'node1',
  width: 100,
  height: 100,
```

```

        offsetX: 300,
        offsetY: 100,
        ports: [port2]
    },
];
let connectors: ConnectorModel[] = [{
    id: "connector1",
    sourceID: 'node',
    targetID: 'node1',
    sourcePortID: 'port1',
    targetPortID: 'port2'
}];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={900}
            height={900}
            nodes={nodes}
            connectors={connectors}
            getNodeDefaults={(node: NodeModel) => {
                node.height = 100;
                node.width = 100;
                node.style.fill = '#6BA5D7';
                node.style.strokeColor = 'white';
                return node;
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Similarly, the `sourcePortID` or `targetPortID` can be changed at the runtime by changing the port [sourcePortID](#) or [targetPortID](#).

INDEX.JSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { DiagramComponent, PortVisibility, } from '@syncfusion/ej2-react-diagrams';
let port1 = {
    style: {
        strokeColor: '#366F8C',
        fill: '#366F8C',
    },
};
port1.shape = 'Circle';
port1.id = 'nodeportnew';
port1.visibility = PortVisibility.Visible;
port1.id = 'port';
port1.offset = {
    x: 1,

```

```

    y: 1,
  };
  let port2 = {
    style: {
      strokeColor: '#366F8C',
      fill: '#366F8C',
    },
  };
  port2.offset = {
    x: 1,
    y: 0.5,
  };
  port2.id = 'port1';
  port2.visibility = PortVisibility.Visible;
  port2.shape = 'Circle';
  let port3 = {
    style: {
      strokeColor: '#366F8C',
      fill: '#366F8C',
    },
  };
  port3.offset = {
    x: 0,
    y: 1,
  };
  port3.id = 'newnodeport1';
  port3.visibility = PortVisibility.Visible;
  port3.shape = 'Circle';
  let nodes = [
    {
      id: 'node',
      width: 100,
      height: 100,
      offsetX: 100,
      offsetY: 100,
      ports: [port1],
    },
    {
      id: 'node1',
      width: 100,
      height: 100,
      offsetX: 300,
      offsetY: 100,
      ports: [port2, port3],
    },
  ];
  let diagramInstance;
  let connectors = [
    {
      id: 'connector1',
      sourcePoint: {
        x: 100,
        y: 100,
      },
      type: 'Orthogonal',
      targetPoint: {
        x: 200,

```

```

        y: 200,
      },
      sourceID: 'node',
      targetID: 'node1',
      sourcePortID: 'port',
      targetPortID: 'port1',
    },
  ],
};

function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={900} height={900} nodes={nodes}
    connectors={connectors} getNodeDefaults={(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }}/>);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
// Update the target portID at the run time
diagramInstance.connectors[0].targetPortID = 'newnodeport1';
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import {
  ConnectorModel,
  NodeModel,
  BasicShapeModel,
  PointPortModel,
  Diagram,
  DiagramComponent,
  PortVisibility,
} from '@syncfusion/ej2-react-diagrams';
let port1: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C',
  },
};
port1.shape = 'Circle';
port1.id = 'nodeportnew';
port1.visibility = PortVisibility.Visible;
port1.id = 'port';
port1.offset = {
  x: 1,
  y: 1,
};
let port2: PointPortModel = {
  style: {
    strokeColor: '#366F8C',

```

```

        fill: '#366F8C',
    },
};
port2.offset = {
    x: 1,
    y: 0.5,
};
port2.id = 'port1';
port2.visibility = PortVisibility.Visible;
port2.shape = 'Circle';
let port3: PointPortModel = {
    style: {
        strokeColor: '#366F8C',
        fill: '#366F8C',
    },
};
port3.offset = {
    x: 0,
    y: 1,
};
port3.id = 'newnodeport1';
port3.visibility = PortVisibility.Visible;
port3.shape = 'Circle';
let nodes: NodeModel[] = [
    {
        id: 'node',
        width: 100,
        height: 100,
        offsetX: 100,
        offsetY: 100,
        ports: [port1],
    },
    {
        id: 'node1',
        width: 100,
        height: 100,
        offsetX: 300,
        offsetY: 100,
        ports: [port2, port3],
    },
];
let diagramInstance: DiagramComponent;
let connectors: ConnectorModel[] = [
    {
        id: 'connector1',
        sourcePoint: {
            x: 100,
            y: 100,
        },
        type: 'Orthogonal',
        targetPoint: {
            x: 200,
            y: 200,
        },
        sourceID: 'node',
        targetID: 'node1',
        sourcePortID: 'port',
    },
];

```

```

        targetPortID: 'port1',
    },
];
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={900}
            height={900}
            nodes={nodes}
            connectors={connectors}
            getNodeDefaults={(node: NodeModel) => {
                node.height = 100;
                node.width = 100;
                node.style.fill = '#6BA5D7';
                node.style.strokeColor = 'white';
                return node;
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
// Update the target portID at the run time
diagramInstance.connectors[0].targetPortID = 'newnodeport1';
{% enddraw %}

```

- When you set PortConstraints to [InConnect](#), the port accepts only an incoming connection to dock in it. Similarly, when you set PortConstraints to [OutConnect](#), the port accepts only an outgoing connection to dock in it.
- When you set PortConstraints to None, the port restricts connector to establish connection in it.

`ts

```

import * as React from "react";
import * as ReactDOM from "react-dom";

import {
    ConnectorModel,
    NodeModel,
    BasicShapeModel,
    PointPortModel,
    Diagram,
    DiagramComponent,
    PortVisibility
} from "@syncfusion/ej2-react-diagrams";

```

```

let port1: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}

port1.shape = 'Circle';
port1.id = 'nodeportnew';
//Enable portConstraints Inconnect
port1.constraints = PortConstraints.InConnect;
let nodes: NodeModel[] = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 150,
  ports: [port1]
},
];
function App() {
  return (
    <DiagramComponent id="container" width={900} height={900} nodes={nodes} />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Segments

The path of the connector is defined with a collection of segments. There are three types of segments.

Straight

To create a straight line, specify the [type](#) of the segment as **straight** and add a straight segment to [segments](#) collection and need to specify [type](#) for the connector. The following code example illustrates how to create a default straight segment.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  type: 'Straight',
  segments: [{
    // Defines the segment type of the connector
    type: 'Straight'
  }],
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  StraightSegmentModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  type: 'Straight',
  segments: [{
    // Defines the segment type of the connector
    type: 'Straight'
  }],
  style: {
    strokeColor: '#6BA5D7',

```



```

        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
}
}];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            connectors={connectors}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The [point](#) property of straight segment allows you to define the end point of it. The following code example illustrates how to define the end point of a straight segment.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
    id: "connector1",
    // Defines the segment type of the connector
    segments: [{
        type: 'Straight',
        // Defines the point of the segment
        point: {
            x: 100,
            y: 150
        }
    }],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {

```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    type: 'Straight',
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
}
});
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel,
    StraightSegmentModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
    id: "connector1",
    // Defines the segment type of the connector
    segments: [{
        type: 'Straight',
        // Defines the point of the segment
        point: {
            x: 100,
            y: 150
        }
    }],
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    type: 'Straight',
    sourcePoint: {

```

```

        x: 100,
        y: 100
      },
      targetPoint: {
        x: 200,
        y: 200
      }
    }
  ]];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Orthogonal

Orthogonal segments is used to create segments that are perpendicular to each other.

Set the segment [type](#) as orthogonal to create a default orthogonal segment and need to specify [type](#). The following code example illustrates how to create a default orthogonal segment.

Multiple segments can be defined one after another. To create a connector with multiple segments, define and add the segments to [connector.segments](#) collection. The following code example illustrates how to create a connector with multiple segments.

The property [maxSegmentThumb](#) is used to limit the segment thumb in the connector.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, ConnectorConstraints, ConnectorEditing }
from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connectors = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  }
}];

```

```

    },
    targetPoint: {
      x: 200,
      y: 200
    },
    type: 'Orthogonal',
    maxSegmentThumb: 3,
    constraints: ConnectorConstraints.Default &
~ConnectorConstraints.DragSegmentThumb,
    segments: [{ type: 'Orthogonal', direction: 'Bottom', length: 50 }],
  }];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  OrthogonalSegmentModel,
  ConnectorConstraints,
  ConnectorEditing
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connectors: ConnectorModel[] = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  type: 'Orthogonal',
  maxSegmentThumb: 3,

```

```

constraints: ConnectorConstraints.Default &
~ConnectorConstraints.DragSegmentThumb,
  segments: [{ type: 'Orthogonal', direction: 'Bottom', length: 50 }],
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

The [length](#) and [direction](#) properties allow to define the flow and length of segment. The following code example illustrates how to create customized orthogonal segments.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  type: 'Orthogonal',
  segments: [{
    type: 'Orthogonal',
    // Defines the direction for the segment lines
    direction: 'Right',
    // Defines the length for the segment lines
    length: 50
  }],
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
},
],

```

```

{
  id: "connector2",
  type: 'Orthogonal',
  // Defines multile segments for the connectors
  segments: [{
    type: 'Orthogonal',
    direction: 'Bottom',
    length: 150
  },
  {
    type: 'Orthogonal',
    direction: 'Right',
    length: 150
  }
  ],
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 200
  }
}
];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  OrthogonalSegmentModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  type: 'Orthogonal',

```

```

    segments: [{
      type: 'Orthogonal',
      // Defines the direction for the segment lines
      direction: 'Right',
      // Defines the length for the segment lines
      length: 50
    }],
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    }
  },
  {
    id: "connector2",
    type: 'Orthogonal',
    // Defines multile segemnts for the connectors
    segments: [{
      type: 'Orthogonal',
      direction: 'Bottom',
      length: 150
    },
    {
      type: 'Orthogonal',
      direction: 'Right',
      length: 150
    }
  ],
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    sourcePoint: {
      x: 300,
      y: 100
    },
  },

```

```

        targetPoint: {
            x: 400,
            y: 200
        }
    }
};

function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            connectors={connectors}
        />
    );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Note: You need to mention the segment type as same as what you mentioned in connector type. There should be no contradiction between connector type and segment type.

Avoid overlapping

Orthogonal segments are automatically re-routed, in order to avoid overlapping with the source and target nodes. The following preview illustrates how orthogonal segments are re-routed.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, PortVisibility } from "@syncfusion/ej2-react-diagrams";
let nodeport = {
    style: {
        strokeColor: '#366F8C',
        fill: '#366F8C'
    }
};
nodeport.shape = 'Circle';
nodeport.visibility = PortVisibility.Visible;
nodeport.id = 'port';
nodeport.offset = {
    x: 0,
    y: 0.5
};
let port2 = {
    style: {
        strokeColor: '#366F8C',
        fill: '#366F8C'
    }
};
port2.offset = {
    x: 0,
    y: 0.5
};

```



```

port2.id = 'port1';
port2.visibility = PortVisibility.Visible;
port2.shape = 'Circle';
let nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  ports: [nodeport]
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  ports: [port2]
},
];
let connectors = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  type: 'Orthogonal',
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  sourceID: 'node',
  targetID: 'node1',
  sourcePortID: 'port',
  targetPortID: 'port1'
}];
function App() {
  return (<DiagramComponent id="container" width={900} height={900}
nodes={nodes} connectors={connectors} getNodeDefaults={(node) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}/>);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  ConnectorModel,
  NodeModel,
  DiagramComponent,
  BasicShapeModel,
  PointPortModel,
  Diagram,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
let nodeport: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
};
nodeport.shape = 'Circle';
nodeport.visibility = PortVisibility.Visible
nodeport.id = 'port';
nodeport.offset = {
  x: 0,
  y: 0.5
};
let port2: PointPortModel = {
  style: {
    strokeColor: '#366F8C',
    fill: '#366F8C'
  }
}
port2.offset = {
  x: 0,
  y: 0.5
};
port2.id = 'port1';
port2.visibility = PortVisibility.Visible
port2.shape = 'Circle';
let nodes: NodeModel[] = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  ports: [nodeport]
},
{
  id: 'node1',
  width: 100,
  height: 100,
```

```

        offsetX: 300,
        offsetY: 100,
        ports: [port2]
      },
    ];
let connectors: ConnectorModel[] = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  type: 'Orthogonal',
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  sourceID: 'node',
  targetID: 'node1',
  sourcePortID: 'port',
  targetPortID: 'port1'
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={900}
      height={900}
      nodes={nodes}
      connectors={connectors}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

How to customize Orthogonal Segment Thumb Shape

The orthogonal connector has a number of segments in between the source and the target point. The segments are rendered with the default shape rhombus. Now, the option has been provided to change the segment thumb shape using the [segmentThumbShape](#) property. The predefined shapes provided are as follows:

- Rhombus
- Square
- Rectangle
- Ellipse
- Arrow
- Diamond
- OpenArrow
- Circle
- Fletch
- OpenFetch
- IndentedArrow
- OutdentedArrow
- DoubleArrow

You can customize the style of the thumb shape by overriding the class e-orthogonal-thumb.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram } from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connector2 = {};
connector2.id = 'connector2';
// Define the type of the segment
connector2.type = 'Orthogonal';
connector2.sourcePoint = { x: 250, y: 250 };
connector2.targetPoint = { x: 350, y: 350 };
connector2.segments = [
  {
    type: 'Orthogonal',
    // Defines the direction for the segment lines
    direction: "Right",
    // Defines the length for the segment lines
    length: 70
  },
  {
    type: 'Orthogonal',
    direction: "Bottom",
    length: 20
  }
];
function App() {
  return (<DiagramComponent id="container" width={'900px'}
    height={'500px'} connectors={[connector2]} getConnectorDefaults={(connector)
=> {
```

```

        connector.constraints =
            ConnectorConstraints.Default |
ConnectorConstraints.DragSegmentThumb;
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

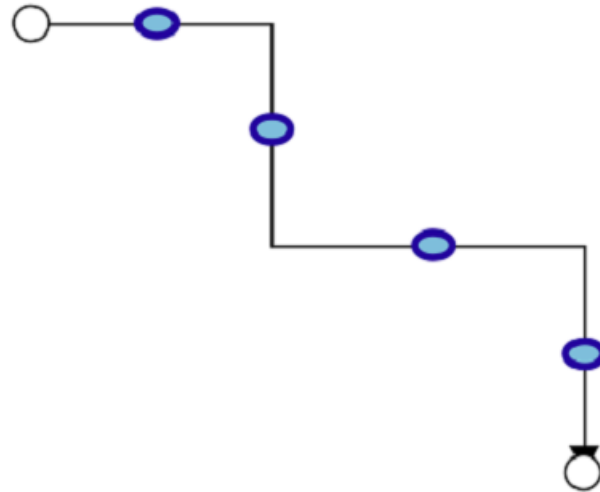
```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, ConnectorModel, ConnectorEditing, DiagramComponent,
ConnectorConstraints } from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connector2: ConnectorModel = {};
connector2.id = 'connector2';
// Define the type of the segment
connector2.type = 'Orthogonal';
connector2.sourcePoint = { x: 250, y: 250 };
connector2.targetPoint = { x: 350, y: 350 };
connector2.segments = [
    {
        type: 'Orthogonal',
        // Defines the direction for the segment lines
        direction: "Right",
        // Defines the length for the segment lines
        length: 70 },
    {
        type: 'Orthogonal',
        direction: "Bottom",
        length: 20 }];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'900px'}
            height={'500px'}
            connectors={[connector2]}
            getConnectorDefaults=(connector) => {
                connector.constraints =
                    ConnectorConstraints.Default |
ConnectorConstraints.DragSegmentThumb;
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```



Use the following CSS to customize the segment thumb shape.

```
`scss
.e-diagram-endpoint-handle {
  fill: rgb(126, 190, 219);
  stroke: #24039e;
  stroke-width: 3px;
}
```

Bezier

Bezier segments are used to create curve segments and the curves are configurable either with the control points or with vectors.

To create a bezier segment, the [segment.type](#) is set as `bezier` and need to specify [type](#) for the connector. The following code example illustrates how to create a default bezier segment.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
```

```

    id: 'connector1',
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    type: 'Bezier',
    segments: [{
      // Defines the type of the segment
      type: 'Bezier',
    }],
    sourcePoint: {
      x: 50,
      y: 100
    },
    targetPoint: {
      x: 150,
      y: 200
    },
  },
  });
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  type: 'Bezier',

```

```

    segments: [{
      // Defines the type of the segment
      type: 'Bezier',
    }],
    sourcePoint: {
      x: 50,
      y: 100
    },
    targetPoint: {
      x: 150,
      y: 200
    },
  },
  ]];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The [point1](#) and [point2](#) properties of bezier segment enable you to set the control points. The following code example illustrates how to configure the bezier segments with control points.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [
  {
    id: 'connector3',
    type: 'Bezier',
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    segments: [{
      type: 'Bezier',
      // First control point: an absolute position from the page
      origin
      point1: {
        x: 100,
        y: 100

```



```

        },
        // Second control point: an absolute position from the page
        origin
        point2: {
            x: 200,
            y: 200
        }
    ]],
    sourcePoint: {
        x: 100,
        y: 200
    },
    targetPoint: {
        x: 200,
        y: 100
    },
},
];
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [
    {
        id: 'connector3',
        type: 'Bezier',
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7',
            strokeWidth: 2
        },
        targetDecorator: {
            style: {
                fill: '#6BA5D7',
                strokeColor: '#6BA5D7'
            }
        },
        segments: [{
            type: 'Bezier',
            // First control point: an absolute position from the page
            origin
            point1: {
                x: 100,
                y: 100
            }
        }
    ]
}

```

```

        },
        // Second control point: an absolute position from the page
        origin
        point2: {
            x: 200,
            y: 200
        }
    }],
    sourcePoint: {
        x: 100,
        y: 200
    },
    targetPoint: {
        x: 200,
        y: 100
    },
    },
    },
    ],
    ];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            connectors={connectors}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The [vector1](#) and [vector2](#) properties of bezier segment enable you to define the vectors. The following code illustrates how to configure a bezier curve with vectors.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
    id: 'connector2',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    },
    // Defines the type of the segment
    type: 'Bezier',
    segments: [{
        type: 'Bezier',

```

```

// Length and angle between the source point and the first
control point
    vector1: {
        distance: 100,
        angle: 90
    },
// Length and angle between the target point and the second
control point
    vector2: {
        distance: 45,
        angle: 270
    }
}],
sourcePoint: {
    x: 100,
    y: 100
},
targetPoint: {
    x: 200,
    y: 200
},
}],
});
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
    id: 'connector2',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    // Defines the type of the segment
    type: 'Bezier',
    segments: [{
        type: 'Bezier',

```

```

    // Length and angle between the source point and the first control
    point
    vector1: {
        distance: 100,
        angle: 90
    },
    // Length and angle between the target point and the second control
    point
    vector2: {
        distance: 45,
        angle: 270
    }
  ]],
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
},
]);
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Avoid overlapping with bezier

By default, when there are no segments defined for a bezier connector, the bezier segments will be created automatically and routed in such a way that avoids overlapping with the source and target nodes.

`ts

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel, ConnectorModel, ConnectorEditing, ConnectorConstraints
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);

```

```
let nodes: NodeModel[] = [{
  id: 'Start',
  offsetX: 250,
  offsetY: 150,
  annotations: [{ content: 'Start' }]
},
{
  id: 'End',
  offsetX: 450,
  offsetY: 200,
  annotations: [{ content: 'End' }]
}];
let connectors: ConnectorModel[] = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: { shape: 'None' },
  // ID of the source and target nodes
  sourceID: "Start",
  targetID: "End",
  type: 'Bezier'
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
```

```

getNodeDefaults=((node: NodeModel) => {
  node.height = 100;
  node.width = 100;
  node.shape = { type: 'Basic', shape: 'Rectangle' };
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
})
getConnectorDefaults=((connector: ConnectorModel) => {
  connector.constraints =
  ConnectorConstraints.Default | ConnectorConstraints.DragSegmentThumb;
})
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Also, the intermediate point of two adjacent bezier segments can be edited interactively based on the `bezierSettings.segmentEditOrientation` property of the connector class.

[How to interact with the bezier segments efficiently](#)

While interacting with multiple bezier segments, maintain their control points at the same distance and angle by using the `bezierSettings.smoothness` property of the connector class.

BezierSmoothness value	Description
----- -----	
SymmetricDistance	Both control points of adjacent segments will be at the same distance when any one of them is editing.
SymmetricAngle	Both control points of adjacent segments will be at the same angle when any one of them is editing.
Default	Both control points of adjacent segments will be at the same angle and same distance when any one of them is editing.
None	Segment's control points are interacted independently from each other.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```
import {
  Diagram,
  DiagramComponent,
  NodeModel, ConnectorModel, ConnectorEditing, ConnectorConstraints
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let nodes: NodeModel[] = [{
  id: 'Start',
  offsetX: 250,
  offsetY: 150,
  annotations: [{ content: 'Start' }],
  ports: [{
    id: 'StartPort',
    visibility: PortVisibility.Visible,
    shape: 'Circle',
    offset: { x: 1, y: 0.5 },
    style: { strokeColor: '#366F8C', fill: '#366F8C' }
  }]
},
{
  id: 'End',
  offsetX: 450,
  offsetY: 200,
  annotations: [{ content: 'End' }],
  ports: [{
    id: 'EndPort',
    visibility: PortVisibility.Visible,
    shape: 'Circle',
    offset: { x: 0, y: 0.5 },
    style: { strokeColor: '#366F8C', fill: '#366F8C' }
  }]
}];
let connectors: ConnectorModel[] = [{
```

```
id: "connector1",
style: {
  strokeColor: '#6BA5D7',
  fill: '#6BA5D7',
  strokeWidth: 2
},
targetDecorator: { shape: 'None' },
// ID of the source and target nodes
sourceID: "Start",
sourcePortID: "StartPort",
targetID: "End",
targetPortID: "EndPort",
type: 'Bezier',
// Configuring settings for bezier interactions
bezierSettings = { smoothness: BezierSmoothness.SymmetricAngle }
});
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.shape = { type: 'Basic', shape: 'Rectangle' };
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
      getConnectorDefaults={(connector: ConnectorModel) => {
```



```

connector.constraints =
ConnectorConstraints.Default | ConnectorConstraints.DragSegmentThumb;
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Also, the visibility of control points can be controlled using the `bezierSettings.controlPointsVisibility` property of the connector class.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, ConnectorEditing,
ConnectorConstraints, PortVisibility, ControlPointsVisibility } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let nodes = [{
  id: 'Start',
  offsetX: 250,
  offsetY: 150,
  annotations: [{ content: 'Start' }],
  ports: [{
    id: 'StartPort',
    visibility: PortVisibility.Visible,
    shape: 'Circle',
    offset: { x: 1, y: 0.5 },
    style: { strokeColor: '#366F8C', fill: '#366F8C' }
  }]
},
{
  id: 'End',
  offsetX: 450,
  offsetY: 200,
  annotations: [{ content: 'End' }],
  ports: [{
    id: 'EndPort',
    visibility: PortVisibility.Visible,
    shape: 'Circle',
    offset: { x: 0, y: 0.5 },
    style: { strokeColor: '#366F8C', fill: '#366F8C' }
  }]
}
];
let connectors = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',

```

```

        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: { shape: 'None' },
    // ID of the source and target nodes
    sourceID: "Start",
    sourcePortID: "StartPort",
    targetID: "End",
    targetPortID: "EndPort",
    type: 'Bezier',
    // Configuring settings for bezier interactions
    bezierSettings : { controlPointsVisibility:
ControlPointsVisibility.Source | ControlPointsVisibility.Target }
    }];
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={nodes} connectors={connectors} getNodeDefaults={(node) => {
    node.height = 100;
    node.width = 100;
    node.shape = { type: 'Basic', shape: 'Rectangle' };
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
}} getConnectorDefaults={(connector) => {
    connector.constraints =
ConnectorConstraints.Default |
ConnectorConstraints.DragSegmentThumb;
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,

    NodeModel, ConnectorModel, ConnectorEditing, ConnectorConstraints, PortVisibilit
y, ControlPointsVisibility
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let nodes: NodeModel[] = [{
    id: 'Start',
    offsetX: 250,
    offsetY: 150,
    annotations: [{ content: 'Start' }],
    ports: [{
        id: 'StartPort',
        visibility: PortVisibility.Visible,
        shape: 'Circle',
        offset: { x: 1, y: 0.5 },

```

```

        style: { strokeColor: '#366F8C', fill: '#366F8C' }
      }}
    },
    {
      id: 'End',
      offsetX: 450,
      offsetY: 200,
      annotations: [{ content: 'End' }],
      ports: [{
        id: 'EndPort',
        visibility: PortVisibility.Visible,
        shape: 'Circle',
        offset: { x: 0, y: 0.5 },
        style: { strokeColor: '#366F8C', fill: '#366F8C' }
      }]
    }
  ]];
let connectors: ConnectorModel[] = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: { shape: 'None' },
  // ID of the source and target nodes
  sourceID: "Start",
  sourcePortID: "StartPort",
  targetID: "End",
  targetPortID: "EndPort",
  type: 'Bezier',
  // Configuring settings for bezier interactions
  bezierSettings : { controlPointsVisibility:
ControlPointsVisibility.Source | ControlPointsVisibility.Target }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.shape = { type: 'Basic', shape: 'Rectangle' };
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
      getConnectorDefaults={(connector: ConnectorModel) => {
        connector.constraints =
          ConnectorConstraints.Default |
ConnectorConstraints.DragSegmentThumb;
      }}
    />
  );
};

```

```

}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Decorator

- Starting and ending points of a connector can be decorated with some customizable shapes like arrows, circles, diamond, or path. The connection end points can be decorated with the [sourceDecorator](#) and [targetDecorator](#) properties of the connector.
- The [shape](#) property of [sourceDecorator](#) allows to define the shape of the decorators. Similarly, the [shape](#) property of [targetDecorator](#) allows to define the shape of the decorators.
- To create custom shape for source decorator, use [pathData](#) property. Similarly, to create custom shape for target decorator, use [pathData](#) property.
- The following code example illustrates how to create decorators of various shapes.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  type: 'Straight',
  // Decorator shape- circle
  sourceDecorator: {
    shape: 'Circle',
    // Defines the style for the sourceDecorator
    style: {
      // Defines the strokeWidth for the sourceDecorator
      strokeWidth: 3,
      // Defines the strokeColor for the sourceDecorator
      strokeColor: 'red'
    },
  },
  // Decorator shape - Diamond
  targetDecorator: {
    // Defines the custom shape for the connector's target decorator
    shape: 'Custom',
    // Defines the path for the connector's target decorator
    pathData: 'M80.5,12.5 C80.5,19.127417 62.59139,24.5 40.5,24.5 C18.40861,24.5 0.5,19.127417 0.5,12.5' +
      'C0.5,5.872583 18.40861,0.5 40.5,0.5 C62.59139,0.5 80.5,5.872583 80.5,12.5 z',
    // defines the style for the target decorator
    style: {
      // Defines the strokeWidth for the targetDecorator
      strokeWidth: 3,
      // Defines the strokeColor for the sourceDecorator
      strokeColor: 'green',
      // Defines the opacity for the sourceDecorator
      opacity: .8
    },
  },
},
],

```

```

    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    }
  },
  {
    id: "connectors2",
    type: 'Straight',
    // Decorator shape - IndentedArrow
    sourceDecorator: {
      shape: 'IndentedArrow',
      style: {
        strokeWidth: 3,
        strokeColor: 'blue'
      },
    },
    // Decorator shape - OutdentedArrow
    targetDecorator: {
      shape: 'OutdentedArrow',
      style: {
        strokeWidth: 3,
        strokeColor: 'yellow'
      },
    },
    sourcePoint: {
      x: 400,
      y: 100
    },
    targetPoint: {
      x: 300,
      y: 200
    }
  }
  ]];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  type: 'Straight',

```

```

// Decorator shape- circle
sourceDecorator: {
  shape: 'Circle',
  // Defines the style for the sourceDecorator
  style: {
    // Defines the strokeWidth for the sourceDecorator
    strokeWidth: 3,
    // Defines the strokeColor for the sourceDecorator
    strokeColor: 'red'
  },
},
// Decorator shape - Diamond
targetDecorator: {
  // Defines the custom shape for the connector's target decorator
  shape: 'Custom',
  //Defines the path for the connector's target decorator
  pathData: 'M80.5,12.5 C80.5,19.127417 62.59139,24.5 40.5,24.5
C18.40861,24.5 0.5,19.127417 0.5,12.5' +
  'C0.5,5.872583 18.40861,0.5 40.5,0.5 C62.59139,0.5 80.5,5.872583
80.5,12.5 z',
  //defines the style for the target decorator
  style: {
    // Defines the strokeWidth for the targetDecorator
    strokeWidth: 3,
    // Defines the strokeColor for the sourceDecorator
    strokeColor: 'green',
    // Defines the opacity for the sourceDecorator
    opacity: .8
  },
},
sourcePoint: {
  x: 100,
  y: 100
},
targetPoint: {
  x: 200,
  y: 200
}
},
{
  id: "connectors2",
  type: 'Straight',
  // Decorator shape - IndentedArrow
  sourceDecorator: {
    shape: 'IndentedArrow',
    style: {
      strokeWidth: 3,
      strokeColor: 'blue'
    },
  },
  // Decorator shape - OutdentedArrow
  targetDecorator: {
    shape: 'OutdentedArrow',
    style: {
      strokeWidth: 3,
      strokeColor: 'yellow'
    },
  },
}

```

```

    },
    sourcePoint: {
      x: 400,
      y: 100
    },
    targetPoint: {
      x: 300,
      y: 200
    }
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Padding

Padding is used to leave the space between the Connector's end point and the object to where it is connected.

- The [sourcePadding](#) property of connector defines space between the source point and the source node of the connector.
- The [targetPadding](#) property of connector defines space between the end point and the target node of the connector.
- The following code example illustrates how to leave space between the connection end points and source and target nodes.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, ConnectorConstraints } from "@syncfusion/ej2-react-diagrams";
let nodes = [{
  id: 'node',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
},
{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
}

```

```

    }
  ];
  let connectors = [{
    // Name of the connector
    id: "connector1",
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    },
    targetDecorator: {
      style: {
        fill: '#6BA5D7',
        strokeColor: '#6BA5D7'
      }
    },
    // ID of the source and target nodes
    sourceID: "node",
    targetID: "node1",
    // Set Source Padding value
    sourcePadding: 20,
    // Set Target Padding value
    targetPadding: 20
  }];
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={nodes} connectors={connectors} getNodeDefaults={(node) => {
      node.height = 100;
      node.width = 100;
      node.shape = { type: 'Basic', shape: 'Rectangle' };
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }} getConnectorDefaults={(connector) => {
      connector.constraints =
        ConnectorConstraints.Default |
        ConnectorConstraints.DragSegmentThumb;
    }}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel, ConnectorConstraints
} from "@syncfusion/ej2-react-diagrams";
let nodes: NodeModel[] = [{
  id: 'node',

```



```

        width: 100,
        height: 100,
        offsetX: 100,
        offsetY: 100,
      },
      {
        id: 'node1',
        width: 100,
        height: 100,
        offsetX: 300,
        offsetY: 100,
      }
    ];
    let connectors: ConnectorModel[] = [{
      // Name of the connector
      id: "connector1",
      style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
      },
      targetDecorator: {
        style: {
          fill: '#6BA5D7',
          strokeColor: '#6BA5D7'
        }
      },
      // ID of the source and target nodes
      sourceID: "node",
      targetID: "node1",
      // Set Source Padding value
      sourcePadding: 20,
      // Set Target Padding value
      targetPadding: 20
    }];
    function App() {
      return (
        <DiagramComponent
          id="container"
          width={'100%'}
          height={'600px'}
          nodes={nodes}
          connectors={connectors}
          getNodeDefaults={(node: NodeModel) => {
            node.height = 100;
            node.width = 100;
            node.shape = { type: 'Basic', shape: 'Rectangle' };
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
          }}
          getConnectorDefaults={(connector: ConnectorModel) => {
            connector.constraints =
              ConnectorConstraints.Default |
              ConnectorConstraints.DragSegmentThumb;
          }}
        />

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

Hit padding

- The [hitPadding](#) property enables you to define the clickable area around the connector path. The default value for hitPadding is 10.
- The following code example illustrates how to specify hit padding for connector.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connectors = [{
  // Name of the connector
  id: "connector1",
  type: "Orthogonal",
  //set hit padding
  hitPadding: 50,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 }
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={nodes} connectors={connectors}
  getConnectorDefaults=(connector) => {
    connector.constraints =
      ConnectorConstraints.Default |
      ConnectorConstraints.DragSegmentThumb;
  }/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel, ConnectorConstraints, ConnectorEditing
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(ConnectorEditing);
let connectors: ConnectorModel[] = [{
  // Name of the connector
  id: "connector1",
  type: "Orthogonal",
  //set hit padding
  hitPadding: 50,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: { x: 100, y: 100 },
  targetPoint: { x: 300, y: 300 }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
      getConnectorDefaults={(connector: ConnectorModel) => {
        connector.constraints =
          ConnectorConstraints.Default |
          ConnectorConstraints.DragSegmentThumb;
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Flip

The diagram Provides support to flip the connector. The [flip](#) is performed to give the mirrored image of the original element.

The flip types are as follows:

- HorizontalFlip

[Horizontal](#) is used to interchange the connector source and target x points.

- VerticalFlip

[Vertical](#) is used to interchange the connector source and target y points.

- Both

[Both](#) is used to interchange the source point as target point and target point as source point

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  // Name of the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  // Flip the connector in horizontal direction
  flip: "Horizontal"
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
```

```

    DiagramComponent,
    ConnectorModel
  } from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  // Name of the connector
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  // Flip the connector in horizontal direction
  flip: "Horizontal"
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Note: The flip is not applicable when the connectors connect in nodes

Bridging

Line bridging creates a bridge for lines to smartly cross over the other lines, at points of intersection. By default, [bridgeDirection](#) is set to top. Depending upon the direction given bridging direction appears.

Bridging can be enabled/disabled either with the `connector.constraints` or `diagram.constraints`. The following code example illustrates how to enable line bridging.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";

```

```

import { DiagramComponent, Inject, ConnectorBridging, DiagramConstraints }
from "@syncfusion/ej2-react-diagrams";
let model = [{
  id: 'Transaction',
  width: 150,
  height: 60,
  offsetX: 300,
  offsetY: 60,
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  },
  annotations: [{
    id: 'label1',
    content: 'Start Transaction',
    offset: {
      x: 0.5,
      y: 0.5
    }
  }],
},
{
  id: 'Verification',
  width: 150,
  height: 60,
  offsetX: 300,
  offsetY: 250,
  shape: {
    type: 'Flow',
    shape: 'Process'
  },
  annotations: [{
    id: 'label2',
    content: 'Verification',
    offset: {
      x: 0.5,
      y: 0.5
    }
  }],
}];
let connectors = [{
  id: 'connector1',
  type: 'Straight',
  sourceID: 'Transaction',
  targetID: 'Verification'
},
{
  id: 'connector2',
  type: 'Straight',
  sourcePoint: {
    x: 200,
    y: 130
  },
  targetPoint: {
    x: 400,
    y: 130
  }
}

```

```

    },
    {
      id: 'connector3',
      type: 'Straight',
      sourcePoint: {
        x: 200,
        y: 170
      },
      targetPoint: {
        x: 400,
        y: 170
      }
    }
  ]];
// Enables bridging for every connector added in the model
function App() {
  return (<DiagramComponent id="container" width={'100%'}
  getConnectorDefaults=(obj) => {
    obj.style.strokeColor = '#6BA5D7';
    obj.style.fill = '#6BA5D7';
    obj.style.strokeWidth = 2;
    obj.targetDecorator.style.fill = '#6BA5D7';
    obj.targetDecorator.style.strokeColor = '#6BA5D7';
    return obj;
  }) getNodeDefaults=(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }) nodes={model} connectors={connectors} height={'600px'}
  // Enables the bridging constraints for the connector
  constraints={DiagramConstraints.Default | DiagramConstraints.Bridging}
  connectors={connectors}>
    <Inject services={[ConnectorBridging]}/>{' '}
  </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorBridging,
  DiagramConstraints,
  ConnectorModel,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let model: NodeModel[] = [{
  id: 'Transaction',

```

```

width: 150,
height: 60,
offsetX: 300,
offsetY: 60,
shape: {
  type: 'Flow',
  shape: 'Terminator'
},
annotations: [{
  id: 'label1',
  content: 'Start Transaction',
  offset: {
    x: 0.5,
    y: 0.5
  }
}],
},
{
  id: 'Verification',
  width: 150,
  height: 60,
  offsetX: 300,
  offsetY: 250,
  shape: {
    type: 'Flow',
    shape: 'Process'
  },
  annotations: [{
    id: 'label2',
    content: 'Verification',
    offset: {
      x: 0.5,
      y: 0.5
    }
  }]
}];
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourceID: 'Transaction',
  targetID: 'Verification'
},
{
  id: 'connector2',
  type: 'Straight',
  sourcePoint: {
    x: 200,
    y: 130
  },
  targetPoint: {
    x: 400,
    y: 130
  }
},
{
  id: 'connector3',
  type: 'Straight',

```



```

    sourcePoint: {
      x: 200,
      y: 170
    },
    targetPoint: {
      x: 400,
      y: 170
    }
  }
}];
// Enables bridging for every connector added in the model
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      getConnectorDefaults={(obj: ConnectorModel): ConnectorModel => {
        obj.style.strokeColor = '#6BA5D7';
        obj.style.fill = '#6BA5D7';
        obj.style.strokeWidth = 2;
        obj.targetDecorator.style.fill = '#6BA5D7';
        obj.targetDecorator.style.strokeColor = '#6BA5D7';
        return obj;
      }}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
      nodes={node1}
      connectors={connectors}
      height={'600px'}
      // Enables the bridging constraints for the connector
      constraints={DiagramConstraints.Default | DiagramConstraints.Bridging}
      connectors={connectors}
    >
      <Inject services={[ConnectorBridging]} />{' '}
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Note: You need to inject connector bridging module into the diagram.

The [bridgeSpace](#) property of connectors can be used to define the width for line bridging.

Limitation: Bezier segments do not support bridging.

Corner radius

Corner radius allows to create connectors with rounded corners. The radius of the rounded corner is set with the [cornerRadius](#) property.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let nodes = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 350,
}
];
let connectors = [{
  id: "connector1",
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  // Sets the radius for the rounded corner
  cornerRadius: 10,
  sourceID: 'node1',
  targetID: 'node2',
  segments: [{
    type: 'Orthogonal',
    direction: 'Right',
    length: 50
  }],
}],
});
function App() {
  return (<DiagramComponent id="container" width={'100%'}
  getNodeDefaults=(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }) nodes={nodes} connectors={connectors} height={'600px'}
  >
  </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 350,
}
];
let connectors: ConnectorModel[] = [{
  id: "connector1",
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  // Sets the radius for the rounded corner
  cornerRadius: 10,
  sourceID: 'node1',
  targetID: 'node2',
  segments: [{
    type: 'Orthogonal',
    direction: 'Right',
    length: 50
  }],
}],
}];
function App() {
  return (
    <DiagramComponent
      id="container"
```

```

width={ '100%' }
getNodeDefaults={ (node: NodeModel) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}
nodes={nodes}
connectors={connectors}
height={ '600px' }
>
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Appearance

- The connector's [strokeWidth](#), [strokeColor](#), [strokeDashArray](#), and [opacity](#) properties are used to customize the appearance of the connector segments.
- The [visible](#) property of the connector enables or disables the visibility of connector.
- Default values for all the connectors can be set using the `getConnectorDefaults` properties. For example, if all connectors have the same type or having the same property then such properties can be moved into `getConnectorDefaults`.

Segment appearance

The following code example illustrates how to customize the segment appearance.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  targetDecorator: {
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    }
  },
  style: {
    // Stroke color
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    // Stroke width of the line
    strokeWidth: 2,
    // Line style
    strokeDashArray: '2,2'
  }
}

```

```

    },
    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    },
    segments: [{
      type: 'Orthogonal',
      direction: 'Right',
      length: 50
    }],
  },
  {
    id: "connector2",
    // Set the visibility of the connector to false
    visible: false,
    targetDecorator: {
      style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
      }
    },
    style: {
      // Stroke color
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      // Stroke width of the line
      strokeWidth: 2,
      // Line style
      strokeDashArray: '2,2'
    },
    sourcePoint: {
      x: 300,
      y: 300
    },
    targetPoint: {
      x: 400,
      y: 400
    },
    segments: [{
      type: 'Orthogonal',
      direction: 'Right',
      length: 50
    }],
  }
];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}>
    // Define the default values for all the connector. Similary we can add
    all the properties
    getConnectorDefaults=(obj) => {
      obj.type = 'Orthogonal';
      return obj;
    }
  );
}

```

```

    }} connectors={connectors}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  targetDecorator: {
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth: 2
    }
  },
  style: {
    // Stroke color
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    // Stroke width of the line
    strokeWidth: 2,
    // Line style
    strokeDashArray: '2,2'
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  segments: [{
    type: 'Orthogonal',
    direction: 'Right',
    length: 50
  }],
},
{
  id: "connector2",
  // Set the visibility of the connector to false
  visible: false,
  targetDecorator: {
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',

```

```

        strokeWidth: 2
      },
    },
    style: {
      // Stroke color
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7',
      // Stroke width of the line
      strokeWidth: 2,
      // Line style
      strokeDashArray: '2,2'
    },
    sourcePoint: {
      x: 300,
      y: 300
    },
    targetPoint: {
      x: 400,
      y: 400
    },
    segments: [{
      type: 'Orthogonal',
      direction: 'Right',
      length: 50
    }],
  },
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Define the default values for all the connector. Similary we can add
      all the properties
      getConnectorDefaults=(obj: ConnectorModel): ConnectorModel => {
        obj.type = 'Orthogonal';
        return obj;
      }
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Decorator appearance

- The source decorator's [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#) properties are used to customize the color, width, and appearance of the decorator.
- To set the border stroke color, stroke width, and stroke dash array for the target decorator, use [strokeColor](#), [strokeWidth](#), and [strokeDashArray](#).
- To set the size for source and target decorator, use width and height property.

The following code example illustrates how to customize the appearance of the decorator.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  type: 'Straight',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  bridgeSpace: 20,
  // Customize the target decorator
  targetDecorator: {
    style: {
      // Fill color of the decorator
      fill: '#6BA5D7',
      // Stroke color of the decorator
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  type: 'Straight',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  }
}
```



```

    },
    bridgeSpace: 20,
    // Customize the target decorator
    targetDecorator: {
      style: {
        // Fill color of the decorator
        fill: '#6BA5D7',
        // Stroke color of the decorator
        strokeColor: '#6BA5D7'
      }
    },
    sourcePoint: {
      x: 100,
      y: 100
    },
    targetPoint: {
      x: 200,
      y: 200
    }
  }
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Interaction

- Diagram allows to edit the connectors at runtime. To edit the connector segments at runtime, refer to [Connection Editing](#).

Automatic line routing

Diagram provides additional flexibility to re-route the diagram connectors. A connector will frequently re-route itself when a shape moves next to it. The following screenshot illustrates how the connector automatically re-routes the segments.

- Dependency LineRouting module should be injected to the application as the following code snippet.

`ts

```
import { Diagram, LineRouting } from "@syncfusion/ej2-react-diagrams";
/
```

- Injecting the automatic line routing module.

```
*/
```

```
Diagram.Inject(LineRouting);
```

```
,
```

- Now, the line routing constraints must be included to the default diagram constraints to enable automatic line routing support like below.

```
`ts
```

```
/
```

- Initialize the Diagram

```
*/
```

```
<DiagramComponent constraints={DiagramConstraints.Default | DiagramConstraints.LineRouting} />
```

```
,
```

- The following code block shows how to create the diagram with specifying nodes, connectors, constraints, and necessary modules for line routing.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, LineRouting, DiagramConstraints } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(LineRouting);
/**
 * Diagram Default sample
 */
//Initializes the nodes for the diagram
let nodes = [
  { id: 'shapel', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 300, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [
  { id: 'connector', sourceID: 'shapel', targetID: 'shape2', type:
'Orthogonal' }
];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'700px'}
nodes={nodes} connectors={connectors} //Sets the default values of a node
constraints={DiagramConstraints.Default |
DiagramConstraints.LineRouting} getNodeDefaults=(node) => {
  node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } };
  return node;
})/>);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram, DiagramComponent, NodeModel, SnapConstraints, LineRouting,
  DiagramConstraints, NodeModel, ConnectorModel, ConnectorConstraints
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(LineRouting);
/**
 * Diagram Default sample
 */
//Initializes the nodes for the diagram
let nodes = [
  { id: 'shape1', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 300, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [
  { id: 'connector', sourceID: 'shape1', targetID: 'shape2', type:
'Orthogonal' }
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'700px'}
      nodes={nodes}
      connectors={connectors} //Sets the default values of a node
      constraints={DiagramConstraints.Default |
DiagramConstraints.LineRouting}
      getNodeDefaults=(node) => {
        node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } };
        return node;
      }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Chart-DataLabel</title>
  <meta charset="utf-8" />
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components"
/>
<meta name="author" content="Syncfusion" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
diagrams/styles/material.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<script src="systemjs.config.js"></script>
<style>
    #loader {
        color: #008cff;
        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    #diagram {
        display: block;
    }
</style>
</head>
<body>
    <div id='diagram'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

- In some situations, automatic line routing enabled diagram needs to ignore a specific connector from automatic line routing. So, in this case, auto routing feature can be disabled to the specific connector using the [constraints](#) property of the connector like the following code snippet.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, LineRouting, DiagramConstraints,
ConnectorConstraints } from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(LineRouting);
/**
 * Diagram Default sample

```

```

*/
//Initializes the nodes for the diagram
let nodes = [
  { id: 'shapel', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 350, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 },
  { id: 'shape4', offsetX: 300, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [
  { id: 'connector', sourceID: 'shapel', targetID: 'shape2', type:
'Orthogonal', annotations: [{ offset: .7, content: ' Routing \n enabled',
style: { fill: "white" } }] },
  { id: 'connector2', sourceID: 'shapel', targetID: 'shape2', annotations:
[{ offset: .6, content: ' Routing \n disabled', style: { fill: "white" } }],
type: 'Orthogonal', constraints: ConnectorConstraints.Default &
~ConnectorConstraints.InheritLineRouting }
];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'700px'}
nodes={nodes} connectors={connectors} //Sets the default values of a node
constraints={DiagramConstraints.Default |
DiagramConstraints.LineRouting} getNodeDefaults={(node) => {
  node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } };
  return node;
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram, DiagramComponent, NodeModel, SnapConstraints, LineRouting,
  DiagramConstraints, NodeModel, ConnectorModel, ConnectorConstraints
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(LineRouting);
/**
 * Diagram Default sample
 */
//Initializes the nodes for the diagram
let nodes = [
  { id: 'shapel', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 350, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 },
  { id: 'shape4', offsetX: 300, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [
  { id: 'connector', sourceID: 'shapel', targetID: 'shape2', type:
'Orthogonal', annotations: [{ offset: .7, content: ' Routing \n enabled',
style: { fill: "white" } }] },

```

```

    { id: 'connector2', sourceID: 'shape1', targetID: 'shape2', annotations:
    [{ offset: .6, content: ' Routing \n disabled', style: { fill: "white" } }],
    type: 'Orthogonal', constraints: ConnectorConstraints.Default &
    ~ConnectorConstraints.InheritLineRouting }
  ];
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={ '100%' }
        height={ '700px' }
        nodes={nodes}
        connectors={connectors} //Sets the default values of a node
        constraints={DiagramConstraints.Default |
DiagramConstraints.LineRouting}
        getNodeDefaults={ (node) => {
          node = { style: { strokeColor: '#6BA5D7', fill: '#6BA5D7' } };
          return node;
        }}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Chart-DataLabel</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components"
/>
  <meta name="author" content="Syncfusion" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></sc
ript>
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
diagrams/styles/material.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <script src="systemjs.config.js"></script>
  <style>
    #loader {
      color: #008cff;

```

```

        height: 40px;
        left: 45%;
        position: absolute;
        top: 45%;
        width: 30%;
    }
    #diagram {
        display: block;
    }
</style>
</head>
<body>
    <div id='diagram'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Constraints

- The [constraints](#) property of connector allows to enable/disable certain features of connectors.
- To enable or disable the constraints, refer [constraints](#).

The following code illustrates how to disable selection.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, ConnectorConstraints } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
    id: "connector1",
    // Disables selection constraints
    constraints: ConnectorConstraints.Default &
    ~ConnectorConstraints.Select,
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 100,
        y: 100
    },
    targetPoint: {
        x: 200,
        y: 200
    }
}];

```

```

    }
  }];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'700px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  DiagramConstraints,
  ConnectorConstraints,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: "connector1",
  // Disables selection constraints
  constraints: ConnectorConstraints.Default &
~ConnectorConstraints.Select,
  type: 'Straight',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'700px'}
      connectors={connectors}
    />
  );
}

```



```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Custom properties

- The [addInfo](#) property of connectors allow you to maintain additional information to the connectors.

```
`ts
```

```
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  // Defines the information about the connector
  addInfo: 'centralconnector',
  type: 'Straight',
  sourceID: 'Transaction',
  targetID: 'Verification'
}];
`
```

Stack order

The connectors [zIndex](#) property specifies the stack order of the connector. A connector with greater stack order is always in front of a connector with a lower stack order.

The following code illustrates how to render connector based on the stack order.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: 'connector1',
  // Defines the z-index value for the connector
  zIndex: 2,
  type: 'Straight',
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 300,
    y: 200
  }
},
{
  id: 'connector2',
  type: 'Straight',
  // Defines the z-index value for the connector
  zIndex: 1
}
];
```

```

        zIndex: 1,
        sourcePoint: {
            x: 100,
            y: 100
        },
        targetPoint: {
            x: 200,
            y: 200
        }
    }
  }];
function App() {
  return (<DiagramComponent id="container" width={'100%'}
getConnectorDefaults=(obj) => {
    obj.style.strokeColor = '#6BA5D7';
    obj.style.fill = '#6BA5D7';
    obj.style.strokeWidth = 2;
    obj.targetDecorator.style.fill = '#6BA5D7';
    obj.targetDecorator.style.strokeColor = '#6BA5D7';
    return obj;
  }) height={'600px'} connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  // Defines the z-index value for the connector
  zIndex: 2,
  type: 'Straight',
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 300,
    y: 200
  }
},
{
  id: 'connector2',
  type: 'Straight',
  // Defines the z-index value for the connector
  zIndex: 1,
  sourcePoint: {
    x: 100,

```

```

        y: 100
      },
      targetPoint: {
        x: 200,
        y: 200
      }
    }
  }];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      getConnectorDefaults=(obj) => {
        obj.style.strokeColor = '#6BA5D7';
        obj.style.fill = '#6BA5D7';
        obj.style.strokeWidth = 2;
        obj.targetDecorator.style.fill = '#6BA5D7';
        obj.targetDecorator.style.strokeColor = '#6BA5D7';
        return obj;
      }
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Line Routing

Line Routing is used to avoid the overlapping of nodes and connectors. Connectors are the most important elements of a diagram used to show the flow and relationship between shapes in the flowcharts, organizational charts, and hierarchy diagrams. Line Routing provides the flexibility to re-route the diagram connectors. A connector will frequently re-route itself when a shape moves next to it.

Note: You need to inject line routing module into the diagram. Line routing is applicable only for orthogonal segment of connector.

The following code illustrates how the automatic line routing is enabled for diagram.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, LineRouting, DiagramConstraints, } from
"@syncfusion/ej2-react-diagrams";
//Initializes the nodes for the diagram
let nodes = [
  { id: 'shape1', offsetX: 100, offsetY: 100, width: 120, height: 50 },
  { id: 'shape2', offsetX: 300, offsetY: 300, width: 120, height: 50 },
  { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [

```

```

    { id: 'connector', sourceID: 'shape1', targetID: 'shape2', type:
'Orthogonal' }
];
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'700px'}
nodes={nodes} connectors={connectors}
constraints={DiagramConstraints.Default | DiagramConstraints.LineRouting}
getNodeDefaults={(node) => {
    node = { style: { strokeColor: '#65B091', fill: '#65B091' } } };
    return node;
}}>
    <Inject services={[LineRouting]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    LineRouting,
    DiagramConstraints,
    ConnectorModel,
    NodeModel,
} from "@syncfusion/ej2-react-diagrams";
//Initializes the nodes for the diagram
let nodes = [
    { id: 'shape1', offsetX: 100, offsetY: 100, width: 120, height: 50 },
    { id: 'shape2', offsetX: 300, offsetY: 300, width: 120, height: 50 },
    { id: 'shape3', offsetX: 150, offsetY: 200, width: 120, height: 50 }
];
//Initializes the connector for the diagram
let connectors = [
    { id: 'connector', sourceID: 'shape1', targetID: 'shape2', type:
'Orthogonal' }
];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'700px'}
            nodes={nodes}
            connectors={connectors}
            constraints={DiagramConstraints.Default |
DiagramConstraints.LineRouting}
            getNodeDefaults={(node) => {
                node = { style: { strokeColor: '#65B091', fill: '#65B091' } } };
                return node;
            }}
        >

```

```

    }}
    >
    <Inject services={[LineRouting]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Enable Connector Splitting

The connectors are used to create a link between two points, ports, or nodes to represent the relationship between them. Split the connector between two nodes when dropping a new node onto an existing connector and create a connection between the new node and existing nodes by setting the [enableConnectorSplit](#) as true. The default value of the [enableConnectorSplit](#) is false.

The following code illustrates how to split the connector and create a connection with new node.

INDEX.JSX

```

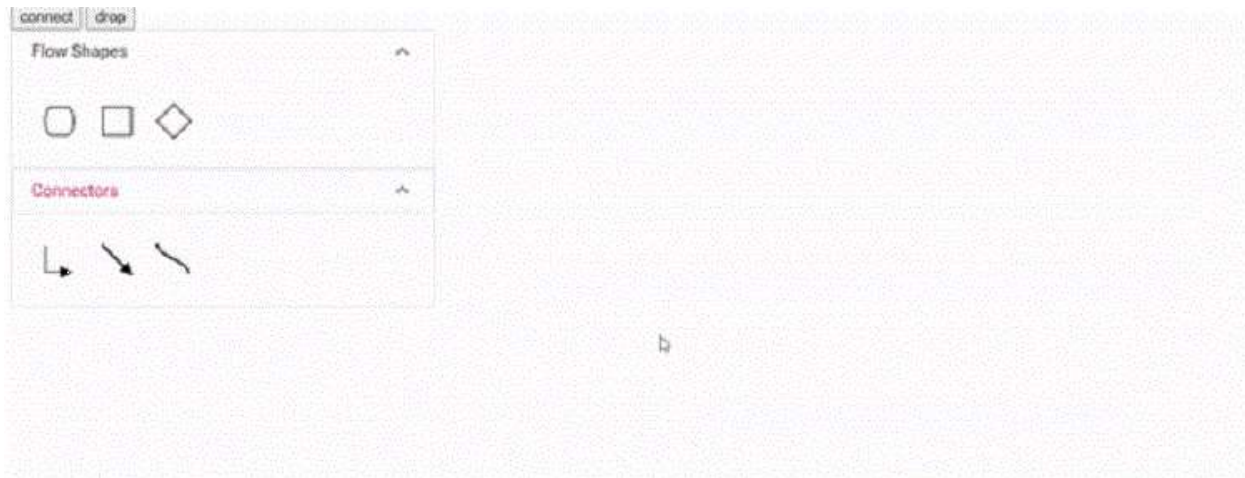
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { ConnectorConstraints, DiagramComponent } from '@syncfusion/ej2-react-diagrams';
let nodes = [{
  id: 'node1',
  width: 100, height: 100,
  offsetX: 150, offsetY: 150,
  annotations: [{ content: 'node1' }]
},
{
  id: 'node2',
  width: 100, height: 100,
  offsetX: 650, offsetY: 150,
  annotations: [{ content: 'node2' }]
},
{
  id: 'node3',
  width: 100, height: 100,
  offsetX: 490, offsetY: 290,
  annotations: [{ content: 'node3' }]
},
];
let connectors = [{
  id: 'connector1', sourceID: "node1", targetID: "node2",
  constraints: ConnectorConstraints.Default |
ConnectorConstraints.AllowDrop
}
];
function App() {
  return (<DiagramComponent id="container" width={1500} height={1000}
enableConnectorSplit={true} nodes={nodes} connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

```
{% enddraw %}
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram, NodeModel, ConnectorModel, ConnectorConstraints, DiagramComponent
} from '@syncfusion/ej2-react-diagrams';
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100, height: 100,
  offsetX: 150, offsetY: 150,
  annotations: [{ content: 'node1' }]
},
{
  id: 'node2',
  width: 100, height: 100,
  offsetX: 650, offsetY: 150,
  annotations: [{ content: 'node2' }]
},
{
  id: 'node3',
  width: 100, height: 100,
  offsetX: 490, offsetY: 290,
  annotations: [{ content: 'node3' }]
},
];
let connectors: ConnectorModel[] = [{
  id: 'connector1', sourceID: "node1", targetID: "node2",
  constraints: ConnectorConstraints.Default |
ConnectorConstraints.AllowDrop
}
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={1500}
      height={1000}
      enableConnectorSplit={true}
      nodes={nodes}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```



See Also

- [How to add annotations to the connector](#)
- [How to enable/disable the behavior of the node](#)
- [How to add connectors to the symbol palette](#)
- [How to perform the interaction on the connector](#)
- [How to create diagram connectors using drawing tools](#)

Group in React Diagram component

Group is used to cluster multiple nodes and connectors into a single element. It acts like a container for its children (nodes, groups, and connectors). Every change made to the group also affects the children. Child elements can be edited individually.

Create group

Add group when initializing diagram

A group can be added to the diagram model through [nodes](#) collection. To define an object as group, add the child objects to the [children](#) collection of the group. The following code illustrates how to create a group node.

- The [padding](#) property of a group node defines the spacing between the group node's edges and its children.
- While creating group, its child node need to be declared before the group declaration.
- Add a node to the existing group child by using the `diagram.group` method.
- The group's `diagram.unGroup` method is used to define whether the group can be ungrouped or not.
- A group can be added into a child of another group.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
```

```

    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    annotations: [{
      content: 'rectangle1'
    }]
  },
  {
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,
    annotations: [{
      content: 'rectangle2'
    }]
  },
  {
    id: "rectangle3",
    offsetX: 400,
    offsetY: 300,
    width: 100,
    height: 100,
    style: {
      fill: 'darkCyan',
      strokeWidth: 2
    },
    annotations: [{
      content: 'rectangle2'
    }]
  },
  // Grouping node 1 and node 2 into a single group
  {
    id: 'group',
    children: ['rectangle1', 'rectangle2'],
    padding: { left: 10, right: 10, top: 10, bottom: 10 }
  },
];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'1500px'} height={'600px'} nodes={nodes}
created={() => {
    diagramInstance.selectAll();
    // Adding the third node into the existing group
    diagramInstance.group();
  }} getNodeDefaults=(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```



```
root.render(<App />);
{% enddraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance:DiagramComponent;
let nodes: NodeModel[] = [{
  id: "rectangle1",
  offsetX: 100,
  offsetY: 100,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle1'
  }]
},
{
  id: "rectangle2",
  offsetX: 200,
  offsetY: 200,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle2'
  }]
},
{
  id: "rectangle3",
  offsetX: 400,
  offsetY: 300,
  width: 100,
  height: 100,
  style: {
    fill: 'darkCyan',
    strokeWidth: 2
  },
  annotations: [{
    content: 'rectangle2'
  }]
},
// Grouping node 1 and node 2 into a single group
{
  id: 'group',
  children: ['rectangle1', 'rectangle2'],
  padding:{left:10,right:10,top:10,bottom:10}
},
];
// initialize Diagram component
```

```
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'1500px'}
      height={'600px'}
      nodes={nodes}
      created={() => {
        diagramInstance.selectAll();
        // Adding the third node into the existing group
        diagramInstance.group();
      }}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

The following code illustrates how a `ungroup` at runtime.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: "rectangle1",
  offsetX: 100,
  offsetY: 100,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle1'
  }]
},
{
  id: "rectangle2",
  offsetX: 200,
  offsetY: 200,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle2'
  }]
},

```

```

    {
      id: 'group',
      children: ['rectangle1', 'rectangle2']
    },
  ];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'1500px'} height={'600px'} nodes={nodes}
getNodeDefaults={(node) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}} created={() => {
  diagramInstance.selectAll();
  // Ungroup the selected group into nodes
  diagramInstance.unGroup();
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance:DiagramComponent;
let nodes: NodeModel[] = [{
  id: "rectangle1",
  offsetX: 100,
  offsetY: 100,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle1'
  }]
},
{
  id: "rectangle2",
  offsetX: 200,
  offsetY: 200,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle2'
  }]
},
{

```

```

        id: 'group',
        children: ['rectangle1', 'rectangle2']
      },
    ];
    function App() {
      return (
        <DiagramComponent
          id="container"
          ref={(diagram) => (diagramInstance = diagram)}
          width={'1500px'}
          height={'600px'}
          nodes={nodes}
          getNodeDefaults={(node: NodeModel) => {
            node.height = 100;
            node.width = 100;
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
          }}
          created={() => {
            diagramInstance.selectAll();
            // Ungroup the selected group into nodes
            diagramInstance.unGroup();
          }}
        />
      );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);
  {% enddraw %}

```

Add group at runtime

A group node can be added at runtime by using the client-side method `diagram.add`.

The following code illustrates how a group node is added at runtime.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: "rectangle1",
  offsetX: 100,
  offsetY: 100,
  width: 100,
  height: 100,
  annotations: [{
    content: 'rectangle1'
  }]
},
{
  id: "rectangle2",
  offsetX: 200,

```

```

        offsetY: 200,
        width: 100,
        height: 100,
        annotations: [{
            content: 'rectangle2'
        }]
    }];
let group = {
    id: 'group2',
    children: ['rectangle1', 'rectangle2']
};
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'1500px'} height={'600px'}
getNodeDefaults={(node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }} nodes={nodes} created={() => {
        // Adds the group into the diagram
        diagramInstance.add(group);
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
    id: "rectangle1",
    offsetX: 100,
    offsetY: 100,
    width: 100,
    height: 100,
    annotations: [{
        content: 'rectangle1'
    }]
}],
{
    id: "rectangle2",
    offsetX: 200,
    offsetY: 200,
    width: 100,
    height: 100,

```

```

        annotations: [{
            content: 'rectangle2'
        }]
    }];
let group: NodeModel = {
    id: 'group2',
    children: ['rectangle1', 'rectangle2']
};
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'1500px'}
            height={'600px'}
            getNodeDefaults={(node: NodeModel) => {
                node.height = 100;
                node.width = 100;
                node.style.fill = '#6BA5D7';
                node.style.strokeColor = 'white';
                return node;
            }}
            nodes={nodes}
            created={() => {
                // Adds the group into the diagram
                diagramInstance.add(group);
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Add children To group at runtime

A childNode can be added to the specified Group at runtime by utilizing the client-side method `diagramInstance.addChildToGroup`.

This functionality is achieved by passing the group and existing children as arguments to the method.

The following code illustrates how a child node and a group node can be passed as arguments to the method and executed at runtime.

`html

```
diagramInstance.addChildToGroup(groupNode, childNode);
```

,

Remove children from group at runtime

A specific child from a group node can be removed at runtime by utilizing the client-side method `diagramInstance.removeChildFromGroup`.

This functionality is achieved by passing the group and its children as arguments to the method.

The following code illustrates how a child node is removed from a group at runtime.

```
`html
```

```
diagramInstance.removeChildFromGroup (groupNode, childNode);
```

```
,
```

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let node = {
  id: 'node1', width: 150, height: 100, offsetX: 100, offsetY: 100,
  annotations: [{ content: 'Node1' }]
};
let node2 = {
  id: 'node2', width: 80, height: 130, offsetX: 200, offsetY: 200,
  annotations: [{ content: 'Node2' }]
};
let group = {
  id: 'group1', children: ['node1', 'node2']
};
let node3 = {
  id: 'node3', width: 100, height: 100, offsetX: 300, offsetY: 300,
  annotations: [{ content: 'Node3' }]
};
document.getElementById('addChild').onclick = function () {
  diagramInstance.addChildToGroup(group, 'node3');
};
document.getElementById('removeChild').onclick = function () {
  diagramInstance.removeChildFromGroup(group, 'node3');
};
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={'1500px'} height={'600px'}
    nodes={ [node, node2, node3, group] }
  />);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let node: NodeModel = {
```

```

    id: 'node1', width: 150, height: 100, offsetX: 100, offsetY: 100,
    annotations: [{ content: 'Node1' }]
  };
  let node2: NodeModel = {
    id: 'node2', width: 80, height: 130, offsetX: 200, offsetY: 200,
    annotations: [{ content: 'Node2' }]
  };
  let group: NodeModel = {
    id: 'group1', children: ['node1', 'node2']
  };
  let node3: NodeModel = {
    id: 'node3', width: 100, height: 100, offsetX: 300, offsetY: 300,
    annotations: [{ content: 'Node3' }]
  };
  (document.getElementById('addChild') as any).onclick = function () {
    diagramInstance.addChildToGroup(group, 'node3');
  };
  (document.getElementById('removeChild') as any).onclick = function () {
    diagramInstance.removeChildFromGroup(group, 'node3');
  };
  function App() {
    return (
      <DiagramComponent
        id="container"
        ref={(diagram) => (diagramInstance = diagram)}
        width={'1500px'}
        height={'600px'}
        nodes={ [node,node2,node3,group] }
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

Container

Containers are used to automatically measure and arrange the size and position of the child elements in a predefined manner.

There are two types of containers available.

Canvas

- The canvas panel supports absolute positioning and provides the least layout functionality to its contained diagram elements.
- Canvas allows you to position its contained elements by using the margin and alignment properties.
- Rendering alone possible in canvas container.
- It allows elements to be either vertically or horizontally aligned.
- Child can be defined with the collection [canvas.children](#) property.
- Basic element can be defined with the collection of [basicElements](#).

The following code illustrates how to add canvas panel.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramElement, Canvas } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let canvas;
let child1;
let child2;
canvas = new Canvas();
canvas.pivot = {
  x: 0,
  y: 0
};
canvas.offsetX = 75;
canvas.offsetY = 75;
canvas.style.fill = '#6BA5D7';
child1 = new DiagramElement();
child1.width = 100;
child1.height = 100;
child1.margin.left = child1.margin.top = 10;
child2 = new DiagramElement();
child2.width = 100;
child2.height = 100;
child2.margin.left = 190;
child2.margin.top = 190;
canvas.children = [child1, child2];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} mode={'SVG'} width={'1000px'} height={'600px'}
    // Defines the basic elements
    basicElements={ [canvas]} />);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  DiagramElement,
  Canvas
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let canvas: Canvas;
let child1: DiagramElement;
let child2: DiagramElement;
canvas = new Canvas();
canvas.pivot = {
  x: 0,
  y: 0
};
```

```

canvas.offsetX = 75;
canvas.offsetY = 75;
canvas.style.fill = '#6BA5D7';
child1 = new DiagramElement();
child1.width = 100;
child1.height = 100;
child1.margin.left = child1.margin.top = 10;
child2 = new DiagramElement();
child2.width = 100;
child2.height = 100;
child2.margin.left = 190;
child2.margin.top = 190;
canvas.children = [child1, child2];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      mode={'SVG'}
      width={'1000px'}
      height={'600px'}
      // Defines the basic elements
      basicElements={[canvas]}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Stack

- Stack panel is used to arrange its children in a single line or stack order, either vertically or horizontally.
- It controls spacing by setting margin properties of child and padding properties of group. By default, a stack panel's [orientation](#) is vertical.

The following code illustrates how to add a stack panel.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, StackPanel, TextElement, } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: 'node5',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  }
}

```

```

    },
    annotations: [{
      content: 'Custom Template',
      offset: {
        y: 1
      },
      verticalAlignment: 'Top'
    }]
  },
],];
let count = 11;
let getTextElement = (text) => {
  let textElement = new TextElement();
  textElement.id = "text" + count;
  textElement.width = 50;
  textElement.height = 20;
  textElement.content = text;
  count++;
  return textElement;
};
let addRows = (column) => {
  column.children.push(getTextElement('Row1'));
  column.children.push(getTextElement('Row2'));
  column.children.push(getTextElement('Row3'));
  column.children.push(getTextElement('Row4'));
};
//Intialize Diagram Component
function App() {
  return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram) } width={900} height={900} nodes={nodes}
    getNodeDefaults={ (node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }} setNodeTemplate={ (obj, diagram) => {
      if (obj.id.indexOf('node5') !== -1) {
        // It will be replaced with grid panel
        let table = new StackPanel();
        table.orientation = 'Horizontal';
        table.padding.left;
        let column1 = new StackPanel();
        column1.children = [];
        column1.children.push(getTextElement('Column1'));
        addRows(column1);
        let column2 = new StackPanel();
        column2.children = [];
        column2.children.push(getTextElement('Column2'));
        addRows(column2);
        table.children = [column1, column2];
        return table;
      }
      return null;
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

```
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  StackPanel,
  TextElement,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance:DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'node5',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  annotations: [{
    content: 'Custom Template',
    offset: {
      y: 1
    },
    verticalAlignment: 'Top'
  }]
}], []];
let count: Number = 11;
let getTextElement: Function = (text: string) => {
  let textElement: TextElement = new TextElement();
  textElement.id = "text" + count;
  textElement.width = 50;
  textElement.height = 20;
  textElement.content = text;
  count++;
  return textElement;
};
let addRows: Function = (column: StackPanel) => {
  column.children.push(getTextElement('Row1'));
  column.children.push(getTextElement('Row2'));
  column.children.push(getTextElement('Row3'));
  column.children.push(getTextElement('Row4'));
};
//Intialize Diagram Component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={900}
    />
  );
}
```

```

height={900}
nodes={nodes}
getNodeDefaults={(node: NodeModel) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}
setNodeTemplate={(obj: NodeModel, diagram: Diagram): StackPanel => {
  if (obj.id.indexOf('node5') !== -1) {
    // It will be replaced with grid panel
    let table: StackPanel = new StackPanel();
    table.orientation = 'Horizontal';
    table.padding.left;
    let column1: StackPanel = new StackPanel();
    column1.children = [];
    column1.children.push(getTextElement('Column1'));
    addRows(column1);
    let column2: StackPanel = new StackPanel();
    column2.children = [];
    column2.children.push(getTextElement('Column2'));
    addRows(column2);
    table.children = [column1, column2];
    return table;
  }
  return null;
}}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Difference between a basic group and containers

| Group | Container |

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

| It arranges the child elements based on the child elements position and size properties. | Each container has a predefined behavior to measure and arrange its child elements. Canvas and stack containers are supported in the diagram. |

| The Padding, Min, and Max Size properties are not applicable for basic group. | It is applicable for container. |

| The Children's margin and alignment properties are not applicable for basic group. | It is applicable for container. |

Interaction

You can edit the group and its children at runtime. For more information about how to interact with a group, refer to [Edit Groups](#).

See Also

- [How to add annotations to the node](#)
- [How to add ports to the node](#)
- [How to enable/disable the behavior of the node](#)
- [How to add nodes to the symbol palette](#)
- [How to create diagram nodes using drawing tools](#)
- [How to perform the interaction on the group](#)

Swim lane in React Diagram component

Swimlane is a type of diagram nodes, which is typically used to visualize the relationship between a business process and the department responsible for it by focusing on the logical relationships between activities.

Create a swimlane

To create a swimlane, the type of shape should be set as [swimlane](#). By Default swimlane's are arranged vertically.

The following code example illustrates how to define a swimlane object.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    // Set the node type as swimlane
    type: 'SwimLane',
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    // Set the node type as swimlane
    type: 'SwimLane',
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Headers

Header was the primary element for swimlanes. The [header](#) property of swimlane allows you to define its textual description and to customize its appearance.

Note: By using this header, the swimlane interaction will be performed, like selection, dragging, etc.

The following code example illustrates how to define a swimlane header.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {

```



```

        annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: 'transparent' } },
        height: 50, style: { fontSize: 11 },
      },
      lanes: [
        {
          id: 'stackCanvas1',
          height: 100,
        },
      ],
      phases: [
        {
          id: 'phase1', offset: 170,
          header: { annotation: { content: 'Phase' } }
        },
      ],
      phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  ]];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Customization of headers

The height and width of swimlane header can be customized with [width](#) and [height](#) properties of swimlane header. set fill color of header by using the [style](#) property. The orientation of swimlane can be customized with the [orientation](#) property of the header.

Note: By default the swimlane orientation has Horizontal.

The following code example illustrates how to customize the swimlane header.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // customize the swimlane header

```

```

        header: {
            annotation: { content: 'SALES PROCESS FLOW CHART', },
            height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
        },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
            },
        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        // customize the swimlane header
        header: {
            annotation: { content: 'SALES PROCESS FLOW CHART', },
            height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
        },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
            },
        ],
    },
}];

```

```

        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  ]];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Dynamic customization of swimlane header

You can customize the swimlane header style and text properties dynamically. The following code illustrates how to dynamically customize the lane header.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'SALES PROCESS FLOW CHART', },
      height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
    },
  },
  lanes: [
    {
      id: 'stackCanvas1',
      height: 100,
    },
  ],
  phases: [
    {

```

```

        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
});
// initialize Diagram component
let diagramInstance;
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
created={() => {
    // change the swimlane header style dynamically
    diagramInstance.nodes[0].shape.header.style.fill = 'red';
    diagramInstance.dataBind();
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'SALES PROCESS FLOW CHART', },
      height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
    },
  },
  lanes: [
    {
      id: 'stackCanvas1',
      height: 100,
    },
  ],
  phases: [
    {
      id: 'phase1', offset: 170,
      header: { annotation: { content: 'Phase' } }
    }
  ]
}

```

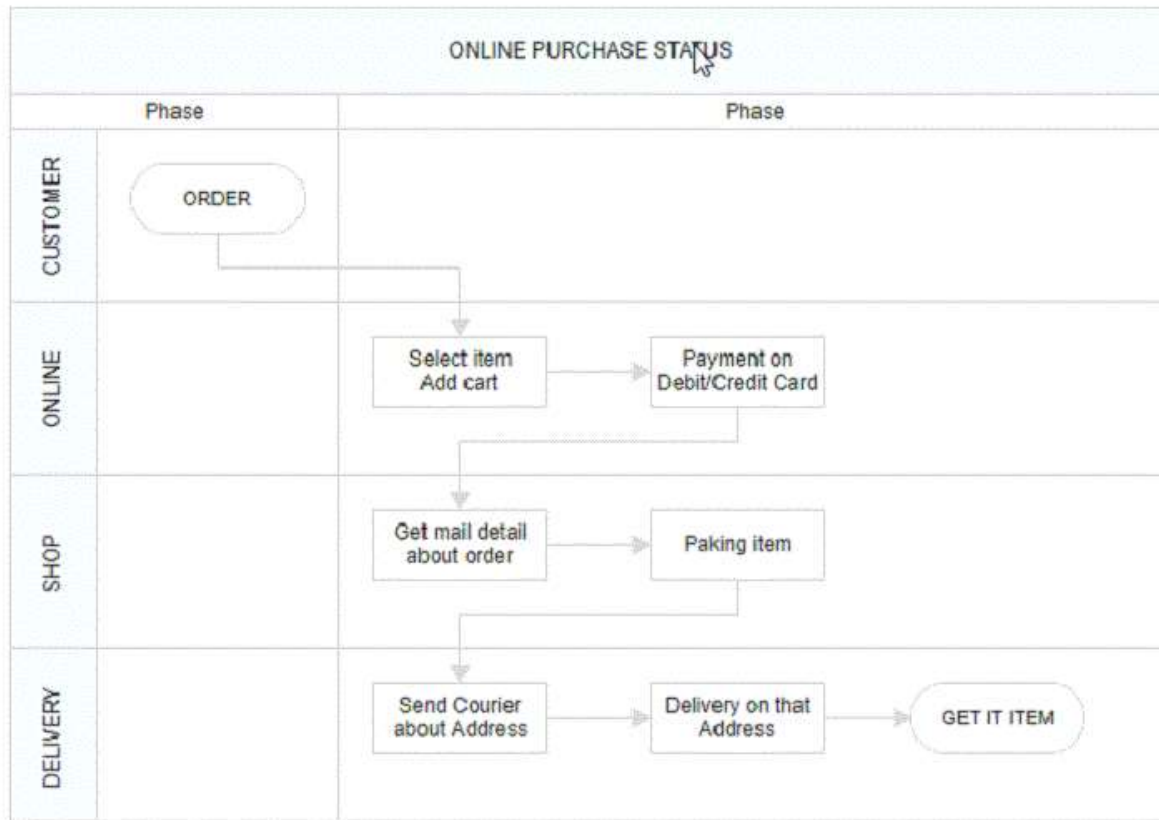
```

        },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  ]];
  // initialize Diagram component
  let diagramInstance: DiagramComponent;
  function App() {
    return (
      <DiagramComponent
        id="container"
        ref={(diagram) => (diagramInstance = diagram)}
        width={'100%'}
        height={'600px'}
        nodes={node}
        created = { () => {
          // change the swimlane header style dynamically
          diagramInstance.nodes[0].shape.header.style.fill = 'red'
          diagramInstance.dataBind();
        }}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

Header editing

Diagram provides the support to edit swimlane headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that. The following image illustrates how to edit the swimlane header.



Lanes

Lane is a functional unit or a responsible department of a business process that helps to map a process within the functional unit or in between other functional units.

The number of [lanes](#) can be added to swimlane. The lanes are automatically stacked inside swimlane based on the order they are added.

Create an empty lane

- The lanes `id` is used to define the name of the lane and its further used to find the lane at runtime and do any customization.
- We can provide additional information to the lane by using the [addInfo](#) property of the lane.

The following code example illustrates how to define a swimlane with lane.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    header: {

```

```

        annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
        height: 50, style: { fontSize: 11 },
    },
    // initialize the lane of swimlane
    lanes: [
        {
            id: 'stackCanvas1',
            // set the lane height
            height: 100,
            // set the lane info
            addInfo: { name: 'lane1' }
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
});
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: 'transparent' } },
            height: 50, style: { fontSize: 11 },
        },
        // initialize the lane of swimlane
        lanes: [
            {

```

```

        id: 'stackCanvas1',
        // set the lane height
        height: 100,
        // set the lane info
        addInfo: { name: 'lane1' }
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
});
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Create lane header

- The [header](#) property of lane allows you to textually describe the lane and to customize the appearance of the description.

The following code example illustrates how to define a lane header.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
        fill: 'transparent' } },
      height: 50, style: { fontSize: 11 },
    }
  }
}];

```



```

    },
    // Intialize lane to swimlane
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        // Intialize header to lane
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
  // Text(label) added to the node
});
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: 'transparent' } } },
      height: 50, style: { fontSize: 11 },
    },
    // Intialize lane to swimlane
    lanes: [

```

```

        {
            id: 'stackCanvas1',
            height: 100,
            // Intialize header to lane
            header: {
                annotation: { content: 'CUSTOMER' }, width: 50,
                style: { fontSize: 11 }
            },
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
// Text(label) added to the node
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Customizing lane header

- The size of lane can be controlled by using the [width](#) and [height](#) properties of the lane.
- The appearance of the lane can be set by using the [style](#) properties.

The following code example illustrates how to customize the lane header.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
    }
}];

```

```

        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                // customization of lane header
                header: {
                    annotation: { content: 'Online Consumer' }, width:
30,
                    style: { fontSize: 11 }, style: { fill: 'red' }
                },
            },
        ],
        phases: [
            {
                id: 'phase1', offset: 170,
                header: { annotation: { content: 'Phase' } }
            },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    });
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: 'transparent' } },
            height: 50, style: { fontSize: 11 },

```

```

    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        // customization of lane header
        header: {
          annotation: { content: 'Online Consumer' }, width:
30,
          style: { fontSize: 11 }, style: { fill: 'red' }
        },
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
});
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Dynamic customization of lane header

You can customize the lane header style and text properties dynamically. The following code illustrates how to dynamically customize the lane header.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {

```

```

        annotation: { content: 'SALES PROCESS FLOW CHART', },
        height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
    },
    },
    lanes: [
        {
            id: 'stackCanvas1',
            height: 100,
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
});
// initialize Diagram component
let diagramInstance;
function App() {
    return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram) } width={'100%'} height={'600px'} nodes={node}
    created={() => {
        // Dynamically change the lane header
        let lane = diagramInstance.nodes[0];
        lane.shape.lanes[0].header.style.fill = 'blue';
        diagramInstance.dataBind();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        // Intialize header to swimlane
        header: {
            annotation: { content: 'SALES PROCESS FLOW CHART', },

```

```

        height: 70, style: { fontSize: 11 }, style: { fill: 'pink'
    },
    },
    lanes: [
        {
            id: 'stackCanvas1',
            height: 100,
        },
    ],
    phases: [
        {
            id: 'phase1', offset: 170,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  ]];
// initialize Diagram component
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created = { () => {
        // Dynamically change the lane header
        let lane : nodeModel = diagramInstance.nodes[0];
        lane.shape.lanes[0].header.style.fill = 'blue';
        diagramInstance.dataBind();
      } }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Add lane at runtime

You can add the a lane at runtime by using the client side API method called `addLanes`. The following code illustrates how to dynamically add lane to swimlane.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {

```

```

    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
product',
fill: 'transparent' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
            style: { fontSize: 11 }
        },
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of
product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100,
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
                content: 'Does \n Consumer want \nthe
product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120,
          },
        ],
      },
    ],
    phases: [
      {
        id: 'phase1', offset: 120,
        header: { annotation: { content: 'Phase' } }, style: {
product',
fill: 'red' }
      }, {
        id: 'phase2', offset: 200,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,

```

```

        height: 200,
        width: 350
    }];
    // initialize Diagram component
    let diagramInstance;
    function App() {
        return (<DiagramComponent id="container" ref={(diagram) =>
        (diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
        created={() => {
            // Dynamically add the lane
            let lane = [{ id: "lane1", height: 100 }];
            diagramInstance.addLanes(diagramInstance.nodes[0], lane, 1);
        }}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);
    {% endraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                header: {
                    annotation: { content: 'CUSTOMER' }, width: 50,
                    style: { fontSize: 11 }
                },
                children: [
                    {
                        id: 'node1',
                        annotations: [
                            {
                                content: 'Consumer learns \n of product',
                                style: { fontSize: 11 }
                            }
                        ]
                    },
                ],
                margin: { left: 60, top: 30 },
            }
        ]
    }
}];

```



```

        height: 40, width: 100,
      }, {
        id: 'node2',
        shape: { type: 'Flow', shape: 'Decision' },
        annotations: [
          {
            content: 'Does \n Consumer want \nthe product',
            style: { fontSize: 11 }
          }
        ],
        margin: { left: 200, top: 20 },
        height: 60, width: 120,
      },
    ],
  },
  phases: [
    {
      id: 'phase1', offset: 120,
      header: { annotation: { content: 'Phase' } }
    },
    {
      id: 'phase2', offset: 200,
      header: { annotation: { content: 'Phase' } }
    }
  ],
  phaseSize: 20,
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
}];
// initialize Diagram component
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created = { () => {
        // Dynamically add the lane
        let lane = [{id:"lane1",height:100}];
        diagramInstance.addLanes(diagramInstance.nodes[0], lane, 1);
      } }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Add children to lane

To add nodes to lane, you should add [children](#) collection of the lane.

The following code example illustrates how to add nodes to lane.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        // Set the children of lane
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of
product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100,
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
                content: 'Does \n Consumer want \nthe
product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120,
          },
        ],
      },
    ],
  },
  phases: [
    {
```

```

        id: 'phase1', offset: 170,
        header: { annotation: { content: 'Phase' } }
      },
    ],
    phaseSize: 20,
  },
  offsetX: 300, offsetY: 200,
  height: 200,
  width: 350
});
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    // Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style:
{ fill: 'transparent' } } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        // Set the children of lane
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of
product',
                style: { fontSize: 11 }
              }
            ]
          }
        ]
      }
    ]
  }
}];

```

```

        ],
        margin: { left: 60, top: 30 },
        height: 40, width: 100,
      }, {
        id: 'node2',
        shape: { type: 'Flow', shape: 'Decision' },
        annotations: [
          {
            content: 'Does \n Consumer want \nthe
product',
            style: { fontSize: 11 }
          }
        ],
        margin: { left: 200, top: 20 },
        height: 60, width: 120,
      },
    ],
  },
  ],
  phases: [
    {
      id: 'phase1', offset: 170,
      header: { annotation: { content: 'Phase' } }
    },
  ],
  phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

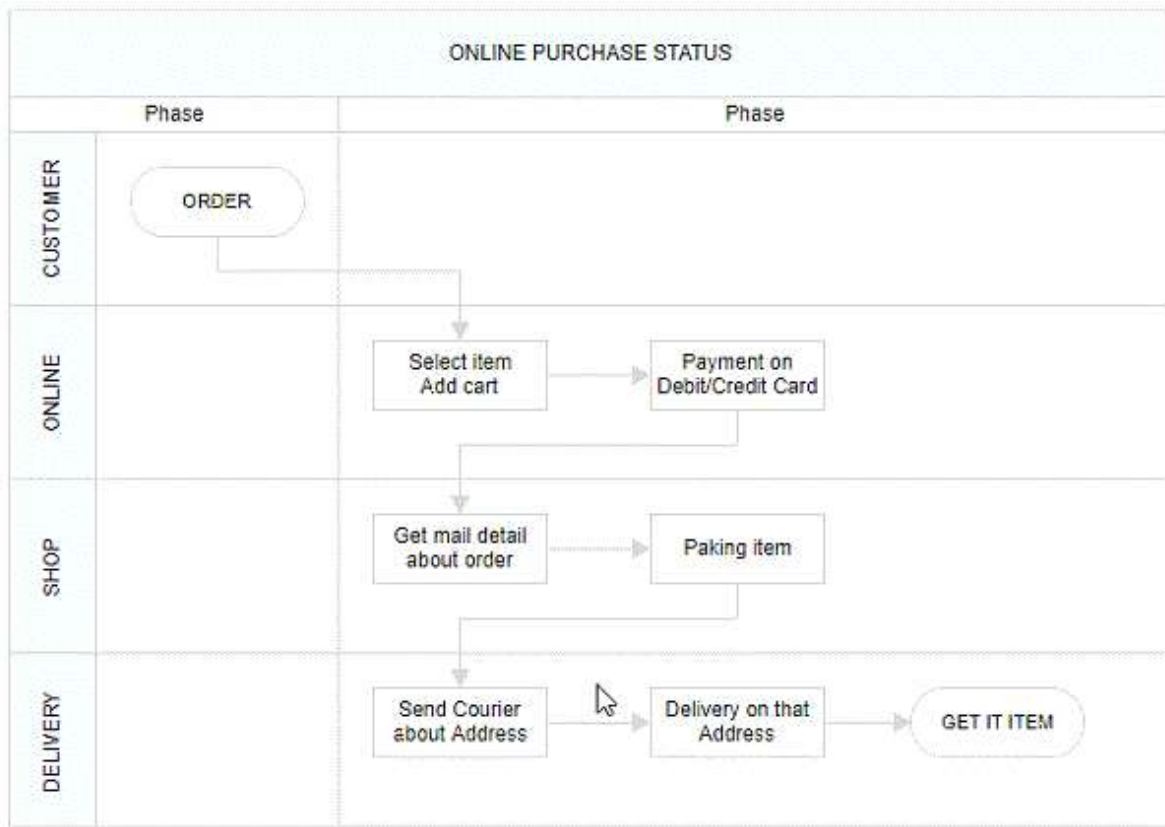
```

Lane interaction

Resizing lane

- Lane can be resized in the bottom and left direction.
- Lane can be resized by using resize selector of the lane.
- Once you can resize the lane, the swimlane will be resized automatically.
- The lane can be resized either resizing the selector or the tight bounds of the child object. If the child node move to edge of the lane it can be automatically resized.

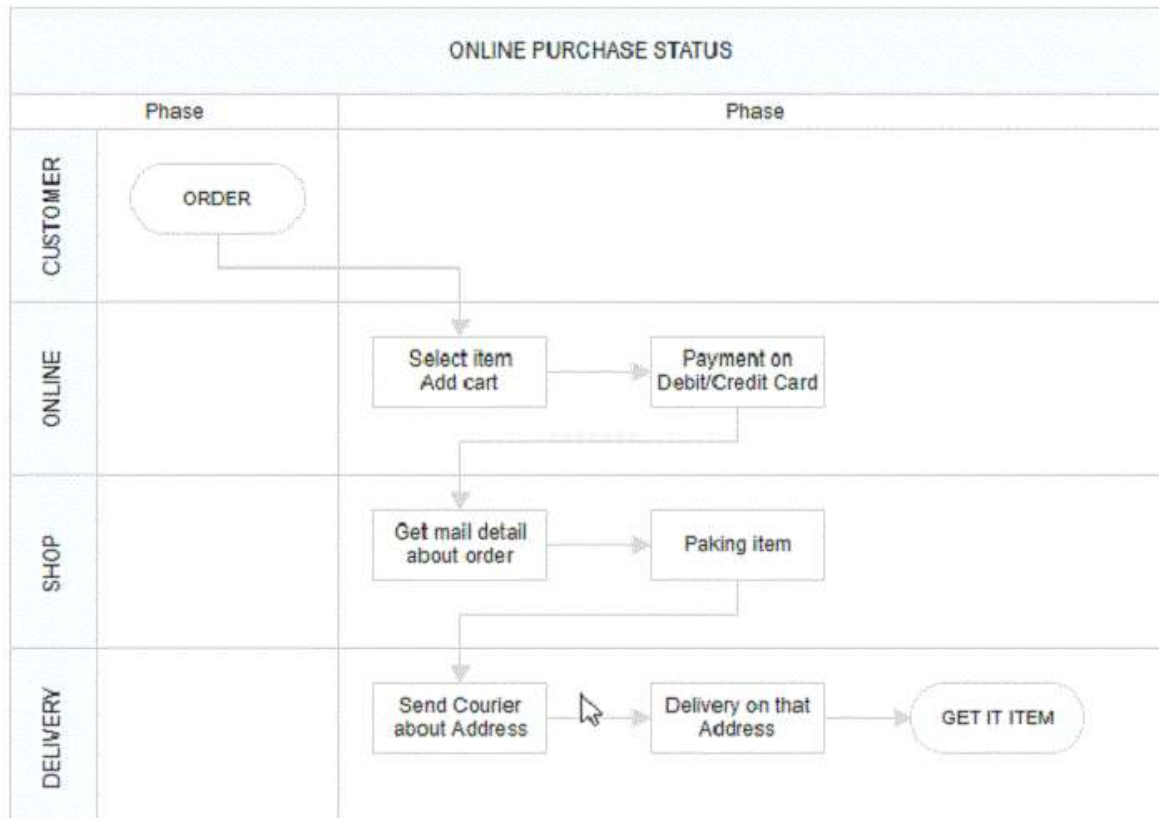
The following image illustrates how to resize the lane.



Lane swapping

- Lanes can be swapped using drag the lanes over another lane.
- Helper should intimate the insertion point while lane swapping.

The following image illustrates how swapping the lane.



Disable Swimlane Lane swapping

You can disable swimlane lane swapping by using the property called `canMove`.

The following code illustrates how to disable swimlane lane swapping.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } }},
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                header: {
                    annotation: { content: 'CUSTOMER' }, width: 50,

```

```

        style: { fontSize: 11 }
      },
      children: [
        {
          id: 'node1',
          annotations: [
            {
              content: 'Consumer learns \n of
product',
              style: { fontSize: 11 }
            }
          ],
          margin: { left: 60, top: 30 },
          height: 40, width: 100, canMove: false
        }, {
          id: 'node2',
          shape: { type: 'Flow', shape: 'Decision' },
          annotations: [
            {
              content: 'Does \n Consumer want \nthe
product',
              style: { fontSize: 11 }
            }
          ],
          margin: { left: 200, top: 20 },
          height: 60, width: 120, canMove: false
        }
      ],
    },
  ],
  phases: [
    {
      id: 'phase1', offset: 120,
      header: { annotation: { content: 'Phase' } }, style: {
fill: 'red' }
    }, {
      id: 'phase2', offset: 200,
      header: { annotation: { content: 'Phase' } }
    },
  ],
  phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
});
// initialize Diagram component
let diagramInstance;
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
created={() => {
    let lane = [{ id: "lane1", height: 100, canMove: false }];
    diagramInstance.addLanes(diagramInstance.nodes[0], lane, 1);
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```

```
root.render(<App />);
{% enddraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } }},
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100, canMove: false
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
                content: 'Does \n Consumer want \nthe product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120, canMove: false
          },
        ],
      },
    ],
  },
}
```



```

        },
        ],
        phases: [
            {
                id: 'phase1', offset: 120,
                header: { annotation: { content: 'Phase' } }
            },
            {
                id: 'phase2', offset: 200,
                header: { annotation: { content: 'Phase' } }
            }
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  }];
// initialize Diagram component
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created = { () => {
        let lane = [{id:"lane1",height:100,canMove: false}];
        diagramInstance.addLanes (diagramInstance.nodes[0],lane,1);
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

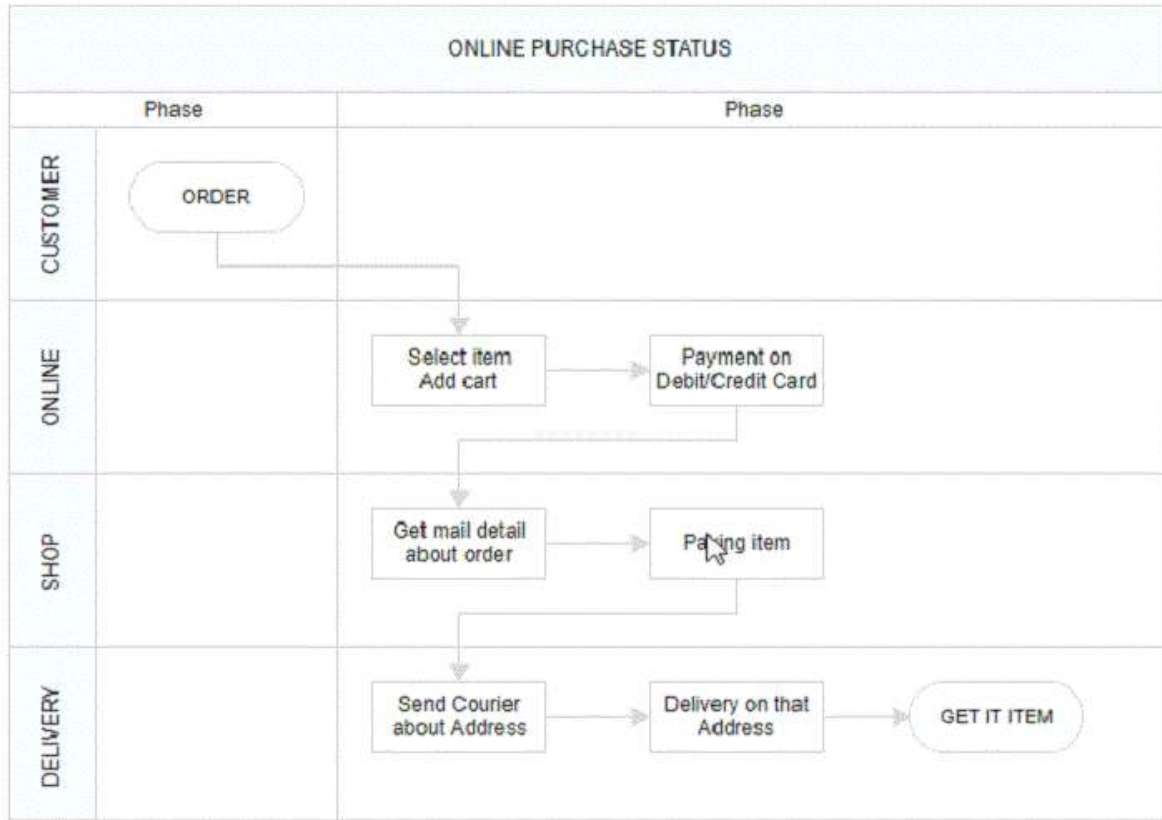
Resize helper

- The special resize helper will be used to resize the lanes.
- The resize cursor will be available on the left and bottom direction alone.
- Once resize the lane the swimlane will be resized automatically.

Children interaction in lanes

- You can resize the child node within swimlanes.
- You can drag the child nodes within lane.
- Interchange the child nodes from one lane to another lane.
- Drag and drop the child nodes from lane to diagram.
- Drag and drop the child nodes from diagram to lane.
- Based on the child node interactions,the lane size should be updated.

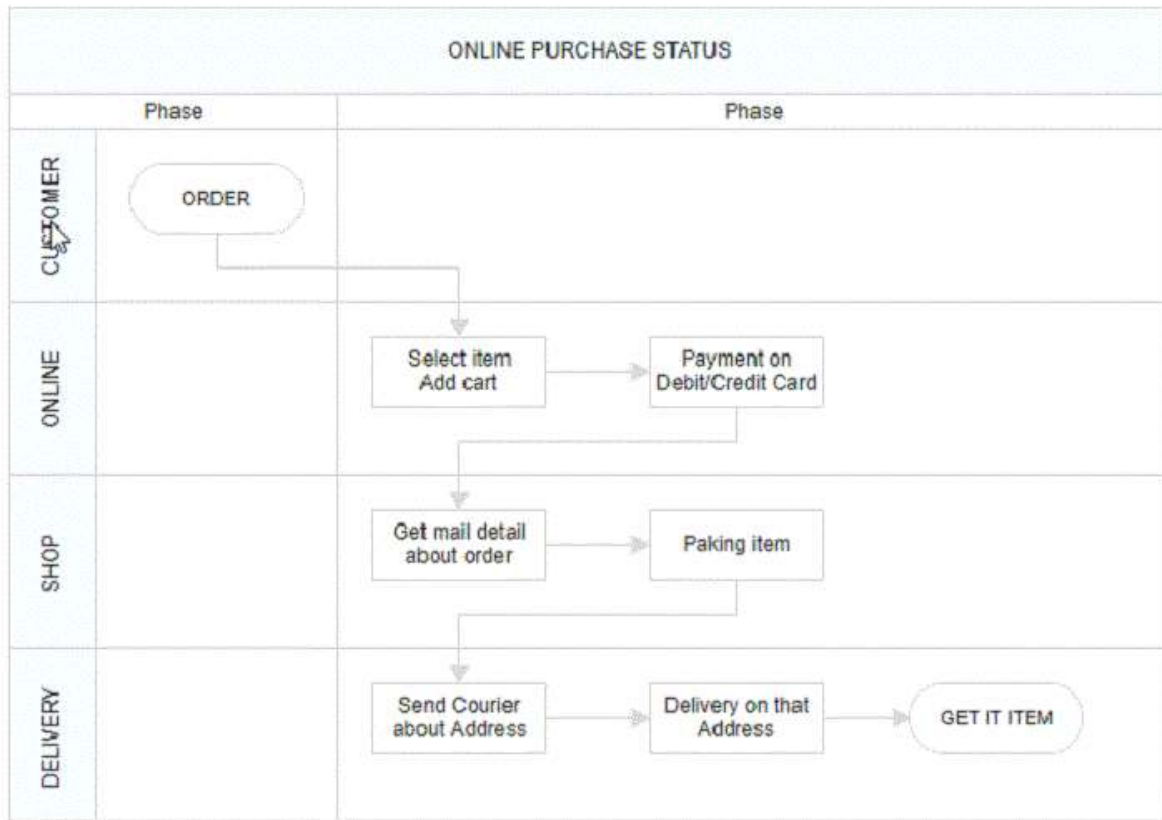
The following image illustrates children interaction in lane.



[Lane header editing](#)

Diagram provides the support to edit Lane headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that.

The following image illustrates how to edit the lane header.



Phase

Phase are the subprocess which will split each lane as horizontally or vertically based on the swimlane orientation. The multiple number of [Phase](#) can be added to swimlane.

The following code example illustrates how to add phase at swimlane.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
    shape: {
        type: 'SwimLane',
        orientation: 'Horizontal',
        //Intialize header to swimlane
        header: {
            annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } },
            height: 50, style: { fontSize: 11 },
        },
        lanes: [
            {
                id: 'stackCanvas1',
                height: 100,
                header: {
                    annotation: { content: 'CUSTOMER' }, width: 50,

```

```

        style: { fontSize: 11 }
      },
      children: [
        {
          id: 'node1',
          annotations: [
            {
              content: 'Consumer learns \n of
product',
              style: { fontSize: 11 }
            }
          ],
          margin: { left: 60, top: 30 },
          height: 40, width: 100,
        }, {
          id: 'node2',
          shape: { type: 'Flow', shape: 'Decision' },
          annotations: [
            {
              content: 'Does \n Consumer want \nthe
product',
              style: { fontSize: 11 }
            }
          ],
          margin: { left: 200, top: 20 },
          height: 60, width: 120,
        },
      ],
    },
  ],
  // Set phase to swimlane
  phases: [
    {
      id: 'phase1', offset: 120,
      header: { annotation: { content: 'Phase' } }
    }, {
      id: 'phase2', offset: 200,
      header: { annotation: { content: 'Phase' } }
    },
  ],
  phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
});
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } }},
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100,
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
                content: 'Does \n Consumer want \nthe product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120,
          },
        ],
      },
    ],
  },
  // Set phase to swimlane
  phases: [
    {
      id: 'phase1', offset: 120,
      header: { annotation: { content: 'Phase' } }
    }
  ]
}];

```

```

        }, {
            id: 'phase2', offset: 200,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
},
offsetX: 300, offsetY: 200,
height: 200,
width: 350
});
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Dynamically add phase to lane

You can add the a phase at runtime by using client side API method called `addPhases`. The following code example illustrates how to add phase at run time.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [
    {
        shape: {
            type: 'SwimLane',
            orientation: 'Horizontal',
            //Initialize header to swimlane
            header: {
                annotation: {
                    content: 'ONLINE PURCHASE STATUS',
                    style: { fill: 'transparent' },
                },
                height: 50,
                style: { fontSize: 11 },
            },
            lanes: [
                {
                    id: 'stackCanvas1',
                    height: 100,
                    header: {
                        annotation: { content: 'CUSTOMER' },

```

```

        width: 50,
        style: { fontSize: 11 },
      },
      children: [
        {
          id: 'node1',
          annotations: [
            {
              content: 'Consumer learns \n of
product',
              style: { fontSize: 11 },
            },
          ],
          margin: { left: 60, top: 30 },
          height: 40,
          width: 100,
        },
        {
          id: 'node2',
          shape: { type: 'Flow', shape: 'Decision' },
          annotations: [
            {
              content: 'Does \n Consumer want \nthe
product',
              style: { fontSize: 11 },
            },
          ],
          margin: { left: 200, top: 20 },
          height: 60,
          width: 120,
        },
      ],
    },
  ],
  phases: [
    {
      id: 'phase1',
      offset: 120,
      header: { annotation: { content: 'Phase' } },
      style: { fill: 'red' },
    },
    {
      id: 'phase2',
      offset: 200,
      header: { annotation: { content: 'Phase' } },
    },
  ],
  phaseSize: 20,
},
offsetX: 300,
offsetY: 200,
height: 200,
width: 350,
},
];
let diagramInstance;
// initialize Diagram component

```

```
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
created={() => {
    let phase = [
      {
        id: 'phase3',
        offset: 220,
        header: { annotation: { content: 'Phase' } } },
    ];
    diagramInstance.addPhases(diagramInstance.nodes[0], phase);
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [
  {
    shape: {
      type: 'SwimLane',
      orientation: 'Horizontal',
      //Intialize header to swimlane
      header: {
        annotation: {
          content: 'ONLINE PURCHASE STATUS',
          style: { fill: 'transparent' },
        },
        height: 50,
        style: { fontSize: 11 },
      },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' },
          width: 50,
          style: { fontSize: 11 },
        },
        children: [
          {
            id: 'node1',
            annotations: [
```



```

        {
            content: 'Consumer learns \n of product',
            style: { fontSize: 11 },
        },
    ],
    margin: { left: 60, top: 30 },
    height: 40,
    width: 100,
},
{
    id: 'node2',
    shape: { type: 'Flow', shape: 'Decision' },
    annotations: [
        {
            content: 'Does \n Consumer want \nthe product',
            style: { fontSize: 11 },
        },
    ],
    margin: { left: 200, top: 20 },
    height: 60,
    width: 120,
},
],
},
],
phases: [
    {
        id: 'phase1',
        offset: 120,
        header: { annotation: { content: 'Phase' } },
        style: { fill: 'red' },
    },
    {
        id: 'phase2',
        offset: 200,
        header: { annotation: { content: 'Phase' } },
    },
],
phaseSize: 20,
},
offsetX: 300,
offsetY: 200,
height: 200,
width: 350,
},
];
let diagramInstance: DiagramComponent;
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'100%'}
            height={'600px'}
            nodes={node}
            created={() => {

```

```

    let phase = [
      {
        id: 'phase3',
        offset: 220,
        header: { annotation: { content: 'Phase' } } },
    ],
  ];
  diagramInstance.addPhases(diagramInstance.nodes[0], phase);
}
}
/>
);
}
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Customizing phase

- The length of region can be set by using the [offset](#) property of the phase.
- Every phase region can be textually described with the [header](#) property of the phase
- You can increase the width of phase by using [phaseSize](#) property of swimlane.
- We can provide additional information to the phase by using the [addInfo](#) property of the phase.

The following code example illustrates how to customize the phase in swimlane.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } } },
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [
          {
            id: 'node1',
            annotations: [

```

```

        content: 'Consumer learns \n of
product',
        style: { fontSize: 11 }
    },
    ],
    margin: { left: 60, top: 30 },
    height: 40, width: 100,
}, {
    id: 'node2',
    shape: { type: 'Flow', shape: 'Decision' },
    annotations: [
        {
            content: 'Does \n Consumer want \nthe
product',
            style: { fontSize: 11 }
        }
    ],
    margin: { left: 200, top: 20 },
    height: 60, width: 120,
},
    ],
    },
    ],
    phases: [
        {
            id: 'phase1', offset: 120,
            // set the phase info
            addInfo: { name: 'phase1' },
            header: { annotation: { content: 'Phase' } }, style: {
fill: 'red' }
        }, {
            id: 'phase2', offset: 200,
            header: { annotation: { content: 'Phase' } }
        },
    ],
    phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,

```

```

NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  shape: {
    type: 'SwimLane',
    orientation: 'Horizontal',
    //Intialize header to swimlane
    header: {
      annotation: { content: 'ONLINE PURCHASE STATUS', style: {
fill: 'transparent' } }},
      height: 50, style: { fontSize: 11 },
    },
    lanes: [
      {
        id: 'stackCanvas1',
        height: 100,
        header: {
          annotation: { content: 'CUSTOMER' }, width: 50,
          style: { fontSize: 11 }
        },
        children: [
          {
            id: 'node1',
            annotations: [
              {
                content: 'Consumer learns \n of product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 60, top: 30 },
            height: 40, width: 100,
          }, {
            id: 'node2',
            shape: { type: 'Flow', shape: 'Decision' },
            annotations: [
              {
                content: 'Does \n Consumer want \nthe product',
                style: { fontSize: 11 }
              }
            ],
            margin: { left: 200, top: 20 },
            height: 60, width: 120,
          },
        ],
      },
    ],
  },
  phases: [
    {
      id: 'phase1', offset: 120,
      // set the phase info
      addInfo:{name:'phase1'},
      header: { annotation: { content: 'Phase' }
},style:{fill:'red'}
    },{
      id: 'phase2', offset: 200,
      header: { annotation: { content: 'Phase' } }
    }
  ]
}];

```

```

        },
        ],
        phaseSize: 20,
    },
    offsetX: 300, offsetY: 200,
    height: 200,
    width: 350
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Phase interaction

Resizing

- The phase can be resized by using its selector.
- You must select the phase header to enable the phase selection.
- Once the phase can be resized, the lane size will be updated automatically.

Resizing helper

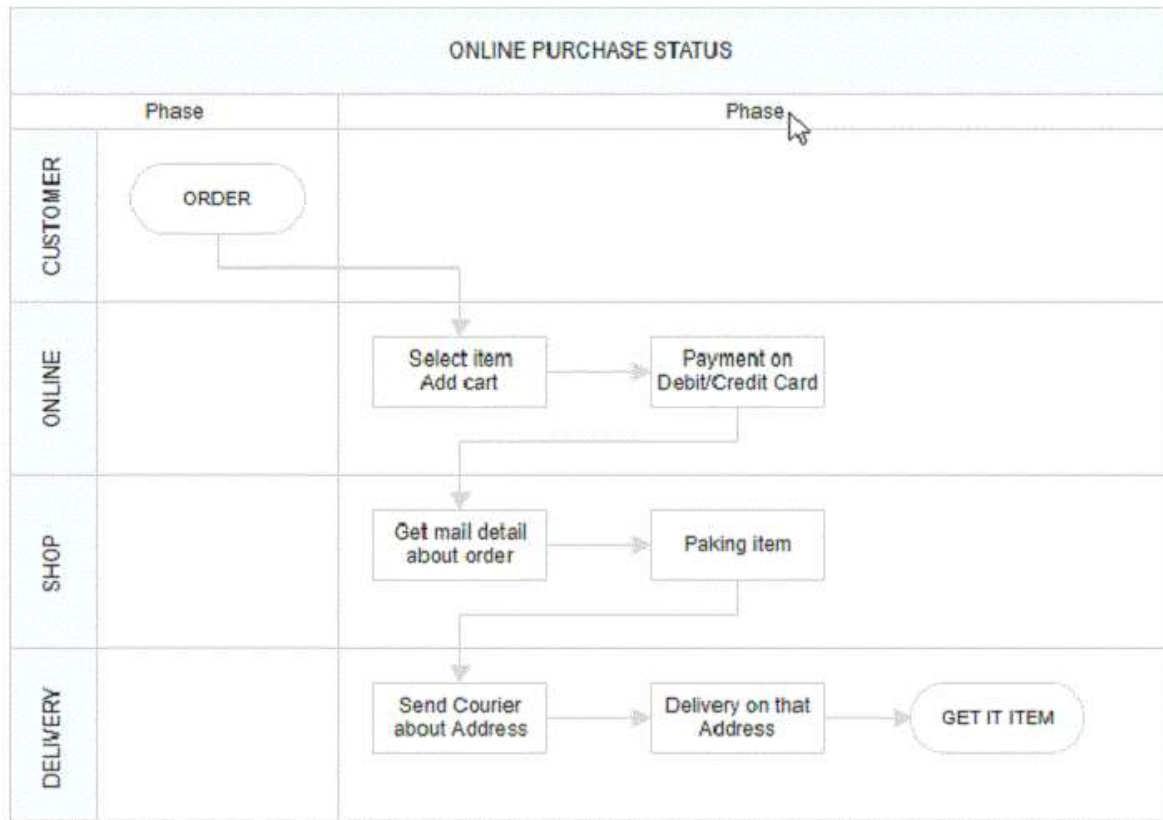
- The special resize selector will be used to resize the phase.
- The resize cursor will be available on the left and bottom direction for horizontal, and the top and bottom direction for vertical swimlane.

Phase header editing

Diagram provides the support to edit phase headers at runtime. We achieve the header editing by double click event. Double clicking the header label will enables the editing of that.

The following image illustrates how to edit the swimlane header.

The following image illustrates how to edit the phase header.



Add swimlane to palette

Diagram provides support to add swimlane and phases to symbol palette. The following code sample illustrate how to add swimlane and phases to palette.

The following code example illustrates how to customize the phase in swimlane.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent } from "@syncfusion/ej2-react-diagrams";
//Initialize the flowshapes for the symbol palette
export function getswimlaneShapes() {
    let swimlaneShapes = [
        {
            id: 'stackCanvas1',
            shape: {
                type: 'SwimLane', lanes: [
                    {
                        id: 'lane1',
                        style: { strokeColor: 'black' }, height: 60, width:
150,
                        header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
                    }
                ],
                orientation: 'Horizontal', isLane: true
            }
        }
    ]
}

```

```

    },
    height: 60,
    width: 140,
    offsetX: 70,
    offsetY: 30,
  }, {
    id: 'stackCanvas2',
    shape: {
      type: 'SwimLane',
      lanes: [
        {
          id: 'lane1',
          style: { strokeColor: 'black' }, height: 150, width:
60,
          header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
        }
      ],
      orientation: 'Vertical', isLane: true
    },
    height: 140,
    width: 60,
    // style: { fill: '#f5f5f5' },
    offsetX: 70,
    offsetY: 30,
  }, {
    id: 'verticalPhase',
    shape: {
      type: 'SwimLane',
      phases: [{ style: { strokeWidth: 1, strokeDashArray: '3,3',
strokeColor: '#A9A9A9' }, }],
      annotations: [{ text: '' }],
      orientation: 'Vertical', isPhase: true
    },
    height: 60,
    width: 140
  }, {
    id: 'horizontalPhase',
    shape: {
      type: 'SwimLane',
      phases: [{ style: { strokeWidth: 1, strokeDashArray: '3,3',
strokeColor: '#A9A9A9' }, }],
      annotations: [{ text: '' }],
      orientation: 'Horizontal', isPhase: true
    },
    height: 60,
    width: 140
  }
];
return swimlaneShapes;
}
function setPaletteNodeDefaults(node) {
  node.width = 70;
  node.height = 70;
  node.style.strokeColor = '#3A3A3A';
}
// Initialize the Symbol palette

```

```
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
    palettes=[
      {
        id: 'swimlane',
        expanded: true,
        symbols: getswimlaneShapes(),
        title: 'Swimlane Shapes',
      },
    ],
    symbolPreview={{
      height: 70,
      width: 70,
      offset: {
        x: 0.5,
        y: 0.5,
      },
    },
    symbolMargin={{
      left: 12,
      right: 12,
      top: 12,
      bottom: 12,
    }}
  )
  //Returns the default properties of node
  getNodeDefaults={setPaletteNodeDefaults} getSymbolInfo={(symbol) => {
    return {
      fit: true
    };
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo,
  SymbolPaletteComponent
} from "@syncfusion/ej2-react-diagrams";
//Initialize the flowshapes for the symbol palette
export function getswimlaneShapes(): NodeModel[] {
  let swimlaneShapes : NodeModel[]= [
    {
      id: 'stackCanvas1',
      shape: {
        type: 'SwimLane', lanes: [
          {
            id: 'lane1',
            style: { strokeColor: 'black' }, height: 60,
width: 150,
```



```

        header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
        }
    ],
    orientation: 'Horizontal', isLane: true
},
height: 60,
width: 140,
offsetX: 70,
offsetY: 30,
}, {
    id: 'stackCanvas2',
    shape: {
        type: 'SwimLane',
        lanes: [
            {
                id: 'lane1',
                style: { strokeColor: 'black' }, height: 150,
width: 60,
                header: { width: 50, height: 50, style: {
strokeColor: 'black', fontSize: 11 } },
                }
            ],
            orientation: 'Vertical', isLane: true
        },
        height: 140,
        width: 60,
        // style: { fill: '#f5f5f5' },
        offsetX: 70,
        offsetY: 30,
    }, {
        id: 'verticalPhase',
        shape: {
            type: 'SwimLane',
            phases: [{ style: { strokeWidth: 1, strokeDashArray:
'3,3', strokeColor: '#A9A9A9' }, }],
            annotations: [{ text: '' }],
            orientation: 'Vertical', isPhase: true
        },
        height: 60,
        width: 140
    }, {
        id: 'horizontalPhase',
        shape: {
            type: 'SwimLane',
            phases: [{ style: { strokeWidth: 1, strokeDashArray:
'3,3', strokeColor: '#A9A9A9' }, }],
            annotations: [{ text: '' }],
            orientation: 'Horizontal', isPhase: true
        },
        height: 60,
        width: 140
    }
    ]
};
return swimlaneShapes;
}
function setPaletteNodeDefaults(node: NodeModel): void {

```

```

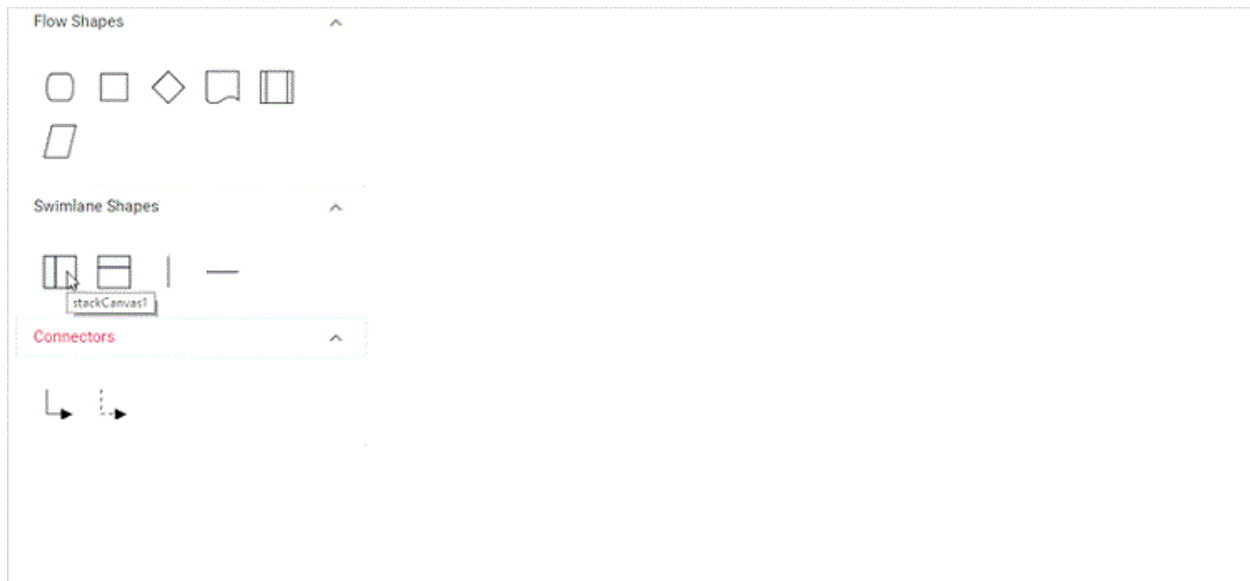
    node.width = 70;
    node.height = 70;
    node.style.strokeColor = '#3A3A3A';
  }
  // Initialize the Symbol palette
  function App() {
    return (
      <SymbolPaletteComponent
        id="container"
        expandMode={'Multiple'}
        palettes={[
          {
            id: 'swimlane',
            expanded: true,
            symbols: getswimlaneShapes(),
            title: 'Swimlane Shapes',
          },
        ]}
        symbolPreview={{
          height: 70,
          width: 70,
          offset: {
            x: 0.5,
            y: 0.5,
          },
        }}
        symbolMargin={{
          left: 12,
          right: 12,
          top: 12,
          bottom: 12,
        }}
      //Returns the default properties of node
      getNodeDefaults={setPaletteNodeDefaults}
      getSymbolInfo = {
        (symbol: NodeModel): SymbolInfo => {
          return {
            fit: true
          };
        }
      }
    ) />
  };
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Drag and drop swimlane to palette

- The drag and drop support for swimlane shapes has been provided.
- When you drag and drop the lane shape, if the diagram already contains swimlane with the same orientation, the lane will be added and stacked inside a swimlane based on the order. Otherwise, it will be added a new swimlane.
- The phase will only drop on swimlane shape with same orientation.

The following image illustrates how to drag symbol from palette.



Limitations

- Connectors cannot be canceled when added directly to swimlane. You must initialize the connector through connector collection.
- We cannot edit the phase line style.

Labels in React Diagram component

[Annotation](#) is a block of text that can be displayed over a node or connector. Annotation is used to textually represent an object with a string that can be edited at runtime. Multiple annotations can be added to a node/connector.

<!-- markdownlint-disable MD033 -->

Create annotation

An annotation can be added to a node/connector by defining the annotation object and adding that to the annotation collection of the node/connector. The [content](#) property of annotation defines the text to be displayed. The following code illustrates how to create a annotation.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
```

```

        strokeColor: 'white'
      },
      // Sets the Annotation for the Node
      annotations: [{
        // Sets the text to be displayed
        content: 'Annotation'
      }]
    }
  ]];
let connector = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 300
  },
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7'
  },
  // Sets the Annotation for the Connector
  annotations: [{
    // Sets the text to be displayed
    content: 'Annotation'
  }]
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={node} connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel, ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  }
}];

```

```

    },
    // Sets the Annotation for the Node
    annotations: [{
      // Sets the text to be displayed
      content: 'Annotation'
    }]
  }];
let connector: ConnectorModel[] = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 300
  },
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7'
  },
  // Sets the Annotation for the Connector
  annotations: [{
    // Sets the text to be displayed
    content: 'Annotation'
  }]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
      connectors = {connector}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Add annotations at runtime

- Annotations can be added at runtime by using the client-side method [addLabels](#). The following code illustrates how to add a annotation to a node.
- The annotation's [ID](#) property is used to define the name of the annotation and its further used to find the annotation at runtime and do any customization.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.

```

```

let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
}];
let diagramInstance;
let annotation = [{
  id: 'labell1',
  content: 'Annotation'
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
    created={() => {
      //Method to add labels at run time
      diagramInstance.addLabels(diagramInstance.nodes[0], annotation);
      diagramInstance.dataBind();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  ShapeAnnotationModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
}];
let diagramInstance: DiagramComponent;
let annotation: ShapeAnnotationModel[] = [{

```

```

        id: 'label1',
        content: 'Annotation'
    }];
    // initialize Diagram component
    function App() {
        return (
            <DiagramComponent
                id="container"
                ref={(diagram) => (diagramInstance = diagram)}
                width={'100%'}
                height={'600px'}
                nodes={node}
                created = {
                    () => {
                        //Method to add labels at run time
                        diagramInstance.addLabels(diagramInstance.nodes[0],
annotation);
                        diagramInstance.dataBind();
                    }
                }
            />
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

Remove annotation

A collection of annotations can be removed from the node by using client-side method [removeLabels](#). The following code illustrates how to remove a annotation to a node.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation'
    }]
}];
// initialize Diagram component
function App() {

```

```

    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
created={() => {
    //Method to remove labels at run time
    diagramInstance.removeLabels(diagramInstance.nodes[0],
diagramInstance.nodes[0].annotations[0]);
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    ShapeAnnotationModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation'
    }]
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'100%'}
            height={'600px'}
            nodes={node}
            created = {
                () => {
                    //Method to remove labels at run time
                    diagramInstance.removeLabels(diagramInstance.nodes[0],
diagramInstance.nodes[0].annotations[0]);
                }
            }
        >
    );
}

```



```

    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Update annotation at runtime

You can change any annotation properties at runtime and update it through the client-side method [dataBind](#).

The following code example illustrates how to change the annotation properties.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation'
  }]
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={node}
    created={() => {
      diagramInstance.nodes[0].annotations[0].content = 'Updated
Annotation';
      //Method to update the annotation at run time
      diagramInstance.dataBind();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {

```

```

    Diagram,
    DiagramComponent,
    NodeModel,
    ShapeAnnotationModel
  } from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation'
  }]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created = {
        () => {
          diagramInstance.nodes[0].annotations[0].content = 'Updated
Annotation';
          //Method to update the annotation at run time
          diagramInstance.dataBind();
        }
      }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Alignment

Annotation can be aligned relative to the node boundaries. It has [margin](#), [offset](#), horizontal, and vertical alignment settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset

The offset property of annotation is used to align the annotations based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Set the size for a nodes annotation by using [width](#) and [height](#) properties.

The following code shows the relationship between the annotation position (black color circle) and offset (fraction values).

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    // Sets the content for the annotation
    content: 'Annotation',
    //Sets the offset for the content
    offset: {
      x: 0,
      y: 1
    }
  }]
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ShapeAnnotationModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
```

```

    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
      // Sets the content for the annotation
      content: 'Annotation',
      //Sets the offset for the content
      offset: {
        x: 0,
        y: 1
      }
    }]
  }];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Horizontal and vertical alignment

The [horizontalAlignment](#) property of annotation is used to set how the annotation is horizontally aligned at the annotation position determined from the fraction values. The [verticalAlignment](#) property is used to set how annotation is vertically aligned at the annotation position.

The following tables illustrates all the possible alignments visually with 'offset (0, 0)'.

Horizontal Alignment	Vertical Alignment	Output with Offset(0,0)
-----	-----	-----

| Left | Top |



|

| Center | Top |

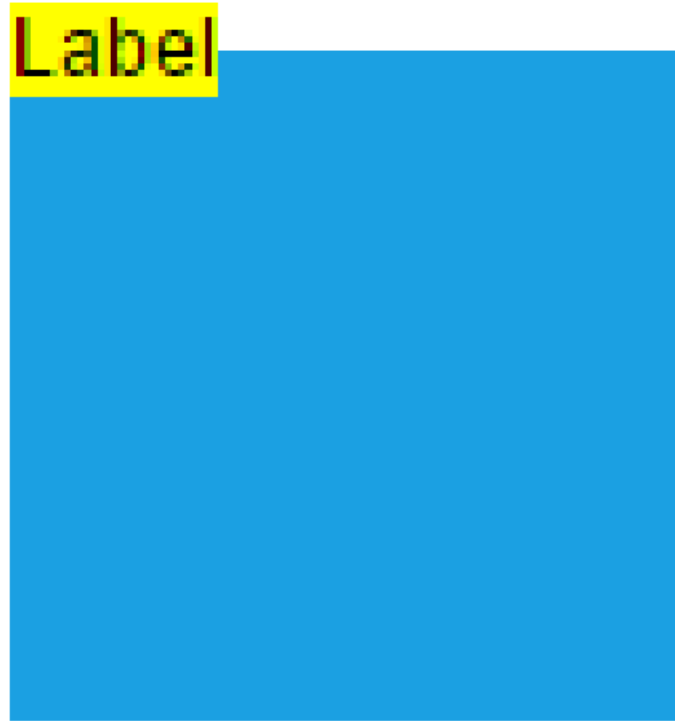


|

| Right | Top |



|



| Left | Center |

|

| Center | Center|

Label



|

| Right | Center |

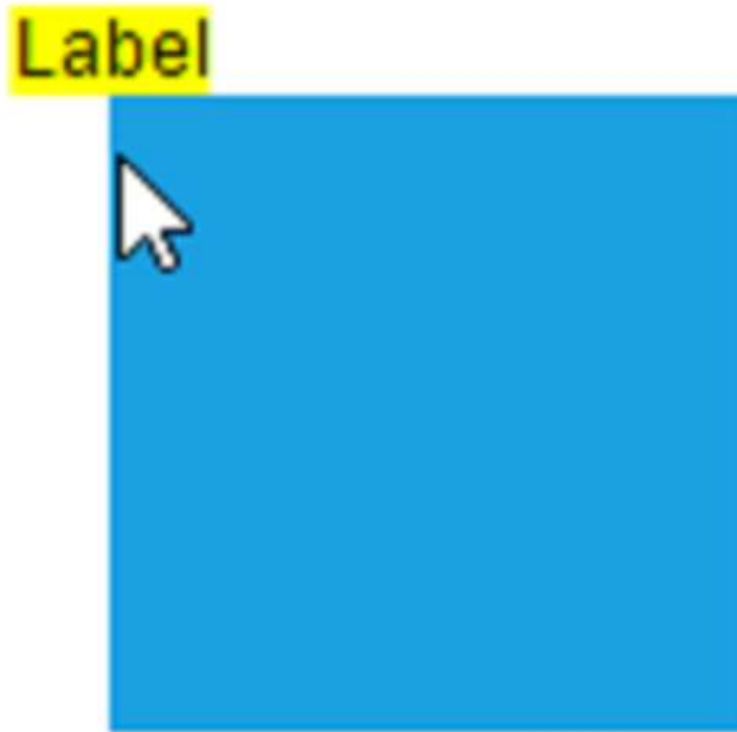
Label



|

Label



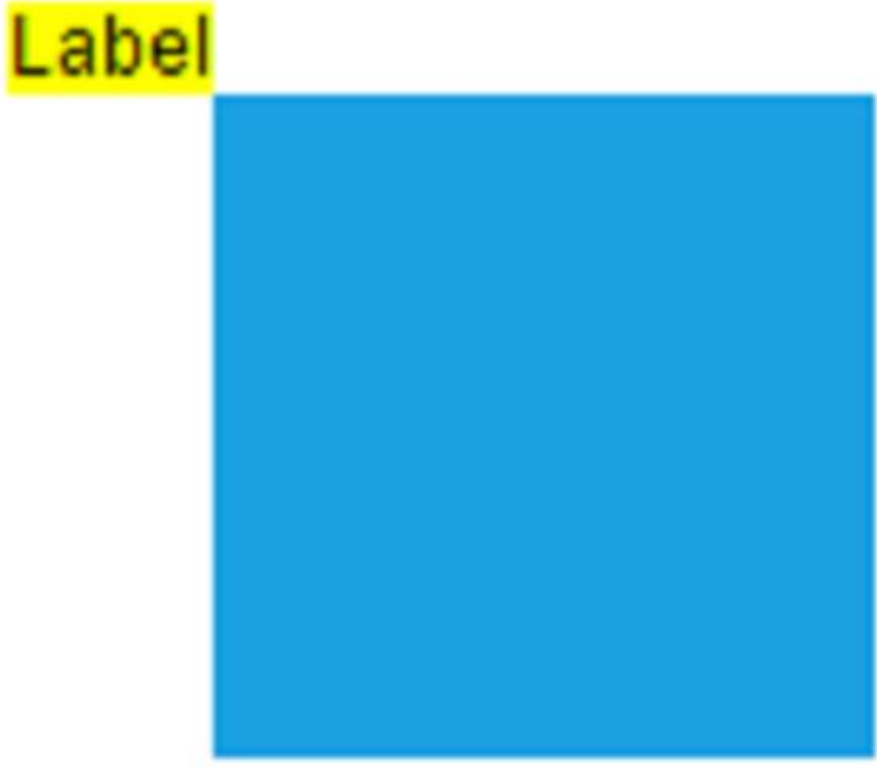


| Center | Bottom |

|

| Right | Bottom |

Label



The following codes illustrates how to align annotations.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation',
    // Sets the horizontal alignment as left
    horizontalAlignment: 'Left',
    // Sets the vertical alignment as Center
    verticalAlignment: 'Center'
  }]
}]
```

```

    }
  }];
  // initialize Diagram component
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={node}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ShapeAnnotationModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation',
    // Sets the horizontal alignment as left
    horizontalAlignment: 'Left',
    // Sets the vertical alignment as Center
    verticalAlignment: 'Center'
  }]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Annotation alignment with respect to segments

The offset and alignment properties of annotation allows you to align the connector annotations with respect to the segments.

The following code example illustrates how to align connector annotations.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Task1'
  }]
}, {
  id: 'node2',
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Task2'
  }]
}];
let connector = [{
  sourceID: 'node1',
  targetID: 'node2',
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7',
    strokeWidth: 2
  },
  // Sets the annotation for the connector
  annotations: [{
    content: '0',
    // Sets the offset for the content
  }]
```

```

        offset: 0
      }, {
        content: '1',
        // Sets the offset for the content
        offset: 1
      }]
    });
  // initialize Diagram component
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    nodes={node} connectors={connector}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel, ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Task1'
  }]
}, {
  id: 'node2',
  // Position of the node
  offsetX: 300,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Task2'
  }]
}

```



```

    ]]
  }];
  let connector: ConnectorModel[] = [{
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Orthogonal',
    style: {
      strokeColor: '#6BA5D7',
      strokeWidth: 2
    },
    // Sets the annotation for the connector
    annotations: [{
      content: '0',
      // Sets the offset for the content
      offset: 0
    }, {
      content: '1',
      // Sets the offset for the content
      offset: 1
    }
  ]
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        nodes={node}
        connectors={connector}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Margin

[Margin](#) is an absolute value used to add some blank space in any one of its four sides. The annotations can be displaced with the margin property.

The following code example illustrates how to align a annotation based on its `offset`, `horizontalAlignment`, `verticalAlignment`, and `margin` values.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,

```

```

    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Sets the annotation for the connector
    annotations: [{
      content: 'Task1',
      // Sets the margin for the content
      margin: {
        top: 10
      },
      horizontalAlignment: 'Center',
      verticalAlignment: 'Top',
      offset: {
        x: 0.5,
        y: 1
      }
    }]
  }];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the connector
  annotations: [{
    content: 'Task1',
    // Sets the margin for the content
    margin: {
      top: 10
    }
  }]
}];

```

```

    },
    horizontalAlignment: 'Center',
    verticalAlignment: 'Top',
    offset: {
      x: 0.5,
      y: 1
    }
  }
  ]]
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Text align

The [textAlign](#) property of annotation allows you to set how the text should be aligned (left, right, center, or justify) inside the text block. The following codes illustrate how to set `textAlign` for an annotation.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the NOde
  annotations: [{
    content: 'Text align is set as Left',
    // Sets the textAlign as left for the content
    style: {
      textAlign: 'Left'
    }
  }]
}];
// initialize Diagram component
function App() {

```

```

    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the NOde
    annotations: [{
        content: 'Text align is set as Left',
        // Sets the textAlign as left for the content
        style: {
            textAlign: 'Left'
        }
    }]
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Hyperlink

Diagram provides a support to add a [hyperlink](#) for the nodes/connectors annotation. It can also be customized.

A User can open the hyperlink in the new window, the same tab and the new tab by using the [hyperlinkOpenState](#) property

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the Node
  annotations: [{
    hyperlink: {
      link: 'https://hr.syncfusion.com/home',
      //Set the link to open in the current tab
      hyperlinkOpenState: 'CurrentTab'
    }
  }]
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
```

```

    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Sets the annotation for the Node
    annotations: [{
      hyperlink: {
        link: 'https://hr.syncfusion.com/home',
        //Set the link to open in the current tab
        hyperlinkOpenState: 'CurrentTab'
      }
    }]
  }];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Template Support for Annotation

Diagram provides template support for annotation. you should define a SVG/HTML content as string in the annotation's [template](#) property.

The following code illustrates how to define a template in node's annotation. similarly, you can define it in connectors.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the Node
  annotations: [{
    // Set an template for annotation
    template: '<div><input type="button" value="Submit"></div>'
  }]
}];

```

```

    ]]
  }];
let connector = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 300
  },
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7'
  },
  // Sets the Annotation for the Connector
  annotations: [{
    // Set an template for annotation
    height: 60, width: 100, offset: 0.5,
    template: '<div><input type="button" value="Submit"></div>'
  }]
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node} connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the Node
  annotations: [{
    // Set an template for annotation

```

```

        template: '<div><input type="button" value="Submit"></div>'
    }],
  }];
let connector: ConnectorModel[] = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 300
  },
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7'
  },
  // Sets the Annotation for the Connector
  annotations: [{
    // Set an template for annotation
    height: 60, width: 100, offset: 0.5,
    template: '<div><input type="button" value="Submit"></div>'
  }]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
      connectors={connector}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Wrapping

When text overflows node boundaries, you can control it by using [text wrapping](#). So, it is wrapped into multiple lines. The wrapping property of annotation defines how the text should be wrapped. The following code illustrates how to wrap a text in a node.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,

```



```

        height: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        },
        //Sets the annotation for the node
        annotations: [{
            content: 'Annotation Text Wrapping',
            // Sets the style for the text wrapping
            style: {
                textWrapping: 'Wrap'
            }
        }]
    }];
    // initialize Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    //Sets the annotation for the node
    annotations: [{
        content: 'Annotation Text Wrapping',
        // Sets the style for the text wrapping
        style: {
            textWrapping: 'Wrap'
        }
    }]
}];
// initialize Diagram component
function App() {
    return (

```

```
<DiagramComponent
  id="container"
  width={'100%'}
  height={'600px'}
  nodes={node}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

| Value | Description | Image |

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

| No Wrap | Text will not be wrapped. |



Label Text Wrapping

|

| Wrap | Text-wrapping occurs, when the text overflows beyond the available node width. |



| WrapWithOverflow (Default) | Text-wrapping occurs, when the text overflows beyond the available node width. However, the text may overflow beyond the node width in the case of a very long word. |



|

Text overflow

The label's [TextOverflow](#) property is used control whether to display the overflowed content in node or not.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
```

```

        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation Text',
        // Sets the style for the text to be displayed
        style: {
            textOverflow: 'Ellipsis'
        }
    }]
    }];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation Text',
        // Sets the style for the text to be displayed
        style: {
            textOverflow: 'Ellipsis'
        }
    }]
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"

```

```

        width={ '100%' }
        height={ '600px' }
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Appearance

- You can change the font style of the annotations with the font specific properties (fontSize, fontFamily, color). The following code illustrates how to customize the appearance of the annotation.
- The label's [bold](#), [italic](#), and [textDecoration](#) properties are used to style the label's text.
- The label's [fill](#), [strokeColor](#), and [strokeWidth](#) properties are used to define the background color and border color of the annotation and the [opacity](#) property is used to define the transparency of the annotations.
- The [visible](#) property of the annotation enables or disables the visibility of annotation.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation Text',
    // Sets the style for the text to be displayed
    style: {
      color: 'black',
      bold: true,
      italic: true,
      fontSize: 12,
      fontFamily: 'TimesNewRoman'
    }
  }]
}];
// initialize Diagram component
function App() {

```

```

    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation Text',
    // Sets the style for the text to be displayed
    style: {
      color: 'black',
      bold: true,
      italic: true,
      fontSize: 12,
      fontFamily: 'TimesNewRoman'
    }
  }]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

The fill, border, and opacity appearances of the text can also be customized with appearance specific properties of annotation. The following code illustrates how to customize background, opacity, and border of the annotation.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation Text',
    style: {
      color: 'black',
      fill: 'white',
      opacity: 0.7,
      strokeColor: 'black',
      strokeWidth: 2
    }
  }]
}];

// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
```



```

    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
      content: 'Annotation Text',
      style: {
        color: 'black',
        fill: 'white',
        opacity: 0.7,
        strokeColor: 'black',
        strokeWidth: 2
      }
    }]
  }
}
];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Interaction

Diagram allows annotation to be interacted by selecting, dragging, rotating, and resizing. Annotation interaction is disabled, by default. You can enable annotation interaction with the [constraints](#) property of annotation. You can also curtail the services of interaction by enabling either selecting, dragging, rotating, or resizing individually with the respective constraints property of annotation. The following code illustrates how to enable annotation interaction.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, AnnotationConstraints } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,

```

```

        height: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        },
        // Sets the annotation for the node
        annotations: [{
            content: 'Annotation Text',
            //Sets the constraints as Interaction
            constraints: AnnotationConstraints.Interaction
        }]
    }];
    // initialize Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    AnnotationConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation Text',
        //Sets the constraints as Interaction
        constraints: AnnotationConstraints.Interaction
    }]
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}

```

```

        height={'600px'}
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Edit

Diagram provides support to edit an annotation at runtime, either programmatically or interactively. By default, annotation is in view mode. But it can be brought to edit mode in two ways;

- Programmatically

By using [startTextEdit](#) method, edit the text through programmatically.

- Interactively
 1. By double-clicking the annotation.
 2. By selecting the item and pressing the F2 key.

Double-clicking any annotation will enable editing and the node enables first annotation editing. When the focus of editor is lost, the annotation for the node is updated.

When you double-click on the node/connector/diagram model, the [doubleClick](#) event gets triggered.

Read-only annotations

Diagram allows to create read-only annotations. You have to set the read-only property of annotation to enable/disable the read-only [constraints](#). The following code illustrates how to enable read-only mode.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, AnnotationConstraints } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the annotation for the node
  annotations: [{
    content: 'Annotation Text',
    //Sets the constraints as Read only
    constraints: AnnotationConstraints.ReadOnly
  }]
}]

```

```

    }];
    // initialize Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    AnnotationConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the annotation for the node
    annotations: [{
        content: 'Annotation Text',
        //Sets the constraints as Read only
        constraints: AnnotationConstraints.ReadOnly
    }]
}];
// initialize Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Drag Limit

- The diagram control now supports defining the [dragLimit](#) to the label while dragging from the connector and also update the position to the nearest segment offset.
- You can set the value to dragLimit [left](#), [right](#), [top](#), and [bottom](#) properties which allow the dragging of connector labels to a certain limit based on the user defined values.
- By default, drag limit will be disabled for the connector. It can be enabled by setting connector constraints as drag.
- The following code illustrates how to set a dragLimit for connector annotations.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, AnnotationConstraints } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector = [{
  id: 'connector2',
  type: 'Orthogonal',
  sourcePoint: { x: 300, y: 300 },
  targetPoint: { x: 400, y: 400 },
  annotations: [
    {
      content: 'connector1', offset: 0.5,
      //Enables drag constraints for a connector.
      constraints: AnnotationConstraints.Interaction |
AnnotationConstraints.Drag,
      //Set drag limit for a connector annotation.
      dragLimit: { left: 20, right: 20, top: 20, bottom: 20 }
    }
  ],
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connector}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  AnnotationConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let connector: ConnectorModel[] = [{
  id: 'connector2',
```

```

    type: 'Orthogonal',
    sourcePoint: { x: 300, y: 300 },
    targetPoint: { x: 400, y: 400 },
    annotations: [
      {
        content: 'connector1', offset: 0.5,
        // Enables drag constraints for a connector.
        constraints: AnnotationConstraints.Interaction |
        AnnotationConstraints.Drag,
        // Set drag limit for a connector annotation.
        dragLimit: { left: 20, right: 20, top: 20, bottom: 20 }
      }
    ],
  }
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={ '100%' }
      height={ '600px' }
      connectors={ connector }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Multiple annotations

You can add any number of annotations to a node or connector. The following code illustrates how to add multiple annotations to a node.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: 'node1',
  // Position of the node
  offsetX: 100,
  offsetY: 100,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Sets the multiple annotation for the node
  annotations: [{
    content: 'Left',
    offset: {
      x: 0.12,
      y: 0.1
    }
  }
]

```

```

        },
        {
            content: 'Center',
            offset: {
                x: 0.5,
                y: 0.5
            }
        },
        {
            content: 'Right',
            offset: {
                x: 0.82,
                y: 0.9
            }
        }
    ]
    });
    // initialize Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    AnnotationConstraints
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: 'node1',
    // Position of the node
    offsetX: 100,
    offsetY: 100,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Sets the multiple annotation for the node
    annotations: [{
        content: 'Left',
        offset: {
            x: 0.12,
            y: 0.1
        }
    }

```

```

    },
    {
      content: 'Center',
      offset: {
        x: 0.5,
        y: 0.5
      }
    },
    {
      content: 'Right',
      offset: {
        x: 0.82,
        y: 0.9
      }
    }
  ]
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

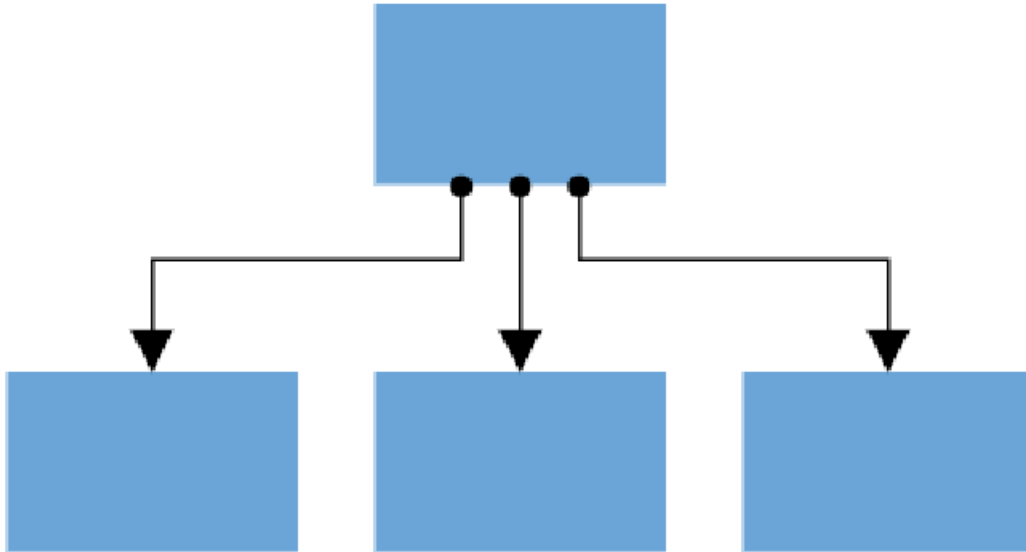
```

Constraints

The constraints property of annotation allows you to enable/disable certain annotation behaviors. For instance, you can disable annotation editing.

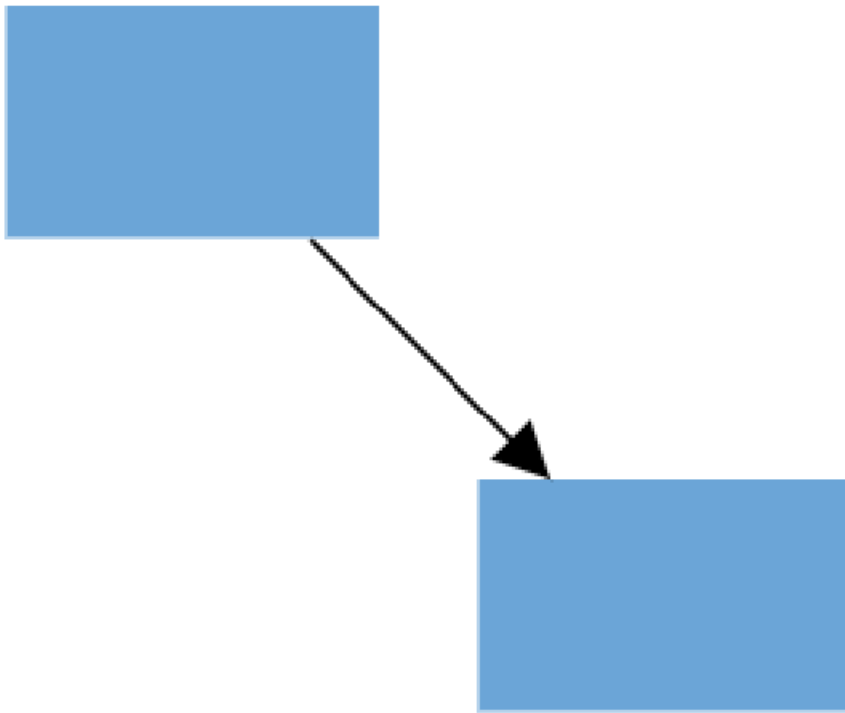
Ports in React Diagram component

Diagram provides support to define custom ports for making connections.

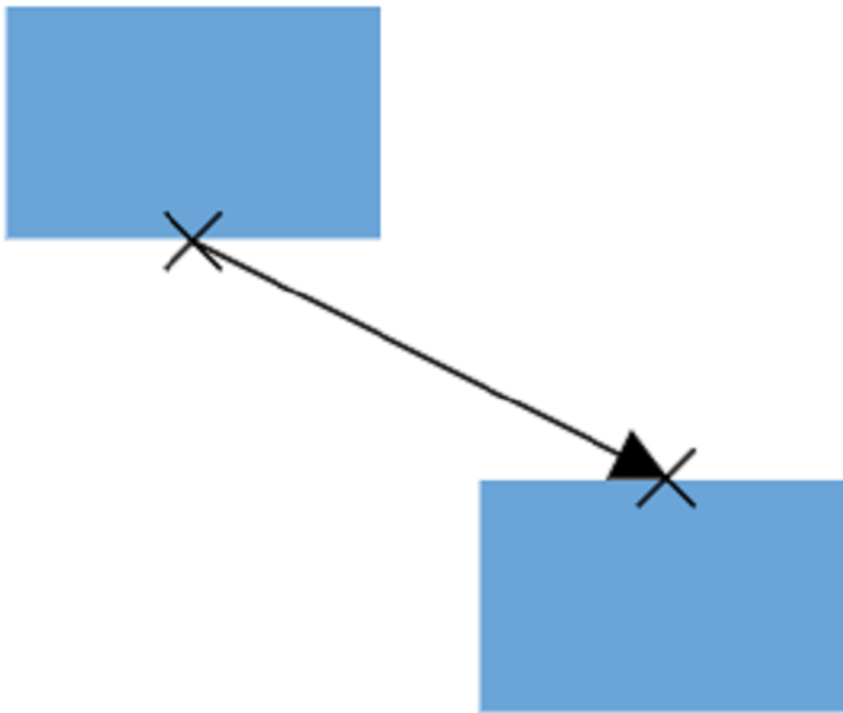


<!-- markdownlint-disable MD033 -->

When a connector is connected between two nodes, its end points are automatically docked to the node's nearest boundary as shown in the following image.



Ports act as the connection points of the node and allows to create connections with only those specific points as shown in the following image.



Create port

Add ports when initializing nodes

To add a connection port, define the port object and add it to node's ports collection. The `offset` property of port accepts an object of fractions and used to determine the position of ports. The following code illustrates how to add ports when initializing the node.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
```

```
offsetX: 250,
offsetY: 250,
// Size of the node
width: 100,
height: 100,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white'
},
// Initialize port collection
ports: [{
  // Sets the position for the port
  offset: {
    x: 0.5,
    y: 0.5
  },
  visibility: PortVisibility.Visible
}]
});
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
    // Add node
      nodes={node}
    // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
```

```
root.render(<App />);
```

```
,
```

Add ports at runtime

Add ports at runtime by using the client-side method [addPorts](#). The following code illustrates how to add ports to node at runtime.

The port's ID property is used to define the unique ID for the port and its further used to find the port at runtime.

If ID is not set, then default ID is automatically set.

```
`ts
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import {
```

```
  Diagram,
```

```
  DiagramComponent,
```

```
  NodeModel,
```

```
  PointPortModel,
```

```
  PortVisibility
```

```
} from "@syncfusion/ej2-react-diagrams";
```

```
let diagramInstance:DiagramComponent;
```

```
// A node is created and stored in nodes array.
```

```
let node: NodeModel[] = [{
```

```
  // Position of the node
```

```
  offsetX: 250,
```

```
  offsetY: 250,
```

```
  // Size of the node
```

```
  width: 100,
```

```
  height: 100,
```

```
  style: {
```

```
    fill: '#6BA5D7',
```

```
    strokeColor: 'white'
```

```
  },
```

```
  }];
```

```
// Initialize port collection
```

```
let port: PointPortModel[] = [{
```

```
id: 'port1',
offset: {
  x: 0,
  y: 0.5
},
visibility: PortVisibility.Visible
}{
id: 'port2',
offset: {
  x: 1,
  y: 0.5
},
visibility: PortVisibility.Visible
},
{
id: 'port3',
offset: {
  x: 0.5,
  y: 0
},
visibility: PortVisibility.Visible
},
{
id: 'port4',
offset: {
  x: 0.5,
  y: 1
},
visibility: PortVisibility.Visible
}
];
// initialize Diagram component
function App() {
```

```

return (
  <DiagramComponent
    id="container"
    ref={({diagram}) => (diagramInstance = diagram)}
    width={'100%'}
    height={'600px'}
    created = { () => {
      // Method to add ports through run time
      diagramInstance.addPorts(diagramInstance.nodes[0], port);
    }}
    // Add node
    nodes={node}
  // render initialized Diagram
  />
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Remove ports at runtime

Remove ports at runtime by using client-side method [removePorts](#). Refer to the following example which shows how to remove ports at runtime.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, PortVisibility } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection

```

```

        ports: [{
            id: 'port1',
            offset: {
                x: 0,
                y: 0.5
            },
            visibility: PortVisibility.Visible
        },
        {
            id: 'port2',
            offset: {
                x: 1,
                y: 0.5
            },
            visibility: PortVisibility.Visible
        },
        {
            id: 'port3',
            offset: {
                x: 0.5,
                y: 0
            },
            visibility: PortVisibility.Visible
        },
        {
            id: 'port4',
            offset: {
                x: 0.5,
                y: 1
            },
            visibility: PortVisibility.Visible
        }
    ]
    }];
let diagramInstance;
// initialize Diagram component
let ports = [{
    id: 'port1',
}, {
    id: 'port2',
}, {
    id: 'port3',
}, {
    id: 'port4',
}];
function App() {
    return (<DiagramComponent id="container" ref={ (diagram) =>
        (diagramInstance = diagram) } width={'100%'} height={'600px'} nodes={node}
        created={() => {
            diagramInstance.removePorts(diagramInstance.nodes[0], ports);
        }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```


INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  Node,
  PointPortModel,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection
  ports: [{
    id: 'port1',
    offset: {
      x: 0,
      y: 0.5
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port2',
    offset: {
      x: 1,
      y: 0.5
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port3',
    offset: {
      x: 0.5,
      y: 0
    },
    visibility: PortVisibility.Visible
  },
  {
    id: 'port4',
    offset: {
      x: 0.5,
      y: 1
    },
    visibility: PortVisibility.Visible
  }
]
```

```

    }
  ]
}];
let diagramInstance:DiagramComponent;
// initialize Diagram component
let ports: PointPortModel[] = [{
  id: 'port1',
}, {
  id: 'port2',
}, {
  id: 'port3',
}, {
  id: 'port4',
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created={() => {
        diagramInstance.removePorts(diagramInstance.nodes[0], ports);
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Update port at runtime

You can change any port properties at runtime and update it through the client-side method [dataBind](#).

The following code example illustrates how to change the port properties.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{

```

```
// Position of the node
offsetX: 250,
offsetY: 250,
// Size of the node
width: 100,
height: 100,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white'
},
// Initialize port collection
ports: [{
  offset: {
    x: 0.5,
    y: 0.5
  },
  visibility: PortVisibility.Visible
}]
});

let diagramInstance:DiagramComponent;
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={node}
      created={() => {
        diagramInstance.nodes[0].ports[0].offset = {
          x: 1,
          y: 1
```

```

};
diagramInstance.dataBind();
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Appearance

- The shape of port can be changed by using its shape property. To explore the different types of port shapes, refer to Port Shapes. If you need to render a custom shape, then you can set shape as path and define path using path data property of port.
- The appearance of ports can be customized by using [strokeColor](#), [strokeWidth](#), and [fill](#) properties of the port.
- Customize the port size by using the [width](#) and [height](#) properties of port.
- The ports [visibility](#) property allows you to define, when the port should be visible.

The following code illustrates how to change the appearance of port.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, PortVisibility } from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection
  ports: [{
    offset: {
      x: 1,
      y: 0.5
    },
    visibility: PortVisibility.Visible,
    //Set the style for the port
    style: {
      fill: 'red',

```

```

        strokeWidth: 2,
        strokeColor: 'black'
    },
    width: 12,
    height: 12,
    // Sets the shape of the port as Circle
    shape: 'Circle'
  ]]
  });
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  PortVisibility
} from "@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Initialize port collection
  ports: [{
    offset: {
      x: 1,
      y: 0.5
    },
    visibility: PortVisibility.Visible,
    //Set the style for the port
    style: {
      fill: 'red',
      strokeWidth: 2,
      strokeColor: 'black'
    },
    width: 12,
    height: 12,
    // Sets the shape of the port as Circle
    shape: 'Circle'
  }
  ]
  }
];

```

```

    ]]
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Offset

The offset property of port is used to align the port based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Constraints

The constraints property allows to enable/disable certain behaviors of ports. For more information about port

constraints, refer to [Port Constraints](#).

Constraints in React Diagram component

Constraints are used to enable/disable certain behaviors of the diagram, nodes and connectors. Constraints are provided as flagged enumerations, so that multiple behaviors can be enabled/disabled using Bitwise operators (&, |, ~, <<, etc.).

To know more about Bitwise operators, refer to [Bitwise Operations](#).

Diagram constraints

Diagram constraints allow to enable or disable the following behaviors:

- Page editing
- Bridging
- Zoom and pan
- Undo/redo
- Tooltip

The following example illustrates how to disable page editing using the diagram constraints.

`ts

```

function App() {
  return (
    <DiagramComponent
      id="container"
      width={700}

```

```
height={600}
constraints={
  DiagramConstraints.Default & ~DiagramConstraints.PageEditable
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`
```

For more information about diagram constraints, refer to [DiagramConstraints](#).

Node constraints

Node constraints allows to enable or disable the following behaviors of node. They are as follows:

- Selection
- Deletion
- Drag
- Resize
- Rotate
- Connect
- Shadow
- Tooltip

The following example illustrates how to disable rotation using the node constraints.

```
`ts
let nodes: NodeModel[] = [
  {
    id: "node",
    offsetX: 100,
    offsetY: 100,
    constraints: NodeConstraints.Default & ~NodeConstraints.Rotate,
  },
];

function App() {
  return (
    <DiagramComponent id="container" width={700} height={600} nodes={nodes} />
  );
}
```

```
}  
const root = ReactDOM.createRoot(document.getElementById("diagram"));  
root.render(<App />);  
`
```

For more information about node constraints, refer to [NodeConstraints](#).

Connector constraints

Connector constraints allow to enable or disable certain behaviors of connectors.

- Selection
- Deletion
- Drag
- Segment editing
- Tooltip
- Bridging

The following code illustrates how to disable selection by using connector constraints.

```
`ts  
let connectors: ConnectorModel[] = [{  
  id: 'connector1',  
  type: 'Straight',  
  sourcePoint: {  
    x: 100,  
    y: 100  
  },  
  targetPoint: {  
    x: 200,  
    y: 200  
  },  
  constraints: {  
    ConnectorConstraints.Default & ~ConnectorConstraints.Select  
  }  
}];  
function App() {  
  return (  
    <DiagramComponent id="container"  
      width={700}  
      height={600}
```



```
connectors={connectors}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

For more information about connector constraints, refer to [ConnectorConstraints](#).

Port constraints

You can enable or disable certain behaviors of port. They are as follows:

- Connect
- ConnectOnDrag

The following code illustrates how to disable creating connections with a port.

```
`ts
let nodes: NodeModel[] = [
{
id: "node",
offsetX: 100,
offsetY: 100,
ports: [
{
constraints: PortConstraints.None,
},
],
},
];
`
```

For more information about port constraints, refer to [PortConstraints](#).

Annotation constraints

You can enable or disable read-only mode for the annotations by using the annotation constraints.

The following code illustrates how to enable read-only mode for the annotations.

```
`ts
let nodes: NodeModel[] = [
```

```

{
  id: "node",
  offsetX: 100,
  offsetY: 100,
  annotations: [
    {
      id: "anotation_1",
      content: "annotation",
      constraints: AnnotationConstraints.ReadOnly,
    },
  ],
};

function App() {
  return (
    <DiagramComponent id="container" width={700} height={600} nodes={nodes} />
  );
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

```

For more details about annotation constraints, refer to [AnnotationConstraints](#).

Selector constraints

Selector visually represents the selected elements with certain editable thumbs. The visibility of the thumbs can be controlled with selector constraints. The part of selector is categorized as follows:

- Resizer
- Rotator
- User handles

The following code illustrates how to hide rotator.

```

{% raw %}
`ts
function App() {
  return (
    <DiagramComponent

```

```

id="container"
width={700}
height={600}
selectedItems={{
constraints: SelectorConstraints.All & ~SelectorConstraints.Rotate,
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

```

```
{% endraw %}
```

For more information about selector constraints, refer to [SelectorConstraints](#).

Snap constraints

Snap constraints control the visibility of gridlines and enable/disable snapping. Snap constraints allow to set the following behaviors.

- Show only horizontal or vertical gridlines.
- Show both horizontal and vertical gridlines.
- Snap to either horizontal or vertical gridlines.
- Snap to both horizontal and vertical gridlines.

The following code illustrates how to show only horizontal gridlines.

```

{% raw %}
`ts
function App() {
return (
<DiagramComponent
id="container"
width={700}
height={600}
snapSettings={{
constraints: SnapConstraints.ShowHorizontalLines,
}}
/>

```

```
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`
```

```
{% endraw %}
```

For more information about snap constraints, refer to [SnapConstraints](#).

Boundary constraints

Boundary constraints defines a boundary for the diagram inside which the interaction should be done. Boundary constraints allow to set the following behaviors.

- Infinite boundary
- Diagram sized boundary
- Page sized boundary

The following code illustrates how to limit the interaction done inside a diagram within a page.

```
{% raw %}
`ts
function App() {
  return (
    <DiagramComponent
      id="container"
      width={700}
      height={600}
      pageSettings={{
        boundaryconstraints: "Page",
      }}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

{% endraw %}
```

For more information about selector constraints, refer to [BoundaryConstraints](#).

Inherit behaviors

Some of the behaviors can be defined through both the specific object (node/connector) and diagram. When the behaviors are contradictorily defined through both, the actual behavior is set through inherit options.

The following code example illustrates how to inherit the line bridging behavior from the diagram model.

```
`ts
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  constraints = {
    ConnectorConstraints.Default & ConnectorConstraints.InheritBridging
  }
}];

function App() {
  return (
    <DiagramComponent
      id="container"
      width={700}
      height={600}
      connectors={connectors}
      constraints={DiagramConstraints.Default | DiagramConstraints.Bridging}

      <Inject services={[ConnectorBridging]} />
    </DiagramComponent>
  );
}
```

```

}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Bitwise operations

Bitwise operations are used to manipulate the flagged enumerations [enum]. In this section, Bitwise operations are illustrated by using node constraints. The same is applicable while working with node constraints, connector constraints, or port constraints.

Add operation

You can add or enable multiple values at a time by using Bitwise '|' (OR) operator.

```

`ts
node.constraints = NodeConstraints.Select | NodeConstraints.Rotate;
`

```

In the previous example, you can do both the selection and rotation operation.

Remove Operation

You can remove or disable values by using Bitwise '&~' (XOR) operator.

```

`ts
node.constraints = node.constraints & ~NodeConstraints.Rotate;
`

```

In the previous example, rotation is disabled but other constraints are enabled.

Check operation

You can check any value by using Bitwise '&' (AND) operator.

```

`ts
if ((node.constraints & NodeConstraints.Rotate) == NodeConstraints.Rotate);
`

```

In the previous example, check whether the rotate constraints are enabled in a node. When node constraints have rotate constraints, the expression returns a rotate constraint.

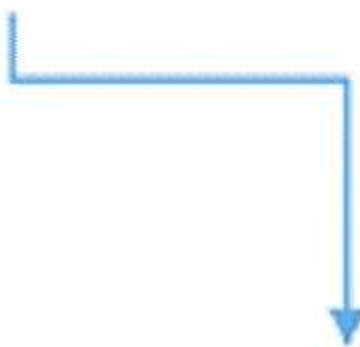
Interaction in React Diagram component

Selection

Selector provides a visual representation of selected elements. It behaves like a container and allows to update the size, position, and rotation angle of the selected elements through interaction and by using program. Single or multiple elements can be selected at a time.

Single selection

An element can be selected by clicking that element. During single click, all previously selected items are cleared. The following image shows how the selected elements are visually represented.



- While selecting the diagram elements, the following events can be used to do your customization.
- When selecting/unselecting the diagram elements, the [selectionChange](#) event gets triggered.

Selecting a group

When a child element of any group is clicked, its contained group is selected instead of the child element. With consecutive clicks on the selected element, selection is changed from top to bottom in the hierarchy of parent group to its children.

Multiple selection

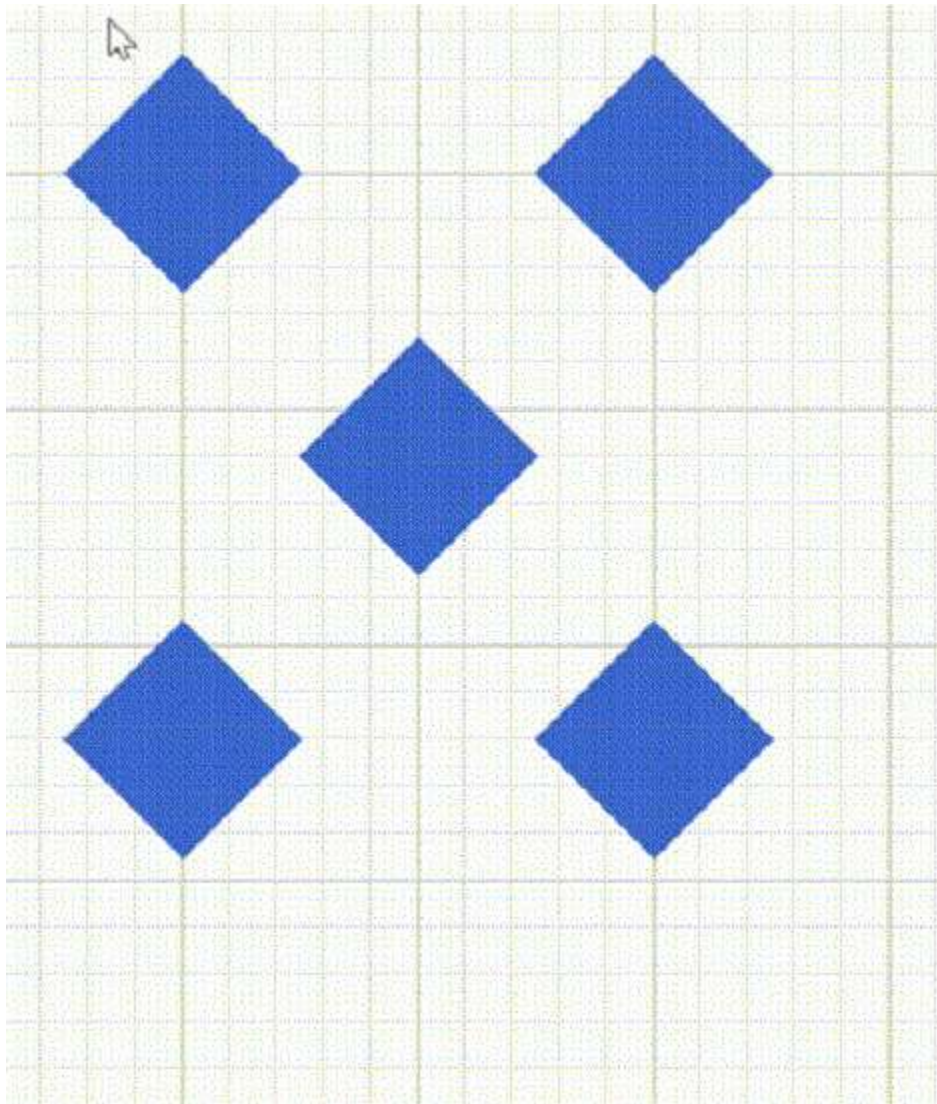
Multiple elements can be selected with the following ways:

- Ctrl+Click

During single click, any existing item in the selection list be cleared, and only the item clicked recently is there in the selection list. To avoid cleaning the old selected item, Ctrl key must be on hold when clicking.

- Selection rectangle/rubber band selection

Clicking and dragging the diagram area allows to create a rectangular region. The elements that are covered under the rectangular region are selected at the end.



Select/Unselect elements using program

The client-side methods [select](#) and [clearSelection](#) help to select or clear the selection of the elements at runtime. The following code example illustrates how to select or clear the selection of an item using program.

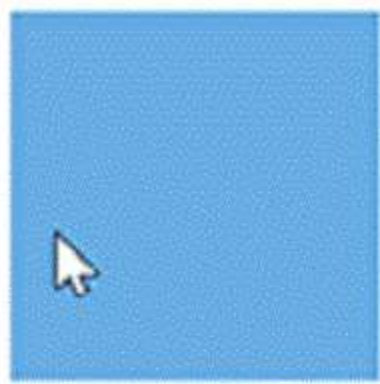
Get the current selected items from the [nodes](#) and [connectors](#) collection of the [selectedItems](#) property of the diagram model.

Select entire elements in diagram programmatically

The client-side method [selectAll](#) used to select all the elements such as nodes/connectors in the diagram. Refer to the following link which shows how to use [selectAll](#) method on the diagram.

Drag

- An object can be dragged by clicking and dragging it. When multiple elements are selected, dragging any one of the selected elements move every selected element.
- When you drag the elements in the diagram, the [positionChange](#) event gets triggered and to do customization in this event.



Resize

- Selector is surrounded by eight thumbs. When dragging these thumbs, selected items can be resized.
- When one corner of the selector is dragged, opposite corner is in a static position.
- When a node is resized, the [sizeChange](#) event gets triggered.



Note: While dragging and resizing, the objects are snapped towards the nearest objects to make better alignments. For better alignments, refer to **Snapping**.

Customize the **resize-thumb**

You can change the size of the node resize thumb and the connector end point handle by using the **handleSize** property. The appearance such as fill, stroke, and stroke width of the node resize thumb and connector end point handle can be customized by overriding the **e-diagram-resize-handle** and **e-diagram-endpoint-handle** classes respectively.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, } from "@syncfusion/ej2-react-diagrams";
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
```

```

        // Size of the node
        width: 100,
        height: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white'
        }
    }
    });
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node} selectedItems={{
        handleSize: 40,
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

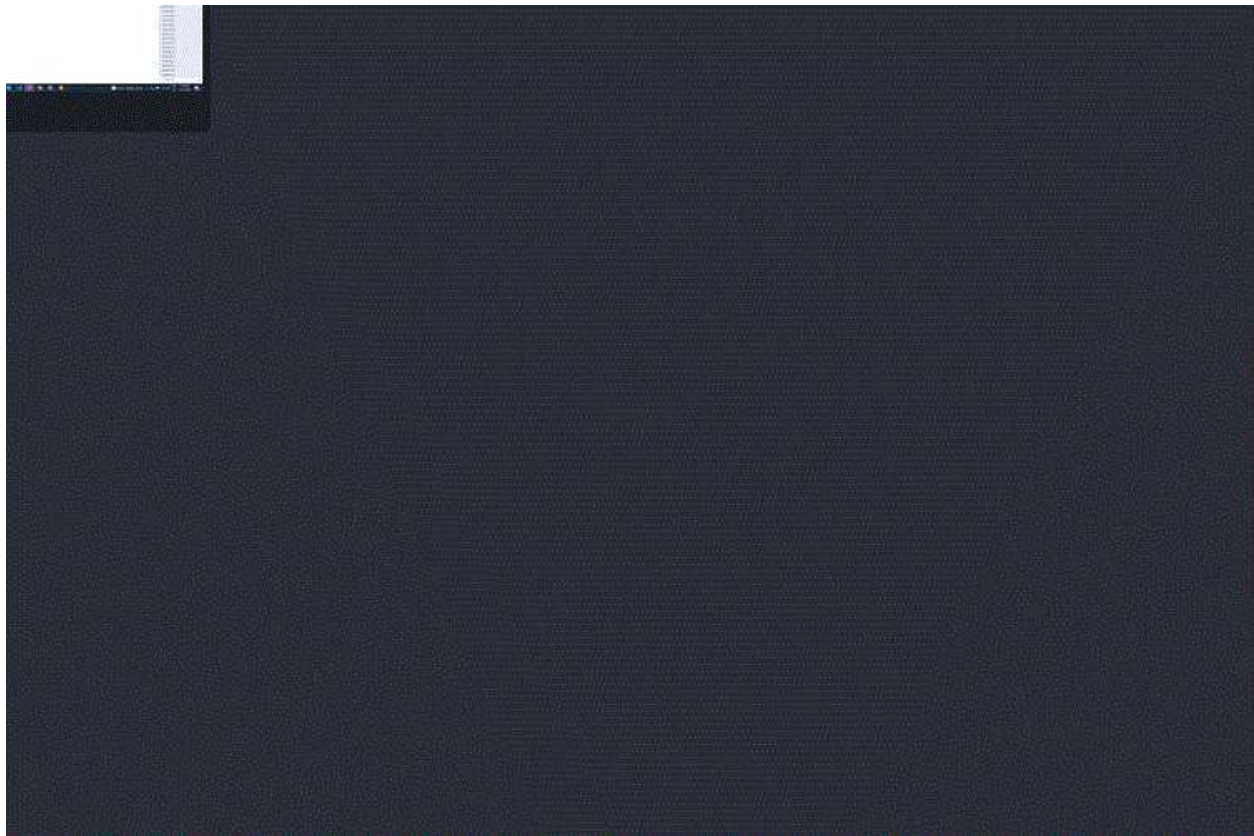
INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel,
    NodeModel,
} from "@syncfusion/ej2-react-diagrams";
let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    }
}
    ];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            nodes={node}
            selectedItems={{
                handleSize: 40,
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

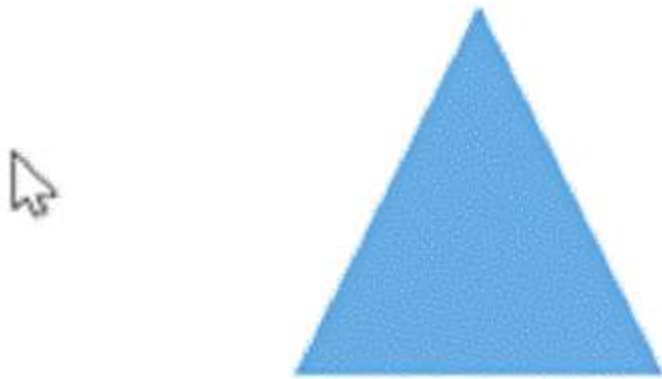
```

```
{% enddraw %}
```



Rotate

- A rotate handler is placed above the selector. Clicking and dragging the handler in a circular direction lead to rotate the node.
- The node is rotated with reference to the static pivot point.
- Pivot thumb (thumb at the middle of the node) appears while rotating the node to represent the static point.



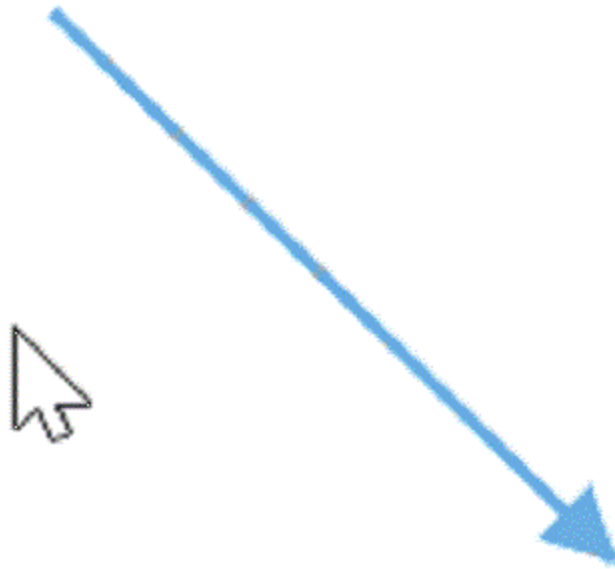
Connection editing

- Each segment of a selected connector is editable with some specific handles/thumbs.

Note: For connector editing, you have to inject the [ConnectorEditing](#) module.

End point handles

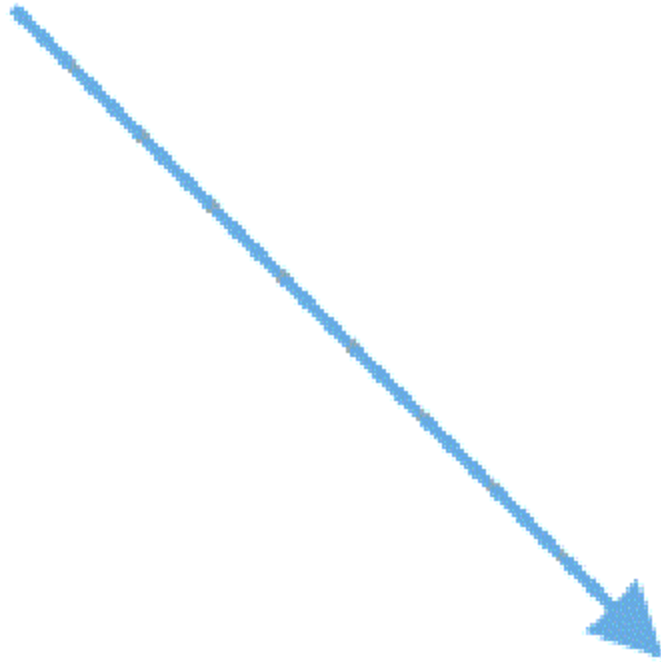
Source and target points of the selected connectors are represented with two handles. Clicking and dragging those handles help you to adjust the source and target points.



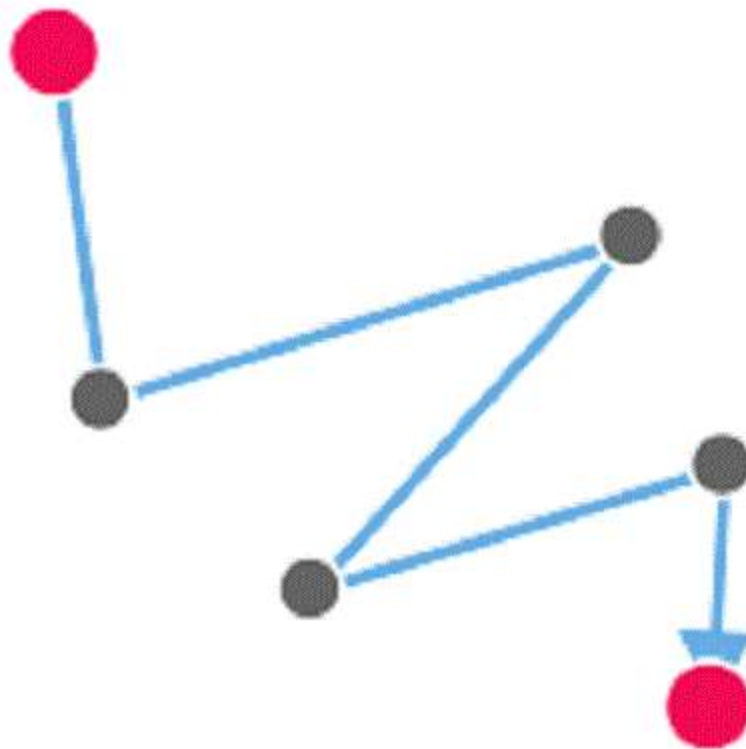
- If you drag the connector end points, then the following events can be used to do your customization.
- When the connector source point is changed, the [sourcePointChange](#) event gets triggered.
- When the connector target point is changed, the [targetPointChange](#) event gets triggered.
- When you connect connector with ports/node or disconnect from it, the [connectionChange](#) event gets triggered.

Straight segment editing

- End point of each straight segment is represented by a thumb that enables to edit the segment.
- Any number of new segments can be inserted into a straight line by clicking, when Shift and Ctrl keys are pressed (Ctrl+Shift+Click).

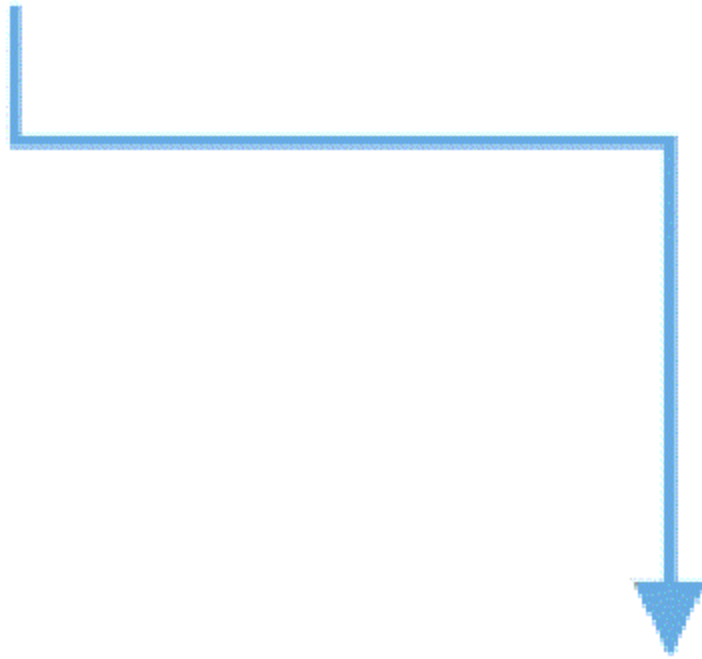


- Straight segments can be removed by clicking the segment end point, when Ctrl and Shift keys are pressed (Ctrl+Shift+Click).



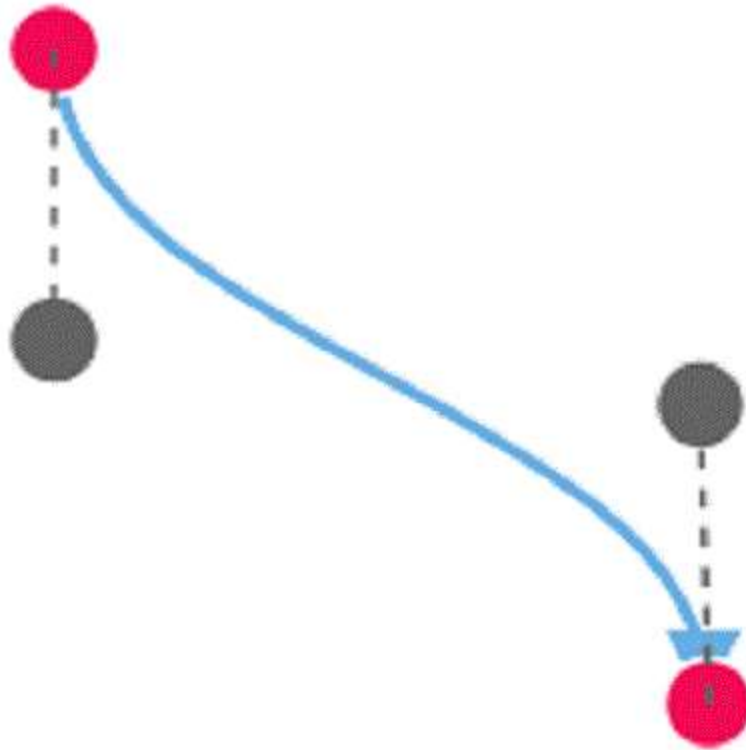
Orthogonal thumbs

- Orthogonal thumbs allow you to adjust the length of adjacent segments by clicking and dragging it.
- When necessary, some segments are added or removed automatically, when dragging the segment. This is to maintain proper routing of orthogonality between segments.



Bezier thumbs

- Bezier segments are annotated with two thumbs to represent the control points. Control points of the curve can be configured by clicking and dragging the control thumbs.



Drag and drop nodes over other elements

Diagram provides support to drop a node/connector over another node/connector. The [drop](#) event is raised to notify that an element is dropped over another one and it is disabled, by default. It can be enabled with the `constraints` property.

User handles

- User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the `userHandles` collection of the `selectedItems` property.
- The `name` property of user handle is used to define the name of the user handle and is further used to find the user handle at runtime and do any customization.

Alignment

User handles can be aligned relative to the node boundaries. It has `margin`, `offset`, `side`, `horizontalAlignment`, and `verticalAlignment` settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset

The [offset](#) property of [userHandles](#) is used to align the user handle based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Side

The [side](#) property of [userHandles](#) is used to align the user handle by using the [Top](#), [Bottom](#), [Left](#), and [Right](#) options.

Horizontal and vertical alignments

The [horizontalAlignment](#) property of [userHandles](#) is used to set how the user handle is horizontally aligned at the position based on the [offset](#). The [verticalAlignment](#) property is used to set how user handle is vertically aligned at the position.

Margin

Margin is an absolute value used to add some blank space in any one of its four sides. The [userHandles](#) can be displaced with the [margin](#) property.

Notification for the mouse button clicked

The diagram component notifies the mouse button clicked. For example, whenever the right mouse button is clicked, the clicked button is notified as right. The mouse click is notified with,

Notification	Description
Left	When the left mouse button is clicked, left is notified
Middle	When the mouse wheel is clicked, middle is notified
Right	When the right mouse button is clicked, right is notified

```
`ts
```

```
let diagramInstance: DiagramComponent;
```

```
// initialize Diagram component
```

```
function App() {
```

```
  return (
```

```
    <DiagramComponent
```

```
      id="container"
```

```
      ref={{(diagram) => (diagramInstance = diagram)}}
```

```
      width={'100%'}
```

```
      height={'600px'}
```

```
      click={{(args: IClickEventArgs) => {
```

```
        //Obtains the button clicked
```

```
        let button = args.button;
```

```
      }}
    )
```

```
  />
```

```
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Appearance

The appearance of the user handle can be customized by using the [size](#), [borderColor](#), [backgroundColor](#), [visible](#), [pathData](#), and [pathColor](#) properties of the [userHandles](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, MoveTool, SelectorConstraints, randomId,
cloneObject } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let shape = {
  type: 'Basic',
  shape: 'Rectangle'
};
let node1 = [{
  id: 'node',
  offsetX: 100,
  offsetY: 100,
  shape: shape
}];
let handles = [{
  name: 'clone',
  pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z
M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-
2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-
30V34.4h30V72.5z',
  visible: true,
  offset: 0,
  side: 'Bottom',
  pathColor: "white",
  margin: {
    top: 0,
    bottom: 0,
    left: 0,
    right: 0
  }
}];
function App() {
  return (<DiagramComponent id="container" ref={diagram =>
(diagramInstance = diagram)} width={"100%"} height={"600px"} nodes={node1}
selectedItems={{
  constraints: SelectorConstraints.UserHandle,
  userHandles: handles
}}
  //set Node default value
  getNodeDefaults=(node) => {
```

```

        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = '#6BA5D7';
        return node;
    }}
    //set CustomTool
    getCustomTool={getTool}/>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
function getTool(action) {
    let tool;
    if (action === 'clone') {
        tool = new CloneTool(diagramInstance.commandHandler);
    }
    return tool;
}
//Defines the clone tool used to copy Node/Connector
class CloneTool extends MoveTool {
    mouseDown(args) {
        let newObject;
        if (diagramInstance.selectedItems.nodes.length > 0) {
            newObject = cloneObject(diagramInstance.selectedItems.nodes[0]);
        }
        else {
            newObject =
cloneObject(diagramInstance.selectedItems.connectors[0]);
        }
        newObject.id += randomId();
        diagramInstance.paste([newObject]);
        args.source = diagramInstance.nodes[diagramInstance.nodes.length -
1];
        args.sourceWrapper = args.source.wrapper;
        super.mouseDown(args);
        this.inAction = true;
    }
}
{% enddraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    DiagramComponent,
    NodeModel,
    ConnectorModel,
    Diagram,
    BasicShapeModel,
    MoveTool,
    MouseEventArgs,
    IElement,
    UserHandleModel,
    ToolBase,
    SelectorConstraints,

```

```

    Actions,
    randomId,
    cloneObject,
    Node,
    Side,
    SnapConstraints
  } from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let shape: BasicShapeModel = {
  type: 'Basic',
  shape: 'Rectangle'
};
let model: NodeModel[] = [{
  id: 'node',
  offsetX: 100,
  offsetY: 100,
  shape: shape
}];
let handles: UserHandleModel[] = [{
  name: 'clone',
  pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z
M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-
2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-
30V34.4h30V72.5z',
  visible: true,
  offset: 0,
  side: 'Bottom',
  pathColor: "white",
  margin: {
    top: 0,
    bottom: 0,
    left: 0,
    right: 0
  }
}];
function App() {
  return (
    <DiagramComponent id="container" ref={diagram => (diagramInstance =
diagram)}
    width = {
      "100%"
    }
    height = {
      "600px"
    }
    nodes = {
      model
    }
    selectedItems = {
      {
        constraints: SelectorConstraints.UserHandle,
        userHandles: handles
      }
    }
    //set Node default value
    getNodeDefaults = {
      (node: NodeModel): NodeModel => {

```

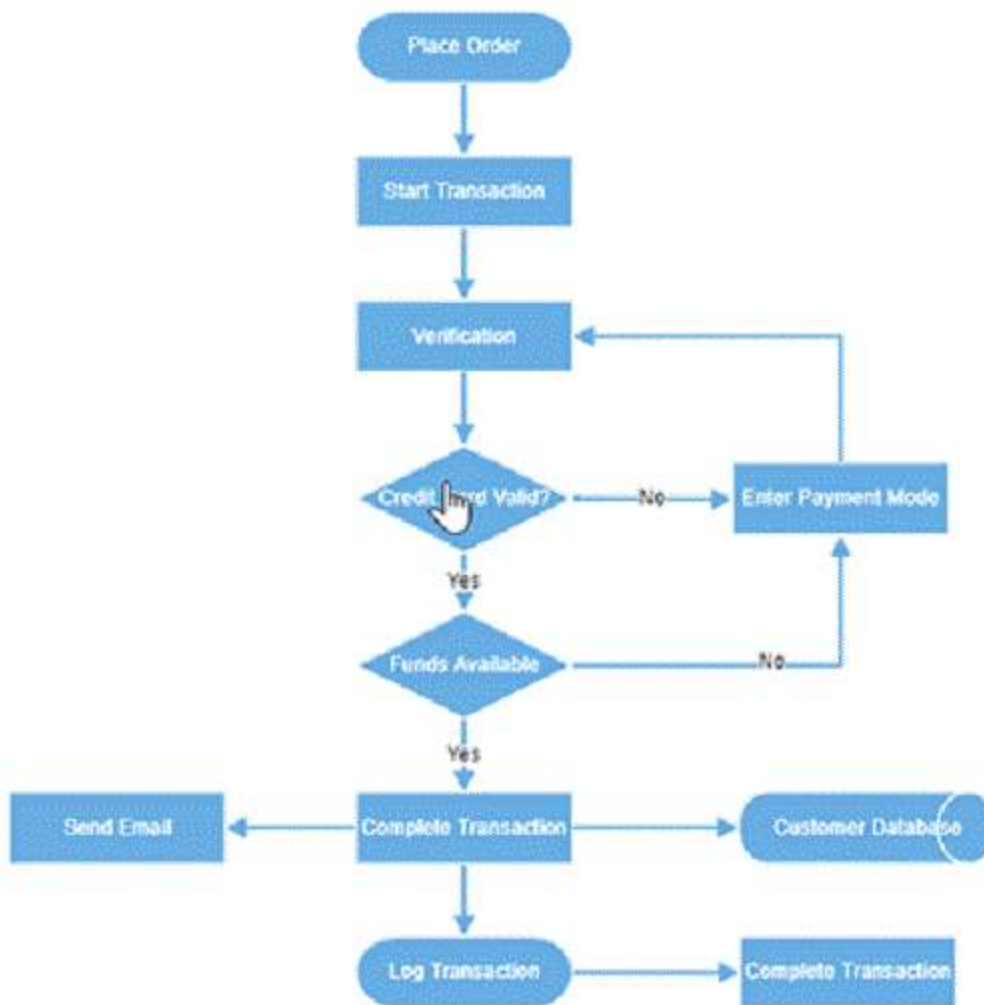
```

        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = '#6BA5D7';
        return node;
    }
}
//set CustomTool
getCustomTool = {
    getTool
}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
function getTool(action: string): ToolBase {
    let tool: ToolBase;
    if (action === 'clone') {
        tool = new CloneTool(diagramInstance.commandHandler);
    }
    return tool;
}
//Defines the clone tool used to copy Node/Connector
class CloneTool extends MoveTool {
    public mouseDown(args: MouseEventArgs): void {
        let newObject: any;
        if (diagramInstance.selectedItems.nodes.length > 0) {
            newObject = cloneObject(diagramInstance.selectedItems.nodes[0])
as NodeModel;
        } else {
            newObject =
cloneObject(diagramInstance.selectedItems.connectors[0]) as ConnectorModel;
        }
        newObject.id += randomId();
        diagramInstance.paste([newObject]);
        args.source = diagramInstance.nodes[diagramInstance.nodes.length -
1] as IElement;
        args.sourceWrapper = args.source.wrapper;
        super.mouseDown(args);
        this.inAction = true;
    }
}

```

Zoom pan

- When a large diagram is loaded, only certain portion of the diagram is visible. The remaining portions are clipped. Clipped portions can be explored by scrolling the scrollbars or panning the diagram.
- Diagram can be zoomed in or out by using Ctrl + mouse wheel.
- When the diagram is zoomed or panned, the [scrollChange](#) event gets triggered.



Zoom pan status

Diagram provides the support to notify the pan status of the zoom pan tool. When ever the diagram is panning the [scrollChange](#) event is triggered and hence the pan status can be obtained. The pan status is notified with Start, Progress, and Completed.

Pan Status	Description
Start	When the mouse is clicked and dragged the status is notified as start.
Progress	When the mouse is in motion the status is notified as progress.
Completed	When panning is stopped the status is notified with completed.

```
let diagramInstance: DiagramComponent;
```

```
// initialize Diagram component
```



```

function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      scrollChange={(args: IScrollChangeEventArgs) => {
        //Obtains the pan status
        let panStatus = args.panState;
      }}
      created={() => {
        diagramInstance.tool = DiagramTools.ZoomPan;
        diagramInstance.dataBind();
      }}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Keyboard

Diagram provides support to interact with the elements with key gestures. By default, some in-built commands are bound with a relevant set of key combinations.

The following table illustrates those commands with the associated key values.

Shortcut Key	Command	Description
-----	-----	-----
Ctrl + A	selectAll	Select all nodes/connectors in the diagram.
Ctrl + C	copy	Copy the diagram selected elements.
Ctrl + V	paste	Pastes the copied elements.
Ctrl + X	cut	Cuts the selected elements.
Ctrl + Z	undo	Reverses the last editing action performed on the diagram.
Ctrl + Y	redo	Restores the last editing action when no other actions have occurred since the last undo on the diagram.

| Delete | delete | Deletes the selected elements. |

| Ctrl/Shift + Click on object | | Multiple selection (Selector binds all selected nodes/connectors). |

| Up Arrow | nudge("up") | **nudgeUp**: Moves the selected elements towards up by one pixel. |

| Down Arrow | nudge("down") | **nudgeDown**: Moves the selected elements towards down by one pixel. |

| Left Arrow | nudge("left") | **nudgeLeft**: Moves the selected elements towards left by one pixel. |

| Right Arrow | nudge("right") | **nudgeRight**: Moves the selected elements towards right by one pixel. |

| Ctrl + MouseWheel | zoom | Zoom (Zoom in/Zoom out the diagram). |

| F2 | **startLabelEditing** | Starts to edit the label of selected element. |

| Esc | **endLabelEditing** | Sets the label mode as view and stops editing. |

| Tab | Tab to Focus | Select the diagram element based on the rendering order when using the "Tab" key. |

| Shift + Tab | Go to Previous Object | This command is employed to shift the selection to the preceding object based on the z-order. |

| Control + B | Bold | Toggle bold formatting for the selected text. |

| Control + I | Italic | Toggle italic formatting for the selected text. |

| Control + U | Underline | Toggle underline formatting for the selected text. |

| Control + D | Duplicate | Duplicate a selected shape. |

| Control + G | Group | Group together multiple selected shapes, allowing them to be treated as a single shape. |

| Control + Shift + U | UnGroup | Ungroup shapes within a previously grouped selection. |

| Control + R | Rotate clockwise | Rotate the selected nodes in clockwise. |

| Control + L | Rotate anti-clockwise | Rotate the selected nodes in counterclockwise. |

| Control + H | Flip Horizontal | Flip the selected elements horizontally. |

| Control + J | Flip Vertical | Flip the selected elements vertically. |

| Control + 1 | Pointer tool | Activate the pointer tool. |

| Control + 2 | Text tool | Activate the text tool. |

| Control + 3 | Connector tool | Activate the connector tool. |

| Control + 5 | Freeform tool | Activate the freeform tool. |

| Control + 6 | Line tool | Activate the polyline tool. |

| Control + + | Zoom In | Zoom in the diagram. |

| Control + - | Zoom Out | Zoom out the diagram. |

| Shift + Up Arrow | Up | Moves the selected elements towards up by 5 pixel. |

| Shift + Down Arrow | Down | Moves the selected elements towards down by 5 pixel. |

- | Shift + Left Arrow | Left | Moves the selected elements towards left by 5 pixel. |
- | Shift + Right Arrow | Right | Moves the selected elements towards right by 5 pixel. |
- | Control + Shift + L | Align Text Left | Align the selected text to the left. |
- | Control + Shift + C | Center Text Horizontally | Center the selected text horizontally. |
- | Control + Shift + R | Align Text Right | Align the selected text to the right. |
- | Control + Shift + J | Justify Text Horizontally | Justify the selected text, aligning it to both the left and right margins. |
- | Control + Shift + E | Top-align Text Vertically | Align the selected text to the top vertically. |
- | Control + Shift + M | Center Text Vertically | Center the selected text vertically. |
- | Control + Shift + V | Bottom-align Text Vertically | Align the selected text to the bottom vertically. |
- | Control + Shift + B | Send To Back | Send the selected shape backward in the stacking order, making it appear behind other shapes. |
- | Control + Shift + F | Bring To Front | Bring the selected shape forward in the stacking order, making it appear in front of other shapes. |
- | Control + [| Send Backward | Move the selected shape one step backward in the layer order. |
- | Control +] | Bring Forward | Move the selected shape one step forward in the layer order. |

See Also

- [How to create diagram nodes using drawing tools](#)
- [How to create diagram connectors using drawing tools](#)
- [How to disable the diagram interaction](#)
- [How to control the diagram history](#)
- [How to create overview control to the diagram](#)

Tools in React Diagram component

Drawing tools

Drawing tool allows you to draw any kind of node/connector during runtime by clicking and dragging on the diagram page.

Shapes

To draw a shape, set the JSON of that shape to the drawType property of the diagram and activate the drawing tool by using the [tool](#) property. The following code example illustrates how to draw a rectangle at runtime.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  BasicShapeModel,
```

```
NodeModel,  
DiagramTools,  
DiagramComponent  
} from "@syncfusion/ej2-react-diagrams";  
let diagramInstance: DiagramComponent;  
function App() {  
  return (  
    <DiagramComponent  
      id="container"  
      ref={(diagram) => (diagramInstance = diagram)}  
      width={700}  
      height={700}  
      created={() => {  
        //JSON to create a rectangle  
        let drawingshape: BasicShapeModel = {  
          type: 'Basic',  
          shape: 'Rectangle',  
        };  
        let node: NodeModel = {  
          shape: drawingshape,  
        };  
        diagramInstance.drawingObject = node;  
        //To draw an object once, activate draw once  
        diagramInstance.tool = DiagramTools.DrawOnce;  
        diagramInstance.dataBind();  
      }}  
      //customize the appearance of the shape  
      getNodeDefaults={(obj: NodeModel) => {  
        obj.borderWidth = 1;  
        obj.style = {  
          fill: '#6BA5D7',  
          strokeWidth: 2,  
          strokeColor: '#6BA5D7',
```

```

};
return obj;
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

The following code example illustrates how to draw a path.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  PathModel,
  NodeModel,
  DiagramTools,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created = {
        () => {
          //JSON to create a path
          let node: NodeModel = {
            id: "Path",

```

```
style: {
  fill: "#fbe172"
},
annotations: [{
  content: "Path"
}],
shape: {
  type: 'Path',
  data: 'M13.560 67.524 L 21.941 41.731 L 0.000 25.790 L 27.120 25.790 L 35.501 0.000 L 43.882 25.790 L
71.000 25.790 L 49.061 41.731 L 57.441 67.524 L 35.501 51.583 z'
}
as PathModel
};
diagramInstance.drawingObject = node;
//To draw an object once, activate draw once
diagramInstance.tool = DiagramTools.DrawOnce;
diagramInstance.dataBind();
}
}
//customize the appearance of the shape
getNodeDefaults=(obj, diagramInstance) => {
  obj.borderWidth = 1;
  obj.style = {
    fill: '#6BA5D7',
    strokeWidth: 2,
    strokeColor: '#6BA5D7',
  };
  return obj;
}
}/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
```

```
root.render(<App />);
```

```
,
```

Connectors

To draw connectors, set the JSON of the connector to the `drawType` property. The drawing [tool](#) can be activated by using the `tool` property. The following code example illustrates how to draw a straight line connector.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  ConnectorModel,
  NodeModel,
  DiagramTools,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created = {
        () => {
          //JSON to create a Connector
          let connectors: ConnectorModel = {
            id: 'connector1',
            type: 'Straight',
            segments: [{
              type: "polyline"
            }]
          }
        }
      }
    >
```

```

diagramInstance.drawingObject = connectors;
//To draw an object once, activate draw once
diagramInstance.tool = DiagramTools.DrawOnce;
diagramInstance.dataBind();
}
}
//customize the appearance of the shape
getNodeDefaults=(obj, diagramInstance) => {
  obj.borderWidth = 1;
  obj.style = {
    fill: '#6BA5D7',
    strokeWidth: 2,
    strokeColor: '#6BA5D7',
  };
  return obj;
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Text

Diagram allows you to create a textNode, when you click on the diagram page. The following code illustrates how to draw a text.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  TextModel,
  NodeModel,
  DiagramTools,
  DiagramComponent

```



```
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={{(diagram) => (diagramInstance = diagram)}}
      width={700}
      height={700}
      created={() => {
        //JSON to create a text
        let node: NodeModel = {
          shape: {
            type: 'Text',
          }as TextModel
        };
        diagramInstance.drawingObject = node;
        //To draw an object once, activate draw once
        diagramInstance.tool = DiagramTools.DrawOnce;
        diagramInstance.dataBind();
      }}
      //customize the appearance of the shape
      getNodeDefaults={{(obj, diagramInstance) => {
        obj.borderWidth = 1;
        obj.style = {
          fill: '#6BA5D7',
          strokeWidth: 2,
          strokeColor: '#6BA5D7',
        };
        return obj;
      }}
    />
  );
}
```

```

}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Once you activate the TextTool, perform label editing of a node/connector.

Polygon shape

Diagram allows to create the polygon shape by clicking and moving the mouse at runtime on the diagram page.

The following code illustrates how to draw a polygon shape.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  BasicShapeModel,
  NodeModel,
  DiagramTools,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created = {
        () => {
          let drawingshape: BasicShapeModel = {
            type: 'Basic',
            shape: 'Polygon'
          };

```

```
//JSON to create a polygon
let node: NodeModel = {
  shape: drawingshape
};
diagramInstance.drawingObject = node;
//To draw an object once, activate draw once
diagramInstance.tool = DiagramTools.DrawOnce;
diagramInstance.dataBind();
}
}
//customize the appearance of the shape
getNodeDefaults=((obj, diagramInstance) => {
  obj.borderWidth = 1;
  obj.style = {
    fill: '#6BA5D7',
    strokeWidth: 2,
    strokeColor: '#6BA5D7',
  };
  return obj;
})
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Polyline Connector

Diagram allows to create the polyline segments with straight lines and angled vertices at the control points by clicking and moving the mouse at runtime on the diagram page.

The following code illustrates how to draw a polyline connector.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
```

```
Diagram,
ConnectorModel,
DiagramTools,
DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={({diagram}) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created = {
        () => {
          let connector: ConnectorModel = {
            id: 'connector1',
            type: 'Polyline'
          };
          diagramInstance.drawingObject = connector;
          //To draw an object once, activate draw once
          diagramInstance.tool = DiagramTools.DrawOnce;
          diagramInstance.dataBind();
        }
      }
    //customize the appearance of the shape
    getNodeDefaults=({obj, diagramInstance}) => {
      obj.borderWidth = 1;
      obj.style = {
        fill: '#6BA5D7',
        strokeWidth: 2,
        strokeColor: '#6BA5D7',
      };
    }
  );
}
```

```

return obj;
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Tool selection

There are some functionalities that can be achieved by clicking and dragging on the diagram surface. They are as follows.

- Draw selection rectangle: MultipleSelect tool
- Pan the diagram: Zoom pan
- Draw nodes/connectors: DrawOnce/DrawOnce

As all the three behaviors are completely different, you can achieve only one behavior at a time based on the tool that you choose.

When more than one of those tools are applied, a tool is activated based on the precedence given in the following table.

Precedence	Tools	Description
1st	ContinuesDraw	Allows you to draw the nodes or connectors continuously. Once it is activated, you cannot perform any other interaction in the diagram.
2nd	DrawOnce	Allows you to draw a single node or connector. Once you complete the DrawOnce action, SingleSelect, and MultipleSelect tools are automatically enabled.
3rd	ZoomPan	Allows you to pan the diagram. When you enable both the SingleSelect and ZoomPan tools, you can perform the basic interaction as the cursor hovers node/connector. Panning is enabled when cursor hovers the diagram.
4th	MultipleSelect	Allows you to select multiple nodes and connectors. When you enable both the MultipleSelect and ZoomPan tools, cursor hovers the diagram. When panning is enabled, you cannot select multiple nodes.
5th	SingleSelect	Allows you to select individual nodes or connectors.
6th	None	Disables all tools.

Set the desired [tool](#) to the tool property of the diagram model. The following code illustrates how to enable Zoom pan in the diagram.

```

`ts
import * as React from "react";

```

```
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
},
{
  id: 'Init',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 140,
  annotations: [{
    id: 'label2',
    content: 'End'
  }],
  shape: {
```

```
type: 'Flow',
shape: 'process'
},
annotations: [{
content: 'var i = 0;'
}]
}
];
let connectors: ConnectorModel[] = [{
// Name of the connector
id: "connector1",
style: {
strokeColor: '#6BA5D7',
fill: '#6BA5D7',
strokeWidth: 2
},
targetDecorator: {
style: {
fill: '#6BA5D7',
strokeColor: '#6BA5D7'
}
},
// ID of the source and target nodes
sourceID: "Start",
targetID: "Init",
connectorSpacing: 7,
type: 'Orthogonal'
}];
function App() {
return (
<DiagramComponent
id="container"
ref={{(diagram) => (diagramInstance = diagram)}}

```

```

width={700}
height={700}
tool={DiagramTools.DrawOnce | DiagramTools.ZoomPan}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Events

[elementDraw](#) event is triggered when node or connector is drawn using drawing tool.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  ConnectorModel,
  NodeModel,
  DiagramTools,
  DiagramComponent,
  IElementDrawEventArgs
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={({diagram}) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created={() => {
        //JSON to create a Connector
        let connectors = {

```



```

id: 'connector1',
type: 'Straight',
segments: [
{
type: 'Straight',
},
],
};
diagramInstance.drawingObject = connectors;
//To draw an object once, activate draw once
diagramInstance.tool = DiagramTools.DrawOnce;
diagramInstance.dataBind();
}}
elementDraw={() => {
alert('Event Triggered');
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Freehand Drawing

Diagram has support for free-hand drawing to draw anything on the diagram page independently. Free-hand drawing will be enabled by using the drawingObject property and setting its value to Freehand.

The following code illustrates how to draw a freehand drawing.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
Diagram,
ConnectorModel,
NodeModel,
DiagramTools,

```

DiagramComponent

```

} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={700}
      height={700}
      created={() => {
        //JSON to create a Connector
        let connectors: ConnectorModel = {
          id: 'connector1',
          type: 'Freehand',
        };
        diagramInstance.drawingObject = connectors;
        //To draw an object once, activate draw once
        diagramInstance.tool = DiagramTools.DrawOnce;
        diagramInstance.dataBind();
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Grid lines in React Diagram component

Gridlines are the pattern of lines drawn behind the diagram elements. It provides a visual guidance while dragging or arranging the objects on the diagram surface.

The model's [snapSettings](#) property is used to customize the gridlines and control the snapping behavior in the diagram.

Customize the gridlines visibility

The [snapSettings.snapConstraints](#) enables you to show/hide the gridlines. The following code example illustrates how to show or hide gridlines.

If you need to enable snapping, then inject snapping module into the diagram.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let snapSettings = {
  constraints: SnapConstraints.ShowLines,
};
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  // Add node
  nodes={node} snapSettings={snapSettings}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  SnapSettingsModel,
  SnapConstraints,
  UndoRedo,
  GridlinesModel,
  Snapping,
  Inject
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
  constraints: SnapConstraints.ShowLines,
};
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
```

```

    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      snapSettings={snapSettings}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

To show only horizontal/vertical gridlines or to hide gridlines, refer to [Constraints](#).

Appearance

The appearance of the gridlines can be customized by using a set of predefined properties.

- The [horizontalGridLines](#) and the [verticalGridLines](#) properties allow to customize the appearance of the horizontal and vertical gridlines respectively.
- The horizontal gridlines [lineColor](#) and [lineDashArray](#) properties are used to customizes the line color and line style of the horizontal gridlines.
- The vertical gridlines [lineColor](#) and [lineDashArray](#) properties are used to customizes the line color and line style of the vertical gridlines.

The following code example illustrates how to customize the appearance of gridlines.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let gridlines = {
  lineColor: "blue",
  lineDashArray: '2 2'
};
let snapSettings = {
  constraints: SnapConstraints.ShowLines,

```

```

    horizontalGridlines: gridlines,
    verticalGridlines: gridlines
  };
  // A node is created and stored in nodes array.
  let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
  // initialize Diagram component
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node} snapSettings={snapSettings}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  SnapSettingsModel,
  SnapConstraints,
  UndoRedo,
  GridlinesModel,
  Snapping,
  Inject
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let gridlines: GridlinesModel = {
  lineColor: "blue",
  lineDashArray: '2 2'
};
let snapSettings: SnapSettingsModel = {
  constraints: SnapConstraints.ShowLines,
  horizontalGridlines: gridlines,
  verticalGridlines: gridlines
};
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,

```

```

    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      snapSettings={snapSettings}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Line intervals

Thickness and the space between gridlines can be customized by using horizontal gridlines's [linesInterval](#) and vertical gridlines's [linesInterval](#) properties. In the lines interval collections, values at the odd places are referred as the thickness of lines and values at the even places are referred as the space between gridlines.

The following code example illustrates how to customize the thickness of lines and the line intervals.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let interval;
interval = [
    1,
    9,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
```

```

    9.75,
    0.25,
    9.75,
    0.25,
    9.75,
    0.25,
    9.75
];
let gridlines = {
  lineColor: 'blue',
  lineDashArray: '2 2',
  lineIntervals: interval
};
let snapSettings = {
  constraints: SnapConstraints.ShowLines,
  horizontalGridlines: gridlines,
  verticalGridlines: gridlines
};
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node} snapSettings={snapSettings}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  SnapSettingsModel,
  SnapConstraints,
  UndoRedo,
  GridlinesModel,
  Snapping,
  Inject
} from "@syncfusion/ej2-react-diagrams";

```

```
Diagram.Inject(Snapping);
let interval: number[];
interval = [
  1,
  9,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75,
  0.25,
  9.75
];
let gridlines: GridlinesModel = {
  lineColor: 'blue',
  lineDashArray: '2 2',
  lineIntervals: interval
};
let snapSettings: SnapSettingsModel = {
  constraints: SnapConstraints.ShowLines,
  horizontalGridlines: gridlines,
  verticalGridlines: gridlines
};
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
    >
```



```

        snapSettings={snapSettings}
        // render initialized Diagram
    />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Snapping

Snap to lines

This feature allows the diagram objects to snap to the nearest intersection of gridlines while being dragged or resized. This feature enables easier alignment during layout or design.

Snapping to gridlines can be enabled/disabled with the [snapSettings.snapConstraints](#). The following code example illustrates how to enable/disable the snapping to gridlines.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let snapSettings = {
    constraints: SnapConstraints.SnapToLines | SnapConstraints.ShowLines
};
// A node is created and stored in nodes array.
let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white'
    },
    // Text(label) added to the node
}];
// initialize Diagram component
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
        // Add node
        nodes={node} snapSettings={snapSettings}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,

```

```

    NodeModel,
    DiagramComponent,
    SnapSettingsModel,
    SnapConstraints,
    UndoRedo,
    GridlinesModel,
    Snapping,
    Inject
  } from "@syncfusion/ej2-react-diagrams";
  Diagram.Inject(Snapping);
  let snapSettings: SnapSettingsModel = {
    constraints: SnapConstraints.SnapToLines | SnapConstraints.ShowLines
  };
  // A node is created and stored in nodes array.
  let node: NodeModel[] = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
  // initialize Diagram component
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={'100%'}
        height={'600px'}
        // Add node
        nodes={node}
        snapSettings={snapSettings}
        // render initialized Diagram
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Customization of snap intervals

By default, the objects are snapped towards the nearest gridline. The gridline or position towards where the diagram object snaps can be customized with the horizontal gridlines's [snapInterval](#) and the vertical gridlines's [snapInterval](#) properties.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);

```

```

let gridlines = {
  // Defines the snap interval for object
  snapIntervals: [10],
};
let snapSettings = {
  constraints: SnapConstraints.All,
  horizontalGridlines: gridlines,
  verticalGridlines: gridlines
};
// A node is created and stored in nodes array.
let node = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node} snapSettings={snapSettings}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  SnapSettingsModel,
  SnapConstraints,
  UndoRedo,
  GridlinesModel,
  Snapping,
  Inject
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let gridlines: GridlinesModel = {
  // Defines the snap interval for object
  snapIntervals: [10],
};
let snapSettings: SnapSettingsModel = {
  constraints: SnapConstraints.All,
  horizontalGridlines: gridlines,
  verticalGridlines: gridlines
}

```

```

};
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      snapSettings={snapSettings}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Snap to objects

The snap to object provides visual cues to assist with aligning and spacing diagram elements. A node can be snapped with its neighboring objects based on certain alignments. Such alignments are visually represented as smart guides.

The [snapObjectDistance](#) property allows you to define minimum distance between the selected object and the nearest object.

The [snapAngle](#) property allows you to define the snap angle by which the object needs to be rotated.

The [snapConstraints](#) property allows you to enable or disable the certain features of the snapping, refer to [snapConstraints](#).

The [snapLineColor](#) property allows you to define the color of the snapline.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { Diagram, DiagramComponent, SnapConstraints, Snapping } from
"@syncfusion/ej2-react-diagrams";
Diagram.Inject(Snapping);
let snapSettings = {
  snapObjectDistance: 10,

```

```

    snapAngle: 10,
    constraints: SnapConstraints.SnapToObject | SnapConstraints.ShowLines,
    // Set the Snapline color
    snapLineColor: 'red',
  };
  // A node is created and stored in nodes array.
  let node = [{
    // Position of the node
    offsetX: 250,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  },
  {
    // Position of the node
    offsetX: 450,
    offsetY: 250,
    // Size of the node
    width: 100,
    height: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white'
    },
    // Text(label) added to the node
  }];
  // initialize Diagram component
  function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    // Add node
    nodes={node} snapSettings={snapSettings}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  SnapSettingsModel,
  SnapConstraints,
  UndoRedo,
  GridlinesModel,
  Snapping,
  Inject
} from "@syncfusion/ej2-react-diagrams";

```

```

Diagram.Inject(Snapping);
let snapSettings: SnapSettingsModel = {
  snapObjectDistance: 10,
  snapAngle: 10,
  constraints: SnapConstraints.SnapToObject | SnapConstraints.ShowLines,
  // Set the Snapline color
  snapLineColor: 'red',
};
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  // Position of the node
  offsetX: 250,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
},
{
  // Position of the node
  offsetX: 450,
  offsetY: 250,
  // Size of the node
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // Text(label) added to the node
}];
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      // Add node
      nodes={node}
      snapSettings={snapSettings}
      // render initialized Diagram
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Page settings in React Diagram component

Page settings enable to customize the appearance, width, and height of the diagram page.

Page size and appearance

- The size and appearance of the diagram pages can be customized with the page settings property.
- The [width](#) and [height](#) properties of page settings define the size of the page and based on the size, the [orientation](#) will be set for the page. In addition to that, the appearance of the page can be customized with [source](#) and set of appearance specific properties.
- The [color](#) property is used to customize the background color and border color of the page.
- The [margin](#) property is used to define the page margin.
- To explore those properties, refer to [Page Settings](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connector = [{
  id: 'connector1',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourceID: 'node1',
  targetID: 'node2',
}];
let node = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node3'
  }]
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  getNodeDefaults=(node) => {
```

```

        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }} nodes={node} connectors={connector}
    // Defines the pageSettings for the diagram
    pageSettings={
        // Sets the PageOrientation for the diagram to page
        orientation: 'Landscape',
        // Sets the Page Break for diagram
        showPageBreaks: true,
        // Defines the background color and image of diagram
        background: {
            color: 'grey',
        },
        // Sets the width for the Page
        width: 300,
        // Sets the height for the Page
        height: 300,
        // Sets the space to be left between an annotation and its
        parent node/connector
        margin: {
            left: 10,
            top: 10,
            bottom: 10,
        },
    }
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let connector: ConnectorModel[] = [{
    id: 'connector1',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'

```



```

    },
    sourceID: 'node1',
    targetID: 'node2',
  }];
let node: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node3'
  }]
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      getNodeDefaults=(node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }
      nodes={node}
      connectors={connector}
      // Defines the pageSettings for the diagram
      pageSettings={
        // Sets the PageOrientation for the diagram to page
        orientation: 'Landscape',
        // Sets the Page Break for diagram
        showPageBreaks: true,
        // Defines the background color and image of diagram
        background: {
          color: 'grey',
        },
        // Sets the width for the Page
        width: 300,
        // Sets the height for the Page
        height: 300,
        // Sets the space to be left between an annotation and its parent
        // node/connector
        margin: {
          left: 10,

```

```

        top: 10,
        bottom: 10,
    },
    },
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Set background image

Stretch and align the background image anywhere over the diagram area. The [source](#) property of [background](#) allows you to set the path of the image. The [scale](#) and the [align](#) properties help to stretch/align the background images.

The following code illustrates how to stretch and align the background image.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
getNodeDefaults={(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
}} nodes={node} connectors={connector}
// Defines the pageSettings for the diagram
pageSettings={{
    orientation: 'Landscape',
    showPageBreaks: true,
    // Defines the background Image source
    background: {
        source:
'https://www.w3schools.com/images/w3schools_green.jpg',
        // Defines the scale values for the background image
        scale: 'Meet',
        // Defines the align values for the background image
        align: 'XMinYMin'
    },
    width: 300,
    height: 300,
    margin: {
        left: 10,
        top: 10,
        bottom: 10
    },
    },
    }>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```

```
root.render(<App />);
{% enddraw %}
```

INDEX.TSX

```
{% raw %}

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      getNodeDefaults=(node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }
    >
      // Defines the pageSettings for the diagram
      pageSettings = {
        {
          orientation: 'Landscape',
          showPageBreaks: true,
          // Defines the background Image source
          background: {
            source:
'https://www.w3schools.com/images/w3schools\_green.jpg',
            // Defines the scale values for the background image
            scale: 'Meet',
            // Defines the align values for the background image
            align: 'XMinYMin'
          },
          width: 300,
          height: 300,
          margin: {
            left: 10,
            top: 10,
            bottom: 10
          },
        },
      },
    </DiagramComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Multiple page and page breaks

When multiple page is enabled, the size of the page dynamically increases or decreases in multiples of page width and height and completely fits diagram within the page boundaries. Page breaks is used as a visual guide to see how pages are split into multiple pages.

The [multiplePage](#) and [showPageBreak](#) properties of page settings allow you to enable/disable multiple pages and page breaks respectively. The following code illustrates how to enable multiple page and page break lines.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connector = [{
  id: 'connector1',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourceID: 'node1',
  targetID: 'node2',
}];
let node = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Node1'
  }]
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 350,
  annotations: [{
    content: 'Node2'
  }]
}
];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  getNodeDefaults=(node) => {
    node.height = 100;
```

```

        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }} nodes={node} connectors={connector}
    // Defines the pageSettings for the diagram
    pageSettings={{
        orientation: 'Landscape',
        // Sets the Multiple page for diagram
        multiplePage: true,
        // Sets the Page Break for diagram
        showPageBreaks: true,
        width: 300,
        height: 300,
        margin: {
            left: 10,
            top: 10,
            bottom: 10
        },
    }},
    </>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let connector: ConnectorModel[] = [{
    id: 'connector1',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourceID: 'node1',
    targetID: 'node2',
}];
let node: NodeModel[] = [{
    id: 'node1',
    width: 100,

```

```

        height: 100,
        offsetX: 100,
        offsetY: 100,
        annotations: [{
            content: 'Node1 '
        }]
    },
    {
        id: 'node2',
        width: 100,
        height: 100,
        offsetX: 300,
        offsetY: 350,
        annotations: [{
            content: 'Node2 '
        }]
    }
];
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            getNodeDefaults={(node) => {
                node.height = 100;
                node.width = 100;
                node.style.fill = '#6BA5D7';
                node.style.strokeColor = 'white';
                return node;
            }}
            nodes={node}
            connectors={connector}
            // Defines the pageSettings for the diagram
            pageSettings = {
                {
                    orientation: 'Landscape',
                    // Sets the Multiple page for diagram
                    multiplePage: true,
                    // Sets the Page Break for diagram
                    showPageBreaks: true,
                    width: 300,
                    height: 300,
                    margin: {
                        left: 10,
                        top: 10,
                        bottom: 10
                    },
                },
            }
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Boundary constraints

The diagram provides support to restrict/customize the interactive region, out of which the elements cannot be dragged, resized, or rotated. The [boundaryConstraints](#) property of page settings allows you to customize the interactive region. To explore the boundary constraints, refer to [Boundary Constraints](#).

The following code example illustrates how to define boundary constraints with respect to the page.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connector = [{
  id: 'connector1',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 100
  }
}
]];
let node = [{
  id: 'node1',
  width: 150,
  height: 100,
  offsetX: 100,
  offsetY: 100,
}, {
  id: 'node2',
  width: 80,
  height: 130,
  offsetX: 200,
  offsetY: 200,
}
]];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
  getNodeDefaults=(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
```

```

        return node;
    }} nodes={node} connectors={connector}
    // Defines the pageSettings for the diagram
    pageSettings={{
        // Sets the BoundaryConstraints to page
        boundaryConstraints: 'Page',
        background: {
            color: 'grey'
        },
        width: 400,
        height: 400,
        showPageBreaks: true,
    }}/>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    ConnectorModel,
    NodeModel,
    BoundaryConstraints
} from "@syncfusion/ej2-react-diagrams";
let connector: ConnectorModel[] = [{
    id: 'connector1',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    },
    sourcePoint: {
        x: 300,
        y: 100
    },
    targetPoint: {
        x: 400,
        y: 100
    }
}
]];
let node: NodeModel[] = [{
    id: 'node1',
    width: 150,

```



```

    height: 100,
    offsetX: 100,
    offsetY: 100,
  }, {
    id: 'node2',
    width: 80,
    height: 130,
    offsetX: 200,
    offsetY: 200,
  }
  ]];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      getNodeDefaults={(node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
      nodes={node}
      connectors={connector}
      // Defines the pageSettings for the diagram
      pageSettings = {
        {
          // Sets the BoundaryConstraints to page
          boundaryConstraints: 'Page',
          background: {
            color: 'grey'
          },
          width: 400,
          height: 400,
          showPageBreaks: true,
        }
      }
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Scroll settings in React Diagram component

The diagram can be scrolled by using the vertical and horizontal scrollbars. In addition to the scrollbars, mousewheel can be used to scroll the diagram.

Diagram's [scrollSettings](#) enable you to read the current scroll status, view port size, current zoom, and zoom factor. It also allows you to scroll the diagram programmatically.

Get current scroll status

Scroll settings allow you to read the scroll status, [viewPortWidth](#), [viewPortHeight](#), and [currentZoom](#) with a set of properties. To explore those properties, see [Scroll Settings](#).

Define scroll status

Diagram allows you to pan the diagram before loading, so that any desired region of a large diagram is made to view. You can programmatically pan the diagram with the [horizontalOffset](#) and [verticalOffset](#) properties of scroll settings. The following code illustrates how to set pan the diagram programmatically.

In the following example, the vertical scroll bar is scrolled down by 50px and horizontal scroll bar is scrolled to right by 100px.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
//Sets scroll status
let diagramInstance;
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={700} height={700} scrollSettings={{
        horizontalOffset: 100,
        verticalOffset: 50,
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    BasicShapeModel,
    NodeModel,
    DiagramTools,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
//Sets scroll status
let diagramInstance: DiagramComponent;
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={700}
            height={700}
            scrollSettings={{
                horizontalOffset: 100,
                verticalOffset: 50,
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
```

```
root.render(<App />);
{% endraw %}
```

Update scroll status

You can programmatically change the scroll offsets at runtime by using the client-side method update. The following code illustrates how to change the scroll offsets and zoom factor at runtime.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={700} height={700}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
//Updates scroll settings
diagramInstance.scrollSettings.horizontalOffset = 200;
diagramInstance.scrollSettings.verticalOffset = 30;
diagramInstance.dataBind();
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    BasicShapeModel,
    NodeModel,
    DiagramTools,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={700}
            height={700}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
//Updates scroll settings
diagramInstance.scrollSettings.horizontalOffset = 200;
diagramInstance.scrollSettings.verticalOffset = 30;
diagramInstance.dataBind();
```

AutoScroll

Autoscroll feature automatically scrolls the diagram, whenever the node or connector is moved beyond the boundary of the diagram. So that, it is always visible during dragging, resizing, and multiple selection operations. Autoscroll is automatically triggered when any one of the following is done towards the edges of the diagram.

- Node dragging, resizing
- Connection editing
- Rubber band selection
- Label dragging

The diagram client-side event [ScrollChange](#) gets triggered when the autoscroll (scrollbars) is changed and you can do your own customization in this event.

The autoscroll behavior in your diagram can be enabled/disabled by using the [canAutoScroll](#) property of the diagram. The following code example illustrates how to set autoscroll.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{ content: 'Start' }]
}];
let connectors = [{
  // Name of the connector
  id: 'connector1', sourcePoint: { x: 300, y: 100 }, targetPoint: { x:
500, y: 200 },
  style: {
    strokeColor: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
}];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
connectors={connectors} scrollSettings={{
  canAutoScroll: true,
  //Sets the scroll limit
  scrollLimit: 'Infinity',
}} getNodeDefaults={(node) => {
```

```

        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    ConnectorModel,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    annotations: [{ content: 'Start' }]
}];
let connectors: ConnectorModel[] = [{
    // Name of the connector
    id: 'connector1', sourcePoint: { x: 300, y: 100 }, targetPoint: { x:
500, y: 200 },
    style: {
        strokeColor: '#6BA5D7',
        strokeWidth: 2
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    }
}];
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'100%'}
            height={'600px'}
            nodes={nodes}
            connectors={connectors}
            scrollSettings={{

```

```

        canAutoScroll: true,
        //Sets the scroll limit
        scrollLimit: 'Infinity',
    }}
    getNodeDefaults=(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }}
    />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Autoscroll border

The autoscroll border is used to specify the maximum distance between the object and diagram edge to trigger autoscroll. The default value is set as 15 for all sides (left, right, top, and bottom) and it can be changed by using the [autoScrollBorder](#) property of page settings. The following code example illustrates how to set autoscroll border.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'label1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
    // set the autoScrollBorder
    scrollSettings={{
        autoScrollBorder: {
            left: 100,
            right: 100,
            top: 100,
            bottom: 100

```

```

    }
    }} getNodeDefaults=(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }/>;
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  ConnectorModel,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      // set the autoScrollBorder
      scrollSettings = {
        {
          autoScrollBorder: {
            left: 100,
            right: 100,
            top: 100,
            bottom: 100

```

```

    }
  }
}
getNodeDefaults=(node) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Scroll limit

The scroll limit allows you to define the scrollable region of the diagram. It includes the following options:

- Allows to scroll in all directions without any restriction.
- Allows to scroll within the diagram content.
- Allows to scroll within the specified scrollable area.
- The [scrollLimit](#) property of scroll settings helps to limit the scrolling.

The scrollSettings [scrollableArea](#) allow to extend the scrollable region that is based on the scroll limit.

The following code example illustrates how to specify the scroll limit.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {

```



```

    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
scrollSettings={{
    //Sets the scroll limit
    scrollLimit: 'Infinity',
  }} getNodeDefaults={(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  ConnectorModel,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'labell1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      scrollSettings={{
        //Sets the scroll limit
        scrollLimit: 'Infinity',
      }}
    />
  );
}

```

```

    getNodeDefaults=(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }
  }
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Scroll Padding

The [padding](#) property of scroll settings allows you to extend the scrollable region that is based on the scroll limit.

The following code example illustrates how to set scroll padding to diagram region.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: 'Start',
  width: 100, height: 100,
  offsetX: 350, offsetY: 350,
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
    (diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
    scrollSettings={{
      //Sets the scroll padding
      padding: { right: 50, bottom: 50 }
    }} getNodeDefaults=(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    } />);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  ConnectorModel,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 100, height: 100,
  offsetX: 350, offsetY: 350,
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      scrollSettings={{
        //Sets the scroll padding
        padding: { right: 50, bottom: 50 }
      }}
      getNodeDefaults={(node) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Scrollable Area

Scrolling beyond any particular rectangular area can be restricted by using the [scrollableArea](#) property of scroll settings. To restrict scrolling beyond any custom region, set the [scrollLimit](#) as "limited". The following code example illustrates how to customize scrollable area.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Rect, DiagramComponent } from "@syncfusion/ej2-react-diagrams";
```

```

let diagramInstance;
let nodes = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
scrollSettings={{
  //Sets the scroll limit
  scrollLimit: 'Infinity',
  //Sets the scrollable Area
  scrollableArea: new Rect(0, 0, 500, 500),
}} getNodeDefaults={(node) => {
  node.height = 100;
  node.width = 100;
  node.style.fill = '#6BA5D7';
  node.style.strokeColor = 'white';
  return node;
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  ConnectorModel,
  Rect,
  DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{

```

```

        id: 'label1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
function App() {
    return (
        <DiagramComponent
            id="container"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'100%'}
            height={'600px'}
            nodes={nodes}
            scrollSettings={{
                //Sets the scroll limit
                scrollLimit: 'Infinity',
                //Sets the scrollable Area
                scrollableArea : new Rect(0, 0, 500, 500),
            }}
            getNodeDefaults={(node) => {
                node.height = 100;
                node.width = 100;
                node.style.fill = '#6BA5D7';
                node.style.strokeColor = 'white';
                return node;
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

UpdateViewport

The [updateViewPort](#) method is used to update the diagram page and view size at runtime.

Data binding in React Diagram component

- Diagram can be populated with the **nodes** and **connectors** based on the information provided from an external data source.
- Diagram exposes its specific data-related properties allowing you to specify the data source fields from where the node information has to be retrieved from.
- The [dataManager](#) property is used to define the data source either as a collection of objects or as an instance of **DataManager** that needs to be populated in the diagram.
- The [ID](#) property is used to define the unique field of each JSON data.
- The [parentId](#) property is used to defines the parent field which builds the relationship between ID and parent field.
- The [root](#) property is used to define the root node for the diagram populated from the data source.
- To explore those properties, see [DataSourceSettings](#).

- Diagram supports two types of data binding. They are:
 1. Local data
 2. Remote data

Local data

Diagram can be populated based on the user defined JSON data (Local Data) by mapping the relevant data source fields.

To map the user defined JSON data with diagram, configure the fields of [dataSourceSettings](#). The following code example illustrates how to bind local data with the diagram.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DataBinding, HierarchicalTree, DiagramTools,
Inject } from "@syncfusion/ej2-react-diagrams";
import { DataManager } from '@syncfusion/ej2-data';
let species = [
  { 'Name': 'Species', 'fillColor': '#3DD94A' },
  { 'Name': 'Plants', 'Category': 'Species' },
  { 'Name': 'Fungi', 'Category': 'Species' },
  { 'Name': 'Lichens', 'Category': 'Species' },
  { 'Name': 'Animals', 'Category': 'Species' },
  { 'Name': 'Mosses', 'Category': 'Plants' },
  { 'Name': 'Ferns', 'Category': 'Plants' },
  { 'Name': 'Gymnosperms', 'Category': 'Plants' },
  { 'Name': 'Dicotyledens', 'Category': 'Plants' },
  { 'Name': 'Monocotyledens', 'Category': 'Plants' },
  { 'Name': 'Invertebrates', 'Category': 'Animals' },
  { 'Name': 'Vertebrates', 'Category': 'Animals' },
  { 'Name': 'Insects', 'Category': 'Invertebrates' },
  { 'Name': 'Molluscs', 'Category': 'Invertebrates' },
  { 'Name': 'Crustaceans', 'Category': 'Invertebrates' },
  { 'Name': 'Others', 'Category': 'Invertebrates' },
  { 'Name': 'Fish', 'Category': 'Vertebrates' },
  { 'Name': 'Amphibians', 'Category': 'Vertebrates' },
  { 'Name': 'Reptiles', 'Category': 'Vertebrates' },
  { 'Name': 'Birds', 'Category': 'Vertebrates' },
  { 'Name': 'Mammals', 'Category': 'Vertebrates' }
];
//Initializes diagram control
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={490}
    //Configures data source
    dataSourceSettings={{
      id: 'Name',
      parentId: 'Category',
      dataManager: new DataManager(species),
      //binds the external data with node
      doBinding: (nodeModel, data, diagram) => {
        nodeModel.annotations = [
          {
            /* tslint:disable:no-string-literal */
            content: data['Name'],
          }
        ]
      }
    }}
  />);
}
```

```

        margin: {
            top: 10,
            left: 10,
            right: 10,
            bottom: 0,
        },
        style: {
            color: 'black',
        },
    },
];
/* tslint:disable:no-string-literal */
nodeModel.style = {
    fill: '#ffeec7',
    strokeColor: '#f5d897',
    strokeWidth: 1,
};
    },
    })
//Configures HierarchicalTree layout
layout={{
    type: 'HierarchicalTree',
    horizontalSpacing: 15,
    verticalSpacing: 50,
    margin: {
        top: 10,
        left: 10,
        right: 10,
        bottom: 0,
    },
}}
//Sets the default values of nodes
getNodeDefaults=(obj, diagram) => {
    //Initialize shape
    obj.shape = {
        type: 'Basic',
        shape: 'Rectangle',
    };
    obj.style = {
        strokeWidth: 1,
    };
    obj.width = 95;
    obj.height = 30;
}
//Sets the default values of connectors
getConnectorDefaults=(connector, diagram) => {
    connector.type = 'Orthogonal';
    connector.style.strokeColor = '#4d4d4d';
    connector.targetDecorator.shape = 'None';
}
//Disables all interactions except zoom/pan
tool={DiagramTools.ZoomPan} snapSettings={{
    constraints: 0,
}}>
    <Inject services={[DataBinding, HierarchicalTree]}/>
</DiagramComponent>;
}

```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    ConnectorModel,
    Node,
    DataBinding,
    HierarchicalTree,
    DiagramTools,
    Inject
} from "@syncfusion/ej2-react-diagrams";
import {
    DataManager
} from '@syncfusion/ej2-data';
let species: object[] = [
    { 'Name': 'Species', 'fillColor': '#3DD94A' },
    { 'Name': 'Plants', 'Category': 'Species' },
    { 'Name': 'Fungi', 'Category': 'Species' },
    { 'Name': 'Lichens', 'Category': 'Species' },
    { 'Name': 'Animals', 'Category': 'Species' },
    { 'Name': 'Mosses', 'Category': 'Plants' },
    { 'Name': 'Ferns', 'Category': 'Plants' },
    { 'Name': 'Gymnosperms', 'Category': 'Plants' },
    { 'Name': 'Dicotyledens', 'Category': 'Plants' },
    { 'Name': 'Monocotyledens', 'Category': 'Plants' },
    { 'Name': 'Invertebrates', 'Category': 'Animals' },
    { 'Name': 'Vertebrates', 'Category': 'Animals' },
    { 'Name': 'Insects', 'Category': 'Invertebrates' },
    { 'Name': 'Molluscs', 'Category': 'Invertebrates' },
    { 'Name': 'Crustaceans', 'Category': 'Invertebrates' },
    { 'Name': 'Others', 'Category': 'Invertebrates' },
    { 'Name': 'Fish', 'Category': 'Vertebrates' },
    { 'Name': 'Amphibians', 'Category': 'Vertebrates' },
    { 'Name': 'Reptiles', 'Category': 'Vertebrates' },
    { 'Name': 'Birds', 'Category': 'Vertebrates' },
    { 'Name': 'Mammals', 'Category': 'Vertebrates' }
];
//Initializes diagram control
function App() {
    return (
        <DiagramComponent
            id="container"
            width={ '100%' }
            height={ 490 }
            //Configures data source
            dataSourceSettings={{
                id: 'Name',
```



```

    parentId: 'Category',
    dataManager: new DataManager(species),
    //binds the external data with node
    doBinding: (nodeModel: NodeModel, data: DataInfo, diagram: Diagram)
=> {
    nodeModel.annotations = [
      {
        /* tslint:disable:no-string-literal */
        content: data['Name'],
        margin: {
          top: 10,
          left: 10,
          right: 10,
          bottom: 0,
        },
        style: {
          color: 'black',
        },
      },
    ];
    /* tslint:disable:no-string-literal */
    nodeModel.style = {
      fill: '#ffeec7',
      strokeColor: '#f5d897',
      strokeWidth: 1,
    };
  },
}
//Configures HierarchicalTree layout
layout={{
  type: 'HierarchicalTree',
  horizontalSpacing: 15,
  verticalSpacing: 50,
  margin: {
    top: 10,
    left: 10,
    right: 10,
    bottom: 0,
  },
}}
//Sets the default values of nodes
getNodeDefaults={(obj: Node, diagram: Diagram) => {
  //Initialize shape
  obj.shape = {
    type: 'Basic',
    shape: 'Rectangle',
  };
  obj.style = {
    strokeWidth: 1,
  };
  obj.width = 95;
  obj.height = 30;
}}
//Sets the default values of connectors
getConnectorDefaults={(connector: ConnectorModel, diagram: Diagram) => {
  {
    connector.type = 'Orthogonal';
  }
}

```

```

        connector.style.strokeColor = '#4d4d4d';
        connector.targetDecorator.shape = 'None';
    }}
    //Disables all interactions except zoom/pan
    tool={DiagramTools.ZoomPan}
    snapSettings={{
        constraints: 0,
    }}
    >
    <Inject services={[DataBinding, HierarchicalTree]} />
    </DiagramComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
export interface EmployeeInfo {
    Role: string;
    color: string;
}
export interface DataInfo {
    [key: string]: string;
}
{% endraw %}

```

Remote data

You can bind the diagram with remote data by using `[dataManager]`.

It uses two different classes: `DataManager` for processing and `Query` for serving data. `DataManager` communicates with data source and `Query` generates data queries that are read by the [dataManager](#).

To learn more about data manager, refer to [Data Manager](#).

To bind remote data to the diagram, configure the fields of [dataSourceSettings](#). The following code illustrates how to bind remote data to the diagram.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DataBinding, HierarchicalTree, DiagramTools,
Inject } from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from '@syncfusion/ej2-data';
//Initializes diagram control
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={490}
    //Configures hierarchical tree layout
    layout={{
        type: 'HierarchicalTree',
        margin: {
            left: 0,
            right: 0,
            top: 100,
            bottom: 0,
        },
        verticalSpacing: 40,
    }}
    >
    </DiagramComponent>
    );
}
{% endraw %}

```

```

        getLayoutInfo: (node, options) => {
            if (options.level === 3) {
                node.style.fill = '#3c418d';
            }
            if (options.level === 2) {
                node.style.fill = '#108d8d';
                options.type = 'Center';
                options.orientation = 'Horizontal';
            }
            if (options.level === 1) {
                node.style.fill = '#822b86';
            }
        },
    }
}

//Sets the default values of nodes
getNodeDefaults=(obj) => {
    obj.width = 80;
    obj.height = 40;
    //Initialize shape
    obj.shape = {
        type: 'Basic',
        shape: 'Rectangle',
    };
    obj.style = {
        fill: '#048785',
        strokeColor: 'Transparent',
    };
}

//Sets the default values of connector
getConnectorDefaults=(connector) => {
    connector.type = 'Orthogonal';
    connector.style.strokeColor = '#048785';
    connector.targetDecorator.shape = 'None';
}

//Configures data source
dataSourceSettings={{
    id: "Id",
    parentId: "ParentId",
    dataSource: new DataManager({
        url:
            "https://services.syncfusion.com/react/production/api/RemoteData",
        crossDomain: true
    }),
    //binds the external data with node
    doBinding: (nodeModel, data, diagram) => {
        nodeModel.annotations = [
            {
                /* tslint:disable:no-string-literal */
                content: data["Label"],
                style: { color: "white" }
            }
        ];
    }
}

//Disables all interactions except zoom/pan
tool={DiagramTools.ZoomPan} snapSettings={{
    constraints: 0,

```

```

    >>
    <Inject services={[DataBinding, HierarchicalTree]}/>
  </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  Node,
  Connector,
  DataBinding,
  HierarchicalTree,
  TreeInfo,
  DiagramTools,
  Inject,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
import {
  DataManager,
  Query
} from '@syncfusion/ej2-data';
//Initializes diagram control
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={490}
      //Configures hierarchical tree layout
      layout={{
        type: 'HierarchicalTree',
        margin: {
          left: 0,
          right: 0,
          top: 100,
          bottom: 0,
        },
        verticalSpacing: 40,
        getLayoutInfo: (node: Node, options: TreeInfo) => {
          if (options.level === 3) {
            node.style.fill = '#3c418d';
          }
          if (options.level === 2) {
            node.style.fill = '#108d8d';
            options.type = 'Center';
            options.orientation = 'Horizontal';
          }
        }
      }}
    >

```

```

        if (options.level === 1) {
            node.style.fill = '#822b86';
        }
    },
    })
    //Sets the default values of nodes
    getNodeDefaults=(obj: NodeModel) => {
        obj.width = 80;
        obj.height = 40;
        //Initialize shape
        obj.shape = {
            type: 'Basic',
            shape: 'Rectangle',
        };
        obj.style = {
            fill: '#048785',
            strokeColor: 'Transparent',
        };
    }
    //Sets the default values of connector
    getConnectorDefaults=(connector: ConnectorModel) => {
        connector.type = 'Orthogonal';
        connector.style.strokeColor = '#048785';
        connector.targetDecorator.shape = 'None';
    }
    //Configures data source
    dataSourceSettings={{
        id: "Id",
        parentId: "ParentId",
        dataSource: new DataManager({
            url:
                "https://services.syncfusion.com/react/production/api/RemoteData",
            crossDomain: true
        }),
        //binds the external data with node
        doBinding: (nodeModel: NodeModel, data: DataInfo, diagram: Diagram)
    => {
        nodeModel.annotations = [
            {
                /* tslint:disable:no-string-literal */
                content: data["Label"],
                style: { color: "white" }
            }
        ];
    }
    })
    //Disables all interactions except zoom/pan
    tool={DiagramTools.ZoomPan}
    snapSettings={{
        constraints: 0,
    }}
    >
    <Inject services={[DataBinding, HierarchicalTree]} />
    </DiagramComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));

```

```
root.render(<App />);
export interface DataInfo {
  [key: string]: string;
}
{% endraw %}
```

CRUD

This feature allows you to read the data source and perform add or edit or delete the data in data source at runtime.

Read DataSource

- This feature allows you to define the nodes and connectors collection in the data source and connectionDataSource respectively.
- You can set the data collection in the model's dataSourceSettings [dataManager](#) property. The nodes will be generated based on the data specified in the data source.
- You can set the connector collection in the model's dataSourceSettings [connectionDataSource](#) property.
- The dataSourceSettings connectionDataSource [dataManager](#) property is used to set the data source for the connection data source items.
- If you have a data (data will be set in the dataSource property) with parent relationship in the database and also defined the connector in the connectionDataSource simultaneously, then the connectors set in the connectionDataSource will be considered as a priority to render the connector.
- The dataSourceSettings [crudAction's read](#) property specifies the method, which is used to read the data source and its populate the nodes in the diagram.
- The connectionDataSource crudAction's [read](#) specifies the method, which is used to read the data source and its populates the connectors in the diagram.
- The dataSourceSettings's [id](#) and connectionDataSource's [id](#) properties are used to define the unique field of each JSON data.
- The connectionDataSource's [sourceID](#) and [targetID](#) properties are used to set the sourceID and targetID for connection data source item.
- The connectionDataSource's [sourcePointX](#), [sourcePointY](#), [targetPointX](#), and [targetPointY](#) properties are used to define the sourcePoint and targetPoint values for connector from data source.
- The dataSourceSettings crudAction's [customFields](#) property is used to maintain the additional information for nodes.
- Similarly, connectionDataSource's crudAction's [customFields](#) is used to maintain the additional information for connectors.

How to perform Editing at runtime

- The dataSourceSettings crudAction object allows you to define the method, which is used to get the changes done in the data source defined for shapes from the client-side to the server-side.
- Similarly, the connectionDataSource crudAction object allows you to define the method, which is used to get the changes done in the data source defined for connectors from the client-side to the server-side.

InsertData

- The dataSourceSettings crudAction's [create](#) property specifies the method, which is used to get the nodes added from the client-side to the server-side.
- The connectionDataSource crudAction's [create](#) specifies the method, which is used to get the connectors added from the client-side to the server-side.
- The following code example illustrates how to send the newly added or inserted data from the client to server-side.

```
{% raw %}  
`ts  
import * as React from "react";  
import * as ReactDOM from "react-dom";  
import {  
  Diagram,  
  DiagramComponent,  
  Inject,  
  DataBinding  
} from "@syncfusion/ej2-react-diagrams";  
let diagramInstance: DiagramComponent;  
function App() {  
  return (  
    <DiagramComponent  
      id="container"  
      // diagram instance for an diagram  
      ref={{(diagram) => (diagramInstance = diagram)}}  
      width={'100%'}  
      height={'550px'}  
      //Configures data source for diagram  
      dataSourceSettings={{  
        crudAction: {  
          //Url which triggers the server side AddNodes method  
          create:  
            'https://ej2services.syncfusion.com/development/web-services/api/Crud/AddNodes',  
        },  
        connectionDataSource: {
```

```

crudAction: {
  //Url which triggers the server side AddConnectors method
  create:
    'https://ej2services.syncfusion.com/development/web-services/api/Crud/AddConnectors',
  },
},
}}

<Inject services={[DataBinding]} />
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

//Sends the inserted nodes/connectors from client side to the server side through the URL which is
specified in server side.
diagramInstance.insertData();
`

{% endraw %}

```

UpdateData

- The dataSourceSettings crudAction's [update](#) property specifies the method, which is used to get the modified nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [update](#) specifies the method, which is used to get the modified connectors from the client-side to the server-side.
- The following code example illustrates how to send the updated data from the client to the server side.

```

{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,

```


DataBinding

```
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      // diagram instance for an diagram
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'550px'}
      //Configures data source for diagram
      dataSourceSettings={{
        crudAction: {
          //Url which triggers the server side UpdateNodes method
          update:
            'https://ej2services.syncfusion.com/development/web-services/api/Crud/UpdateNodes',
        },
        connectionDataSource: {
          crudAction: {
            //Url which triggers the server side UpdateConnectors method
            update:
              'https://ej2services.syncfusion.com/development/web-services/api/Crud/UpdateConnectors',
          },
        },
      }}

    <Inject services={[DataBinding]} />
    </DiagramComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

//Sends the updated nodes/connectors from client side to the server side through the URL which is specified in server side.

```
diagramInstance.updateData();
```

```
,
```

```
{% endraw %}
```

DeleteData

- The dataSourceSettings crudAction's [destroy](#) property specifies the method, which is used to get the deleted nodes from the client-side to the server-side.
- The connectionDataSource crudAction's [destroy](#) specifies the method, which is used to get the deleted connectors from the client-side to the server-side.

```
{% raw %}
```

```
`ts
```

```
import * as React from "react";
```

```
import * as ReactDOM from "react-dom";
```

```
import {
```

```
Diagram,
```

```
DiagramComponent,
```

```
Inject,
```

```
DataBinding
```

```
} from "@syncfusion/ej2-react-diagrams";
```

```
let diagramInstance: DiagramComponent;
```

```
function App() {
```

```
return (
```

```
<DiagramComponent
```

```
id="container"
```

```
// diagram instance for an diagram
```

```
ref={({diagram}) => (diagramInstance = diagram)}
```

```
width={'100%'}
```

```
height={'550px'}
```

```
//Configures data source for diagram
```

```
dataSourceSettings={{
```

```
crudAction: {
```

```
//Url which triggers the server side DeleteNodes method
```

```
destroy:
```

```

'https://ej2services.syncfusion.com/development/web-services/api/Crud/DeleteNodes',
},
connectionDataSource: {
  crudAction: {
    //Url which triggers the server side DeleteConnectors method
    destroy:
      'https://ej2services.syncfusion.com/development/web-services/api/Crud/DeleteConnectors',
  },
},
}}

<Inject services={[DataBinding]} />
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

//Sends the deleted nodes/connectors from client side to the server side through the URL which is
specified in server side.
diagramInstance.removeData();
`

{% enddraw %}

```

See Also

- [How to arrange the diagram nodes and connectors using varies layout](#)

Automatic layout in React Diagram component

Diagram provides support to auto-arrange the nodes in the diagram area that is referred as **Layout**. It includes the following layout modes:

Layout modes

- Hierarchical layout
- Organization chart
- Radial tree
- Symmetric layout
- Mind Map layout
- Complex hierarchical tree layout

Hierarchical layout

The hierarchical tree layout arranges nodes in a tree-like structure, where the nodes in the hierarchical layout may have multiple parents. There is no need to specify the layout root. To arrange the nodes in a hierarchical structure, specify the layout [type](#) as hierarchical tree. The following example shows how to arrange the nodes in a hierarchical structure.

Note: If you want to use hierarchical tree layout in diagram, you need to inject HierarchicalTree in the diagram.

The following example shows how to arrange the nodes in a hierarchical structure.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
];
let items = new DataManager(data, new Query().take(7));
function App() {
  return (<div className="control-pane">
    <div className="control-section">
      <div className="content-wrapper" style={{ width: "100%" }}>
        <DiagramComponent id="container" width={"100%"} height={"550px"}
//Uses layout to auto-arrange nodes on the diagram page
        layout={{
```

```

        //Sets layout type
        type: "HierarchicalTree",
    }}
    //Configures data source for diagram
    dataSourceSettings={{
        id: "Name",
        parentId: "ReportingPerson",
        dataManager: items,
    }}
    //Sets the default properties for nodes
    getNodeDefaults=(obj) => {
        obj.shape = {
            type: "Text",
            content: obj.data.Name,
        };
        obj.style = {
            fill: "None",
            strokeColor: "none",
            strokeWidth: 2,
            bold: true,
            color: "white",
        };
        obj.borderColor = "white";
        obj.width = 100;
        obj.height = 40;
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        obj.shape.margin = {
            left: 5,
            right: 5,
            top: 5,
            bottom: 5,
        };
        return obj;
    }
    //Sets the default properties for and connectors
    getConnectorDefaults=(connector, diagram) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.type = "Orthogonal";
        return connector;
    }
    <Inject services={[DataBinding, HierarchicalTree]}/>
    </DiagramComponent>
</div>
</div>
</div>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  DiagramConstraints,
  HierarchicalTree,
  TextModel,
  DataBinding,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
],
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <div className="control-pane">
      <div className="control-section">
        <div className="content-wrapper" style={{ width: "100%" }}>
          <DiagramComponent
            id="container"
            width={"100%"}
          />
        </div>
      </div>
    </div>
  );
}
```

```

height={"550px"}
//Uses layout to auto-arrange nodes on the diagram page
layout={{
  //Sets layout type
  type: "HierarchicalTree",
}}
//Configures data source for diagram
dataSourceSettings={{
  id: "Name",
  parentId: "ReportingPerson",
  dataManager: items,
}}
//Sets the default properties for nodes
getNodeDefaults=(obj) => {
  obj.shape = {
    type: "Text",
    content: obj.data.Name,
  };
  obj.style = {
    fill: "None",
    strokeColor: "none",
    strokeWidth: 2,
    bold: true,
    color: "white",
  };
  obj.borderColor = "white";
  obj.width = 100;
  obj.height = 40;
  obj.backgroundColor = "#6BA5D7";
  obj.borderWidth = 1;
  obj.shape.margin = {
    left: 5,
    right: 5,
    top: 5,
    bottom: 5,
  };
  return obj;
}
//Sets the default properties for and connectors
getConnectorDefaults=(connector, diagram) => {
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.type = "Orthogonal";
  return connector;
}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
</div>
</div>
</div>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}
```

Radial tree layout

The radial tree layout arranges nodes on a virtual concentric circle around a root node. Sub-trees formed by the branching of child nodes are located radially around the child nodes. This arrangement result in an ever-expanding concentric arrangement with radial proximity to the root node indicating the node level in the hierarchy. The layout [root](#) property can be used to define the root node of the layout. When no root node is set, the algorithm automatically considers one of the diagram nodes as the root node.

To arrange nodes in a radial tree structure, set the [type](#) of the layout as **RadialTree**. The following code illustrates how to arrange the nodes in a radial tree structure.

Note: If you want to use radial tree layout in diagram, you need to inject DataBinding and RadialTree in the diagram.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, RadialTree, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Id: 1,
    Name: "Ana Trujillo",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
  },
  {
    Id: 2,
    Name: "Lino Rodri",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 3,
    Name: "Philip Cramer",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 4,
    Name: "Pedro Afonso",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
],
```



```
{
  Id: 5,
  Name: "Anto Moreno",
  Designation: "Project Lead",
  RatingColor: "#93B85A",
  ReportingPerson: 1,
},
{
  Id: 6,
  Name: "Elizabeth Roel",
  Designation: "Project Lead",
  RatingColor: "#93B85A",
  ReportingPerson: 1,
},
{
  Id: 7,
  Name: "Aria Cruz",
  Designation: "Project Lead",
  RatingColor: "#93B85A",
  ReportingPerson: 1,
},
{
  Id: 8,
  Name: "Eduardo Roel",
  Designation: "Project Lead",
  RatingColor: "#93B85A",
  ReportingPerson: 1,
},
{
  Id: 9,
  Name: "Howard Snyder",
  Designation: "Project Lead",
  RatingColor: "#68C2DE",
  ReportingPerson: 1,
},
{
  Id: 10,
  Name: "Daniel Tonini",
  Designation: "Project Lead",
  RatingColor: "#93B85A",
  ReportingPerson: 1,
},
{
  Id: 11,
  Name: "Nardo Batista",
  Designation: "Project Lead",
  RatingColor: "#68C2DE",
  ReportingPerson: 1,
},
},
];
let items = new DataManager(data, new Query().take(5));
function App() {
  return (<DiagramComponent id="container" width={"100%"} height={"590px"}
  snapSettings={{
    constraints: 0,
  }}
  //Uses layout to auto-arrange nodes on the diagram page
```

```

    layout={{
      //set the type as Radial tree
      type: "RadialTree",
      root: "parent",
    }}
    //Configures data source for diagram
    dataSourceSettings={{
      id: "Id",
      parentId: "ReportingPerson",
      dataManager: items,
    }}
    //Sets the default properties for nodes and connectors
    getNodeDefaults=(obj, diagram) => {
      obj.height = 15;
      obj.width = 15;
      obj.borderWidth = 1;
      obj.style = {
        fill: "#6BA5D7",
        strokeWidth: 2,
        strokeColor: "#6BA5D7",
      };
      return obj;
    }
    getConnectorDefaults=(connector, diagram) => {
      connector.style = {
        strokeColor: "#6BA5D7",
        strokeWidth: 2,
      };
      connector.targetDecorator.style.fill = "#6BA5D7";
      connector.targetDecorator.style.strokeColor = "#6BA5D7";
      connector.targetDecorator.shape = "None";
      connector.type = "Straight";
      return connector;
    }
  }}
  <Inject services={[DataBinding, RadialTree]}/>
</DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  Node,
  NodeModel,
  Container,
  TextElement,
  StackPanel,
  ImageElement,

```

```
    DataBinding,
    RadialTree,
    TreeInfo,
    DiagramTools,
  } from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object = [
  {
    Id: 1,
    Name: "Ana Trujillo",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
  },
  {
    Id: 2,
    Name: "Lino Rodri",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 3,
    Name: "Philip Cramer",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 4,
    Name: "Pedro Afonso",
    Designation: "Project Manager",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 5,
    Name: "Anto Moreno",
    Designation: "Project Lead",
    RatingColor: "#93B85A",
    ReportingPerson: 1,
  },
  {
    Id: 6,
    Name: "Elizabeth Roel",
    Designation: "Project Lead",
    RatingColor: "#93B85A",
    ReportingPerson: 1,
  },
  {
    Id: 7,
    Name: "Aria Cruz",
    Designation: "Project Lead",
    RatingColor: "#93B85A",
    ReportingPerson: 1,
  },
  {

```

```

    Id: 8,
    Name: "Eduardo Roel",
    Designation: "Project Lead",
    RatingColor: "#93B85A",
    ReportingPerson: 1,
  },
  {
    Id: 9,
    Name: "Howard Snyder",
    Designation: "Project Lead",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
  {
    Id: 10,
    Name: "Daniel Tonini",
    Designation: "Project Lead",
    RatingColor: "#93B85A",
    ReportingPerson: 1,
  },
  {
    Id: 11,
    Name: "Nardo Batista",
    Designation: "Project Lead",
    RatingColor: "#68C2DE",
    ReportingPerson: 1,
  },
],
let items: DataManager = new DataManager(data as JSON[], new
Query().take(5));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"590px"}
      snapSettings={{
        constraints: 0,
      }}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //set the type as Radial tree
        type: "RadialTree",
        root: "parent",
      }}
      //Configures data source for diagram
      dataSourceSettings={{
        id: "Id",
        parentId: "ReportingPerson",
        dataManager: items,
      }}
      //Sets the default properties for nodes and connectors
      getNodeDefaults=(obj, diagram) => {
        obj.height = 15;
        obj.width = 15;
        obj.borderWidth = 1;
        obj.style = {

```

```

        fill: "#6BA5D7",
        strokeWidth: 2,
        strokeColor: "#6BA5D7",
    };
    return obj;
  }}
  getConnectorDefaults=(connector, diagram) => {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.targetDecorator.shape = "None";
    connector.type = "Straight";
    return connector;
  }}
  >
  <Inject services={[DataBinding, RadialTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Organizational Chart

An organizational chart is a diagram that displays the structure of an organization and relationships. To create an organizational chart, the [type](#) of layout should be set as an `OrganizationalChart`. The following code example illustrates how to create an organizational chart.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, HierarchicalTree, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Id: "parent",
    Role: "Project Management",
  },
  {
    Id: 1,
    Role: "R&D Team",
    Team: "parent",
  },
  {
    Id: 3,
    Role: "Philosophy",
    Team: "1",
  },
  {

```

```
    Id: 4,  
    Role: "Organization",  
    Team: "1",  
  },  
  {  
    Id: 5,  
    Role: "Technology",  
    Team: "1",  
  },  
  {  
    Id: 7,  
    Role: "Funding",  
    Team: "1",  
  },  
  {  
    Id: 8,  
    Role: "Resource Allocation",  
    Team: "1",  
  },  
  {  
    Id: 9,  
    Role: "Targeting",  
    Team: "1",  
  },  
  {  
    Id: 11,  
    Role: "Evaluation",  
    Team: "1",  
  },  
  {  
    Id: 156,  
    Role: "HR Team",  
    Team: "parent",  
  },  
  {  
    Id: 13,  
    Role: "Recruitment",  
    Team: "156",  
  },  
  {  
    Id: 113,  
    Role: "Training",  
    Team: "12",  
  },  
  {  
    Id: 112,  
    Role: "Employee Relation",  
    Team: "156",  
  },  
  {  
    Id: 14,  
    Role: "Record Keeping",  
    Team: "12",  
  },  
  {  
    Id: 15,  
    Role: "Compensations & Benefits",
```

```

        Team: "12",
      },
      {
        Id: 16,
        Role: "Compliances",
        Team: "12",
      },
      {
        Id: 17,
        Role: "Production & Sales Team",
        Team: "parent",
      },
      {
        Id: 119,
        Role: "Design",
        Team: "17",
      },
      {
        Id: 19,
        Role: "Operation",
        Team: "17",
      },
      {
        Id: 20,
        Role: "Support",
        Team: "17",
      },
      {
        Id: 21,
        Role: "Quality Assurance",
        Team: "17",
      },
      {
        Id: 23,
        Role: "Customer Interaction",
        Team: "17",
      },
      {
        Id: 24,
        Role: "Support and Maintenance",
        Team: "17",
      },
      {
        Id: 25,
        Role: "Task Coordination",
        Team: "17",
      },
    ],
    ];
    let items = new DataManager(data, new Query().take(5));
    function App() {
      return (<DiagramComponent id="container" width={"100%"} height={"550px"}
        //Uses layout to auto-arrange nodes on the diagram page
        layout={
          {
            //set the type as Organizational Chart
            type: "OrganizationalChart",
            enableAnimation: true,
            margin: { top: 20 },
          }
        }
      >);
    }
  
```

```

        horizontalSpacing: 25,
        verticalSpacing: 30,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
    }}
    //Configures data source for diagram
    dataSourceSettings={{
        id: "Id",
        parentId: "Team",
        dataManager: items,
    }}
    //Sets the default properties for nodes and connectors
    getNodeDefaults=(obj) => {
        obj.shape = {
            type: "Text",
            content: obj.data.Role,
        };
        obj.style = {
            fill: "None",
            strokeColor: "none",
            strokeWidth: 2,
            bold: true,
            color: "white",
        };
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        obj.width = 75;
        obj.height = 40;
        obj.shape.margin = {
            left: 5,
            right: 5,
            top: 5,
            bottom: 5,
        };
        return obj;
    }
    getConnectorDefaults=(connector, diagram) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.type = "Orthogonal";
        return connector;
    }
    <Inject services={[DataBinding, HierarchicalTree]}/>
    </DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
```



```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  Node,
  NodeModel,
  Container,
  TextElement,
  StackPanel,
  ImageElement,
  DataBinding,
  TextModel,
  HierarchicalTree,
  TreeInfo,
  DiagramTools,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Id: "parent",
    Role: "Project Management",
  },
  {
    Id: 1,
    Role: "R&D Team",
    Team: "parent",
  },
  {
    Id: 3,
    Role: "Philosophy",
    Team: "1",
  },
  {
    Id: 4,
    Role: "Organization",
    Team: "1",
  },
  {
    Id: 5,
    Role: "Technology",
    Team: "1",
  },
  {
    Id: 7,
    Role: "Funding",
    Team: "1",
  },
  {
    Id: 8,
    Role: "Resource Allocation",
    Team: "1",
  },
  {

```

```
    Id: 9,
    Role: "Targeting",
    Team: "1",
  },
  {
    Id: 11,
    Role: "Evaluation",
    Team: "1",
  },
  {
    Id: 156,
    Role: "HR Team",
    Team: "parent",
  },
  {
    Id: 13,
    Role: "Recruitment",
    Team: "156",
  },
  {
    Id: 113,
    Role: "Training",
    Team: "12",
  },
  {
    Id: 112,
    Role: "Employee Relation",
    Team: "156",
  },
  {
    Id: 14,
    Role: "Record Keeping",
    Team: "12",
  },
  {
    Id: 15,
    Role: "Compensations & Benefits",
    Team: "12",
  },
  {
    Id: 16,
    Role: "Compliances",
    Team: "12",
  },
  {
    Id: 17,
    Role: "Production & Sales Team",
    Team: "parent",
  },
  {
    Id: 119,
    Role: "Design",
    Team: "17",
  },
  {
    Id: 19,
    Role: "Operation",
```

```

    Team: "17",
  },
  {
    Id: 20,
    Role: "Support",
    Team: "17",
  },
  {
    Id: 21,
    Role: "Quality Assurance",
    Team: "17",
  },
  {
    Id: 23,
    Role: "Customer Interaction",
    Team: "17",
  },
  {
    Id: 24,
    Role: "Support and Maintenance",
    Team: "17",
  },
  {
    Id: 25,
    Role: "Task Coordination",
    Team: "17",
  },
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(5));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //set the type as Organizational Chart
        type: "OrganizationalChart",
        enableAnimation: true,
        margin: { top: 20 },
        horizontalSpacing: 25,
        verticalSpacing: 30,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
      }}
      //Configures data source for diagram
      dataSourceSettings={{
        id: "Id",
        parentId: "Team",
        dataManager: items,
      }}
      //Sets the default properties for nodes and connectors
      getNodeDefaults=(obj: NodeModel) => {
        obj.shape = {
          type: "Text",

```

```

        content: (
            obj.data as {
                Role: "string";
            }
        ).Role,
    };
    obj.style = {
        fill: "None",
        strokeColor: "none",
        strokeWidth: 2,
        bold: true,
        color: "white",
    };
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    obj.width = 75;
    obj.height = 40;
    (obj.shape as TextModel).margin = {
        left: 5,
        right: 5,
        top: 5,
        bottom: 5,
    };
    return obj;
}}
getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
    connector.style = {
        strokeColor: "#6BA5D7",
        strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
}}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Organizational chart layout starts parsing from root and iterate through all its child elements. The `getLayoutInfo` method provides necessary information of a node's children and the way to arrange (direction, orientation, offsets, etc.) them. The arrangements can be customized by overriding this function as explained.

GetLayoutInfo

Set chart orientations, chart types, and offset to be left between parent and child nodes by overriding the method, `diagram.layout.getLayoutInfo`. The [getLayoutInfo](#) method is called to configure every subtree of the organizational chart. It takes the following arguments.

- node: Parent node to that options are to be customized.
- options: Object to set the customizable properties.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, HierarchicalTree,
SnapConstraints, } from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Id: 1,
    Role: "General Manager",
  },
  {
    Id: 2,
    Role: "Assistant Manager",
    Team: 1,
  },
  {
    Id: 3,
    Role: "Human Resource Manager",
    Team: 1,
  },
  {
    Id: 4,
    Role: "Design Manager",
    Team: 1,
  },
  {
    Id: 5,
    Role: "Operation Manager",
    Team: 1,
  },
  {
    Id: 6,
    Role: "Marketing Manager",
    Team: 1,
  },
];
let items = new DataManager(data, new Query().take(7));
function App() {
  return (<DiagramComponent id="container" width={"100%"} height={"530px"}
snapSettings={{
  constraints: SnapConstraints.None,
}}
//Uses layout to auto-arrange nodes on the diagram page
  layout={{
```

```

        //Sets layout type
        type: "OrganizationalChart",
        getLayoutInfo: (node, options) => {
            if (!options.hasSubTree) {
                options.type = "Center";
                options.orientation = "Horizontal";
            }
        },
    },
}

//Configures data source for diagram
dataSourceSettings={{
    id: "Id",
    parentId: "Team",
    dataManager: items,
}}

//Sets the default properties for nodes and connectors
getNodeDefaults=(obj, diagram) => {
    obj.width = 150;
    obj.height = 50;
    obj.style.fill = "#6BA5D7";
    obj.annotations = [
        {
            content: obj.data["Role"],
            style: {
                color: "white",
            },
        },
    ],
};
return obj;
}} getConnectorDefaults=(connector, diagram) => {
    connector.style = {
        strokeColor: "#6BA5D7",
        strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.targetDecorator.shape = "None";
    connector.type = "Orthogonal";
    return connector;
}}>
<Inject services={[DataBinding, HierarchicalTree]}/>
</DiagramComponent>;
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,

```

```

Inject,
ConnectorModel,
Node,
NodeModel,
DataBinding,
HierarchicalTree,
SnapConstraints,
TreeInfo,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Id: 1,
    Role: "General Manager",
  },
  {
    Id: 2,
    Role: "Assistant Manager",
    Team: 1,
  },
  {
    Id: 3,
    Role: "Human Resource Manager",
    Team: 1,
  },
  {
    Id: 4,
    Role: "Design Manager",
    Team: 1,
  },
  {
    Id: 5,
    Role: "Operation Manager",
    Team: 1,
  },
  {
    Id: 6,
    Role: "Marketing Manager",
    Team: 1,
  },
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"530px"}
      snapSettings={{
        constraints: SnapConstraints.None,
      }}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type
        type: "OrganizationalChart",

```

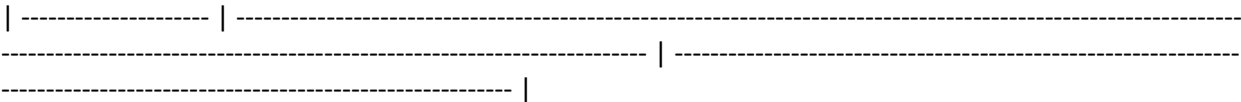
```

    getLayoutInfo: (node: Node, options: TreeInfo) => {
      if (!options.hasSubTree) {
        options.type = "Center";
        options.orientation = "Horizontal";
      }
    },
  },
}
//Configures data source for diagram
dataSourceSettings={{
  id: "Id",
  parentId: "Team",
  dataManager: items,
}}
//Sets the default properties for nodes and connectors
getNodeDefaults=(obj: NodeModel, diagram: Diagram) => {
  obj.width = 150;
  obj.height = 50;
  obj.style.fill = "#6BA5D7";
  obj.annotations = [
    {
      content: obj.data["Role"],
      style: {
        color: "white",
      },
    },
  ],
};
return obj;
}
getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.targetDecorator.shape = "None";
  connector.type = "Orthogonal";
  return connector;
}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

The following table illustrates the properties that “options” argument takes.

Property	Description
Default Value	



| options.assistants | By default, the collection is empty. When any of the child nodes have to be set as **Assistant**, you can remove from children collection and have to insert into assistants collection. | Empty array

| options.orientation | Gets or sets the organizational chart orientation.
| SubTreeOrientation.Vertical

| options.type | Gets or sets the chart organizational chart type.
| For horizontal chart orientation:SubTreeAlignments.Center and for vertical chart orientation:SubTreeAlignments.Alternate

| options.offset | Offset is the horizontal space to be left between parent and child nodes.
| 20 pixels applicable only for vertical chart orientations.

| options.hasSubTree | Gets whether the node contains subtrees.
| Boolean

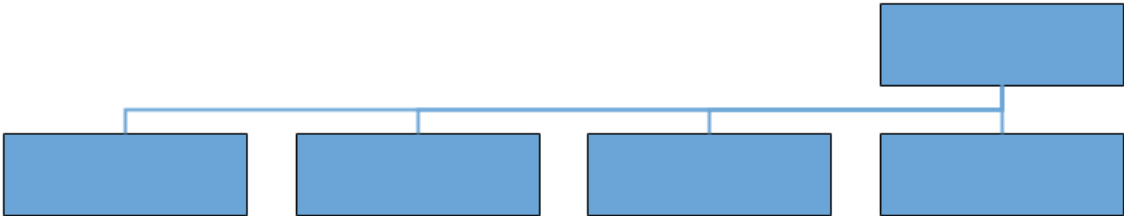
| options.level | Gets the depth of the node from layout root.
| Number

| options.enableRouting | By default, connections are routed based on the chart type and orientations. This property gets or sets whether default routing is to be enabled or disabled.
true

| options.rows | Sets the number of rows on which the child nodes will be arranged. Applicable only for balanced type horizontal tree. | Number

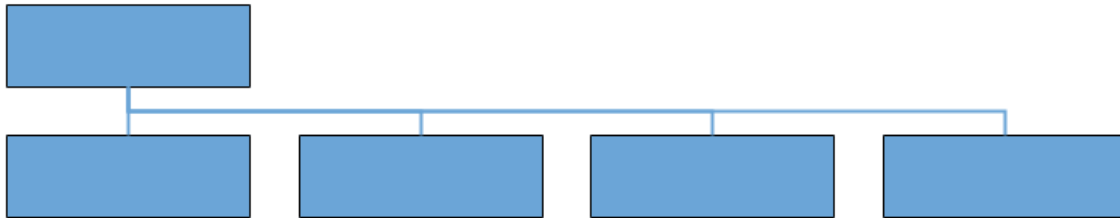
The following table illustrates the different chart orientations and chart types.

Orientation	Type	Description	Example
-----	-----	-----	-----
Horizontal	Left	Arranges the child nodes horizontally at the left side of the parent.	

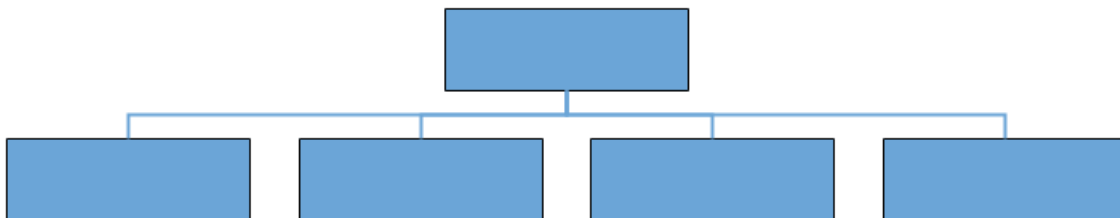


|

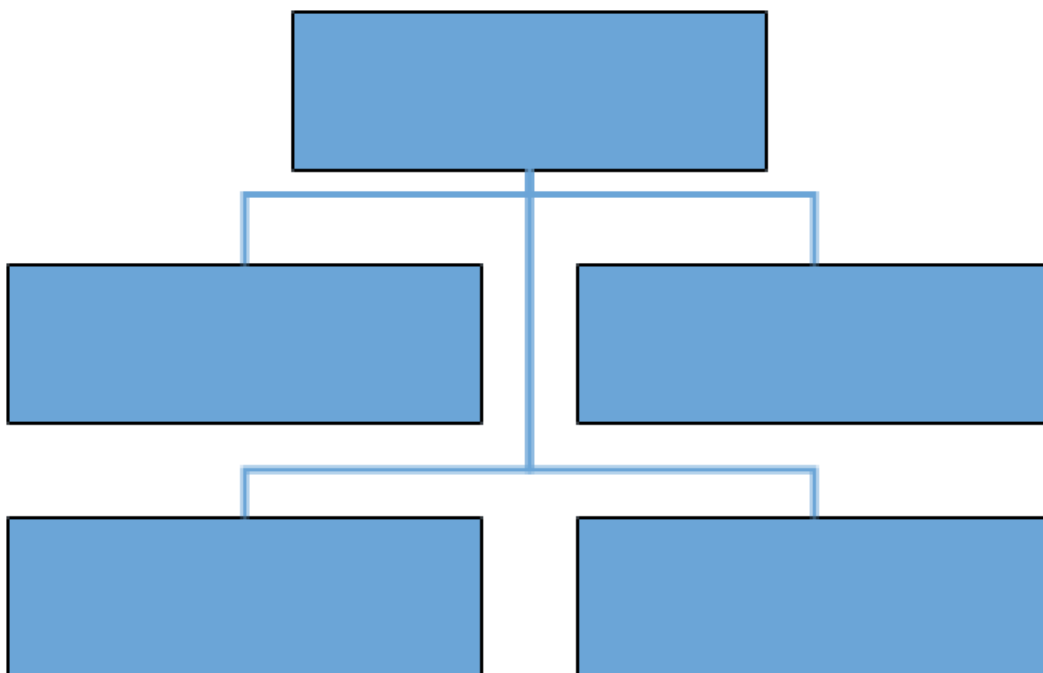
| | Right | Arranges the child nodes horizontally at the right side of the parent. |



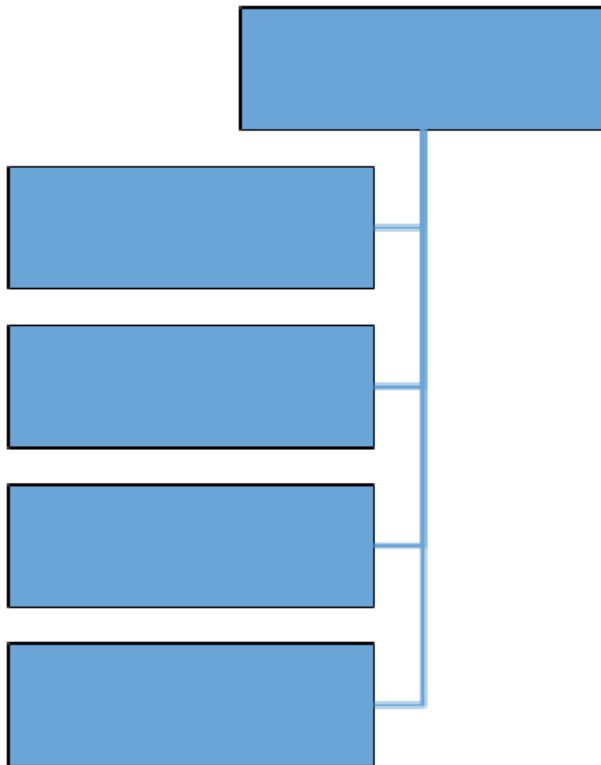
| | Center | Arranges the children like standard tree layout orientation. |



| | Balanced | Arranges the leaf level child nodes in multiple rows. |

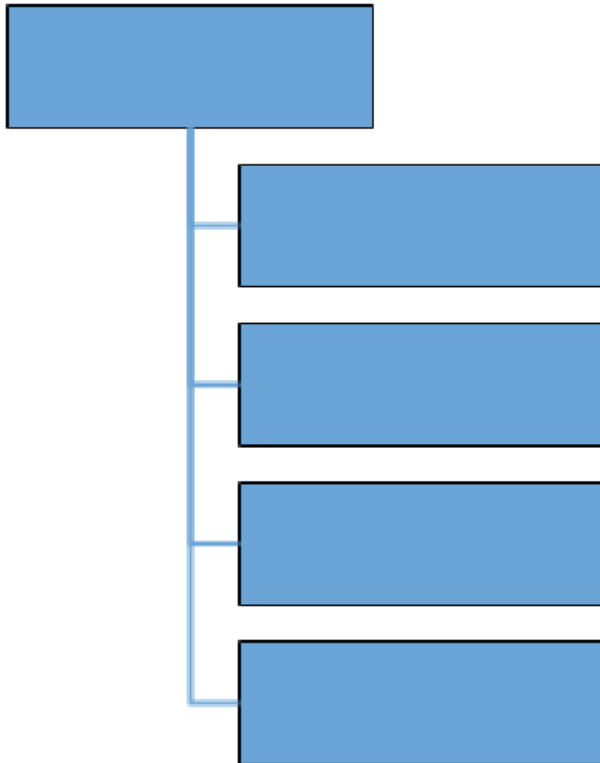


| Vertical | Left | Arranges the children vertically at the left side of the parent. |

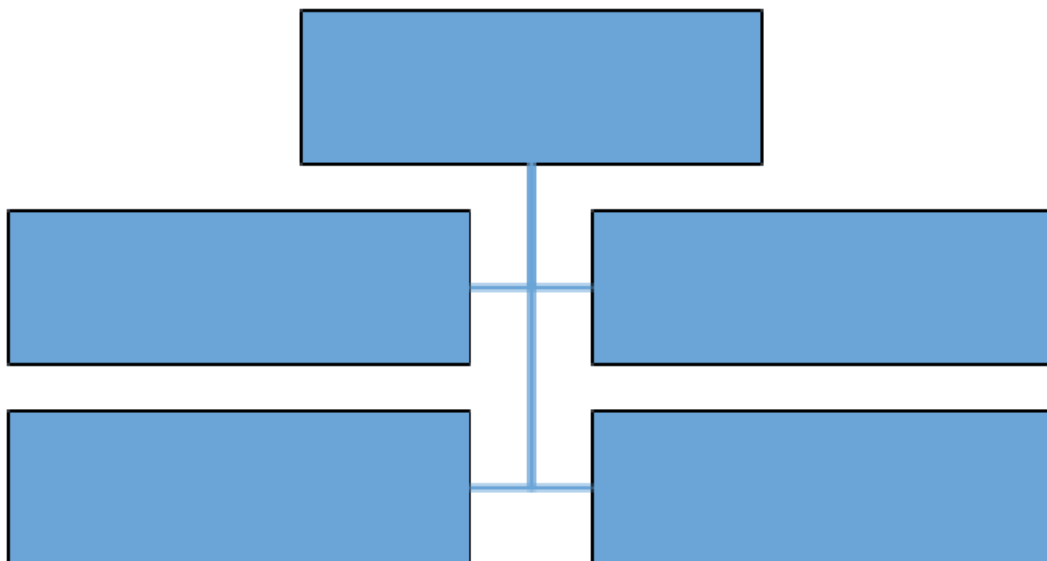


|

| | Right | Arranges the children vertically at the right side of the parent. |



| | Alternate | Arranges the children vertically at both left and right sides of the parent. |



The following code example illustrates how to set the vertical right arrangement to the leaf level trees.

```
{% raw %}  
`ts  
import * as React from "react";  
import * as ReactDOM from "react-dom";  
import {  
  Diagram,  
  DiagramComponent,  
  Inject,  
  ConnectorModel,  
  Node,  
  NodeModel,  
  DataBinding,  
  HierarchicalTree,  
  TreeInfo,  
} from "@syncfusion/ej2-react-diagrams";  
import { DataManager, Query } from "@syncfusion/ej2-data";  
//Initializes data source  
let data: object[] = [  
  {  
    Id: "parent",  
    Role: "Board",  
  },  
  {  
    Id: "1",  
    Role: "General Manager",  
    Manager: "parent",  
  },  
  {  
    Id: "2",  
    Role: "Human Resource Manager",  
    Manager: "1",  
  },  
];
```

```
{
  Id: "3",
  Role: "Trainers",
  Manager: "2",
},
{
  Id: "4",
  Role: "Recruiting Team",
  Manager: "2",
},
{
  Id: "6",
  Role: "Design Manager",
  Manager: "1",
},
{
  Id: "7",
  Role: "Design Supervisor",
  Manager: "6",
},
{
  Id: "8",
  Role: "Development Supervisor",
  Manager: "6",
},
{
  Id: "9",
  Role: "Drafting Supervisor",
  Manager: "6",
},
{
  Id: "10",
  Role: "Marketing Manager",
```

```
Manager: "1",
},
{
  Id: "11",
  Role: "Oversea sales Manager",
  Manager: "10",
},
{
  Id: "12",
  Role: "Petroleum Manager",
  Manager: "10",
},
{
  Id: "13",
  Role: "Service Dept. Manager",
  Manager: "10",
},
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      heigh={"530px"}
      snapSettings={{
        constraints: 0,
      }}
    //Uses layout to auto-arrange nodes on the diagram page
    layout={{
      //Sets layout type
      type: "OrganizationalChart",
      getLayoutInfo: (node: Node, options: TreeInfo) => {
        if (node.data["Role"] === "General Manager") {
```

```
options.assistants.push(options.children[0]);
options.children.splice(0, 1);
}
if (!options.hasSubTree) {
options.type = "Right";
options.orientation = "Vertical";
}
},
}}
//Configures data source for diagram
dataSourceSettings={{
id: "Id",
parentId: "Manager",
dataManager: new DataManager(data as JSON[]),
}}
//Sets the default properties for nodes and connectors
getNodeDefaults=((obj: NodeModel, diagram: Diagram) => {
obj.width = 150;
obj.height = 50;
obj.borderColor = "white";
obj.style.fill = "#6BA5D7";
obj.borderWidth = 1;
obj.annotations = [
{
content: obj.data["Role"],
style: {
color: "white",
},
},
];
return obj;
}}
getConnectorDefaults=((connector: ConnectorModel, diagram: Diagram) => {
```



```

connector.style = {
  strokeColor: "#6BA5D7",
  strokeWidth: 2,
};
connector.targetDecorator.style.fill = "#6BA5D7";
connector.targetDecorator.style.strokeColor = "#6BA5D7";
connector.targetDecorator.shape = "None";
connector.type = "Orthogonal";
return connector;
}

```

```

<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

{% endraw %}

```

Assistant

Assistants are child item that have a different relationship with the parent node. They are laid out in a dedicated part of the tree. A node can be specified as an assistant of its parent by adding it to the `assistants` property of the argument "options".

The following code example illustrates how to add assistants to layout.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, HierarchicalTree, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data
let data = [
  {
    Id: 1,
    Role: "General Manager",
  },
  {
    Id: 2,
    Role: "Assistant Manager",
  },
]

```

```

        Team: 1,
    },
    {
        Id: 3,
        Role: "Human Resource Manager",
        Team: 1,
    },
    {
        Id: 4,
        Role: "Design Manager",
        Team: 1,
    },
    {
        Id: 5,
        Role: "Operation Manager",
        Team: 1,
    },
    {
        Id: 6,
        Role: "Marketing Manager",
        Team: 1,
    },
];
let items = new DataManager(data, new Query().take(7));
function App() {
    return (<DiagramComponent id="container" width={"100%"} height={"530px"}
snapSettings={{
    constraints: 0,
    }}
//Uses layout to auto-arrange nodes on the diagram page
    layout={{
        //Sets layout type
        type: "OrganizationalChart",
        // define the getLayoutInfo
        getLayoutInfo: (node, options) => {
            if (node.data["Role"] === "General Manager") {
                options.assistants.push(options.children[0]);
                options.children.splice(0, 1);
            }
            if (!options.hasSubTree) {
                options.type = "Center";
                options.orientation = "Horizontal";
            }
        },
    }}
    //Initializes the node template.
    dataSourceSettings={{
        id: "Id",
        parentId: "Team",
        dataManager: items,
    }}
    //Sets the default properties for nodes and connectors
    getNodeDefaults=(obj, diagram) => {
        obj.width = 150;
        obj.height = 50;
        obj.borderColor = "white";
        obj.style.fill = "#6BA5D7";
    }

```

```

        obj.borderWidth = 1;
        obj.annotations = [
            {
                content: obj.data["Role"],
                style: {
                    color: "white",
                },
            },
        ];
        return obj;
    }} getConnectorDefaults=({connector, diagram}) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.targetDecorator.shape = "None";
        connector.type = "Orthogonal";
        return connector;
    }}>
    <Inject services={[DataBinding, HierarchicalTree]}/>
</DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    ConnectorModel,
    Node,
    NodeModel,
    DataBinding,
    HierarchicalTree,
    TreeInfo,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data
let data: object[] = [
    {
        Id: 1,
        Role: "General Manager",
    },
    {
        Id: 2,
        Role: "Assistant Manager",
        Team: 1,
    },
],

```

```

{
  Id: 3,
  Role: "Human Resource Manager",
  Team: 1,
},
{
  Id: 4,
  Role: "Design Manager",
  Team: 1,
},
{
  Id: 5,
  Role: "Operation Manager",
  Team: 1,
},
{
  Id: 6,
  Role: "Marketing Manager",
  Team: 1,
},
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"530px"}
      snapSettings={{
        constraints: 0,
      }}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type
        type: "OrganizationalChart",
        // define the getLayoutInfo
        getLayoutInfo: (node: Node, options: TreeInfo) => {
          if (node.data["Role"] === "General Manager") {
            options.assistants.push(options.children[0]);
            options.children.splice(0, 1);
          }
          if (!options.hasSubTree) {
            options.type = "Center";
            options.orientation = "Horizontal";
          }
        },
      }},
      //Initializes the node template.
      dataSourceSettings={{
        id: "Id",
        parentId: "Team",
        dataManager: items,
      }}
      //Sets the default properties for nodes and connectors
      getNodeDefaults=(obj: NodeModel, diagram: Diagram) => {
        obj.width = 150;

```

```

    obj.height = 50;
    obj.borderColor = "white";
    obj.style.fill = "#6BA5D7";
    obj.borderWidth = 1;
    obj.annotations = [
      {
        content: obj.data["Role"],
        style: {
          color: "white",
        },
      },
    ];
    return obj;
  }
}
getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.targetDecorator.shape = "None";
  connector.type = "Orthogonal";
  return connector;
}
}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Symmetric layout

The symmetric layout has been formed using nodes position by closer together or pushing them further apart. This is repeated iteratively until the system comes to an equilibrium state.

The layout's [springLength](#) defined as how long edges should be, ideally. This will be the resting length for the springs. Edge attraction and vertex repulsion forces to be defined by using layout's [springFactor](#), the more sibling nodes repel each other. The relative positions do not change any more from one iteration to the next. The number of iterations can be specified by using layout's [maxIteration](#).

Note: If you want to use symmetric layout in diagram, you need to inject SymmetricLayout in the diagram.

The following code illustrates how to arrange the nodes in a radial tree structure.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, SymmetricLayout, } from "@syncfusion/ej2-react-diagrams";

```

```

let nodes = [];
let connectors = [];
// creating the connection between the layout nodes and connectors.
export function ConnectNodes(parentNode, childNode) {
  let connector = {
    id: parentNode.id + childNode.id,
    sourceID: parentNode.id,
    targetID: childNode.id,
    targetDecorator: {
      shape: "None",
    },
  },
  };
  return connector;
}
// creating the layout nodes as rectangle in shape.
export function GetRectangle(name) {
  let shape = {
    type: "Basic",
    shape: "Ellipse",
  };
  let node = {
    id: name,
    height: 25,
    width: 25,
    borderColor: "white",
    borderWidth: 1,
    style: {
      fill: "#6BA5D7",
    },
    shape: shape,
  };
  return node;
}
// creating the symmetrical layout child elements hierarchy.
export function populateNodes() {
  let parentRect = GetRectangle("p");
  nodes.push(parentRect);
  for (let i = 0; i < 2; i++) {
    let childRect_i = GetRectangle("c" + i);
    nodes.push(childRect_i);
    for (let j = 0; j < 2; j++) {
      let childRect_j = GetRectangle("c" + i + j);
      nodes.push(childRect_j);
      for (let k = 0; k < 6; k++) {
        let childRect_k = GetRectangle("c" + i + j + k);
        nodes.push(childRect_k);
        connectors.push(ConnectNodes(childRect_j, childRect_k));
      }
      connectors.push(ConnectNodes(childRect_i, childRect_j));
    }
    connectors.push(ConnectNodes(parentRect, childRect_i));
  }
  return nodes;
}
//sets the layout child elements
populateNodes();
function App() {

```

```

    return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    layout={{
        //sets the layout as symmetric layout
        type: "SymmetricalLayout",
        springLength: 80,
        springFactor: 0.8,
        maxIteration: 500,
        margin: {
            left: 20,
            top: 20,
        },
    },
    nodes={nodes} connectors={connectors} getNodeDefaults={ (obj,
    diagram) => {
        obj.borderColor = "white";
        obj.borderWidth = 1;
        return obj;
    }} getConnectorDefaults={ (connector, diagram) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        return connector;
    }}>
    <Inject services={[SymmetricLayout]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    ConnectorModel,
    IConnector,
    DataBinding,
    HierarchicalTree,
    LayoutOrientation,
    LayoutType,
    Rect,
    HorizontalAlignment,
    VerticalAlignment,
    NodeModel,
    BasicShapeModel,
    SymmetricLayout,
} from "@syncfusion/ej2-react-diagrams";
let nodes: NodeModel[] = [];
let connectors: ConnectorModel[] = [];

```

```

// creating the connection between the layout nodes and connectors.
export function ConnectNodes(
  parentNode: NodeModel,
  childNode: NodeModel
): ConnectorModel {
  let connector: ConnectorModel = {
    id: parentNode.id + childNode.id,
    sourceID: parentNode.id,
    targetID: childNode.id,
    targetDecorator: {
      shape: "None",
    },
  },
  };
  return connector;
}

// creating the layout nodes as rectangle in shape.
export function GetRectangle(name: string): NodeModel {
  let shape: BasicShapeModel = {
    type: "Basic",
    shape: "Ellipse",
  };
  let node: NodeModel = {
    id: name,
    height: 25,
    width: 25,
    borderColor: "white",
    borderWidth: 1,
    style: {
      fill: "#6BA5D7",
    },
    shape: shape,
  };
  return node;
}

// creating the symmetrical layout child elements hierarchy.
export function populateNodes() {
  let parentRect: NodeModel = GetRectangle("p");
  nodes.push(parentRect);
  for (let i: number = 0; i < 2; i++) {
    let childRect_i: NodeModel = GetRectangle("c" + i);
    nodes.push(childRect_i);
    for (let j: number = 0; j < 2; j++) {
      let childRect_j: NodeModel = GetRectangle("c" + i + j);
      nodes.push(childRect_j);
      for (let k: number = 0; k < 6; k++) {
        let childRect_k: NodeModel = GetRectangle("c" + i + j + k);
        nodes.push(childRect_k);
        connectors.push(ConnectNodes(childRect_j, childRect_k));
      }
      connectors.push(ConnectNodes(childRect_i, childRect_j));
    }
    connectors.push(ConnectNodes(parentRect, childRect_i));
  }
  return nodes;
}

//sets the layout child elements
populateNodes();

```



```
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      layout={{
        //sets the layout as symmetric layout
        type: "SymmetricalLayout",
        springLength: 80,
        springFactor: 0.8,
        maxIteration: 500,
        margin: {
          left: 20,
          top: 20,
        },
      }}
      nodes={nodes}
      connectors={connectors}
      getNodeDefaults={(obj: NodeModel, diagram: Diagram) => {
        obj.borderColor = "white";
        obj.borderWidth = 1;
        return obj;
      }}
      getConnectorDefaults={(connector: ConnectorModel, diagram: Diagram) =>
    {
      connector.style = {
        strokeColor: "#6BA5D7",
        strokeWidth: 2,
      };
      connector.targetDecorator.style.fill = "#6BA5D7";
      connector.targetDecorator.style.strokeColor = "#6BA5D7";
      return connector;
    }}
    >
    <Inject services={[SymmetricLayout]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}
```

Mind Map layout

A mind map is a diagram that displays the nodes as a spider diagram organizes information around a central concept. To create mind map, the [type](#) of layout should be set as **MindMap**.

Tree Orientation in layout

An [Orientation](#) of a **MindMapTreeLayout** is used to arrange the tree layout based on the direction. The default value for the orientation is Horizontal. The different orientation types are defined in the following table:

Orientation Type	Description
------------------	-------------

Horizontal	Aligns the tree layout from left to right
Vertical	Aligns the tree layout from top to bottom

Note: If you want to use mind map layout in diagram, you need to inject MindMap in the diagram.

The following code example illustrates how to create an mindmap layout.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, MindMap, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    id: 1,
    Label: "StackPanel",
  },
  {
    id: 2,
    Label: "Label",
    parentId: 1,
  },
  {
    id: 3,
    Label: "ListBox",
    parentId: 1,
  },
  {
    id: 4,
    Label: "StackPanel",
    parentId: 2,
  },
  {
    id: 5,
    Label: "Border",
    parentId: 2,
  },
  {
    id: 6,
    Label: "Border",
    parentId: 4,
  },
  {
    id: 7,
    Label: "Button",
    parentId: 4,
  },
  {
    id: 8,
    Label: "ContentPresenter",
    parentId: 5,
  },
]
```

```

    },
    {
      id: 9,
      Label: "Text Block",
      parentId: 5,
    },
  ],
];
let items = new DataManager(data, new Query().take(7));
function App() {
  return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    //Uses layout to auto-arrange nodes on the diagram page
    layout={{
      //Sets layout type
      type: "MindMap",
      orientation: "Horizontal"
    }}
    //Configures data source for diagram
    dataSourceSettings={{
      id: "id",
      parentId: "parentId",
      dataManager: items,
      root: String(1),
    }}
    //Sets the default properties for nodes and connectors
    getNodeDefaults=(obj) => {
      obj.shape = {
        type: "Text",
        content: obj.data.Label,
      };
      obj.style = {
        fill: "#6BA5D7",
        strokeColor: "none",
        strokeWidth: 2,
      };
      obj.borderColor = "white";
      obj.backgroundColor = "#6BA5D7";
      obj.borderWidth = 1;
      obj.shape.margin = {
        left: 5,
        right: 5,
        top: 5,
        bottom: 5,
      };
    };
    return obj;
  }} getConnectorDefaults=(connector, diagram) => {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }}>
  <Inject services={[DataBinding, MindMap]}/>
</DiagramComponent>;
}

```

```
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  Node,
  DataBinding,
  LayoutType,
  Rect,
  HorizontalAlignment,
  VerticalAlignment,
  NodeModel,
  TextModel,
  MindMap,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    id: 1,
    Label: "StackPanel",
  },
  {
    id: 2,
    Label: "Label",
    parentId: 1,
  },
  {
    id: 3,
    Label: "ListBox",
    parentId: 1,
  },
  {
    id: 4,
    Label: "StackPanel",
    parentId: 2,
  },
  {
    id: 5,
    Label: "Border",
    parentId: 2,
  },
  {
    id: 6,
    Label: "Border",
    parentId: 4,
  },
],
```

```

{
  id: 7,
  Label: "Button",
  parentId: 4,
},
{
  id: 8,
  Label: "ContentPresenter",
  parentId: 5,
},
{
  id: 9,
  Label: "Text Block",
  parentId: 5,
},
},
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type
        type: "MindMap",
        orientation: "Horizontal"
      }}
      //Configures data source for diagram
      dataSourceSettings={{
        id: "id",
        parentId: "parentId",
        dataManager: items,
        root: String(1),
      }}
      //Sets the default properties for nodes and connectors
      getNodeDefaults=(obj: NodeModel) => {
        obj.shape = {
          type: "Text",
          content: (
            obj.data as {
              Label: "string";
            }
          ).Label,
        };
        obj.style = {
          fill: "#6BA5D7",
          strokeColor: "none",
          strokeWidth: 2,
        };
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        (obj.shape as TextModel).margin = {
          left: 5,

```

```

        right: 5,
        top: 5,
        bottom: 5,
    };
    return obj;
  }}
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
  {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }}
  >
  <Inject services={[DataBinding, MindMap]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Complex hierarchical tree

Complex hierarchical tree layout is the extended version of the hierarchical tree layout. The child had been two or more parents. To create a complex hierarchical tree, the [type](#) of layout should be set as `ComplexHierarchicalTree`.

Note: If you want to use Complex hierarchical layout in diagram, you need to inject `ComplexHierarchicalTree` in the diagram.

The following code example illustrates how to create a complex hierarchical tree.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DataBinding, ComplexHierarchicalTree, }
from "@syncfusion/ej2-react-diagrams";
//Initializes nodes
let node = [
  {
    id: "node1",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node1",
      },
    ],
  },
],
},
{

```

```
    id: "node2",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node2",
      },
    ],
  },
  {
    id: "node3",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node3",
      },
    ],
  },
  {
    id: "node4",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node4",
      },
    ],
  },
  {
    id: "node5",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node5",
      },
    ],
  },
  {
    id: "node8",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node8",
      },
    ],
  },
  {
    id: "node9",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node9",
      },
    ],
  },
]
```

```

    ],
  },
];
let connector = [
  {
    id: "connectr",
    sourceID: "node1",
    targetID: "node4",
  },
  {
    id: "connectr1",
    sourceID: "node2",
    targetID: "node4",
  },
  {
    id: "connectr3",
    sourceID: "node3",
    targetID: "node4",
  },
  {
    id: "connectr4",
    sourceID: "node4",
    targetID: "node5",
  },
];
function App() {
  return (<DiagramComponent id="container" width={1000} height={1000}
nodes={node} connectors={connector}
//Uses layout to auto-arrange nodes on the diagram page
  layout={{
    //Sets layout type
    type: "ComplexHierarchicalTree",
    orientation: "TopToBottom",
  }}
//Sets the default properties for nodes and connectors
  getNodeDefaults=(obj) => {
    obj.shape = {
      type: "Text",
    };
    obj.style = {
      fill: "#6BA5D7",
      strokeColor: "none",
      strokeWidth: 2,
    };
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    obj.shape.margin = {
      left: 5,
      right: 5,
      top: 5,
      bottom: 5,
    };
  };
  return obj;
}} getConnectorDefaults=(connector, diagram) => {
  connector.style = {
    strokeColor: "#6BA5D7",

```



```

        strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }
}
<Inject services={[ComplexHierarchicalTree, DataBinding]} />
</DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  TextModel,
  DataBinding,
  DiagramConstraints,
  ComplexHierarchicalTree,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes nodes
let node: NodeModel[] = [
  {
    id: "node1",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node1",
      },
    ],
  },
  {
    id: "node2",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node2",
      },
    ],
  },
  {
    id: "node3",
    width: 70,

```

```

    height: 70,
    annotations: [
      {
        content: "node3",
      },
    ],
  },
  {
    id: "node4",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node4",
      },
    ],
  },
  {
    id: "node5",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node5",
      },
    ],
  },
  {
    id: "node8",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node8",
      },
    ],
  },
  {
    id: "node9",
    width: 70,
    height: 70,
    annotations: [
      {
        content: "node9",
      },
    ],
  },
];
let connector: ConnectorModel[] = [
  {
    id: "connectr",
    sourceID: "node1",
    targetID: "node4",
  },
  {
    id: "connectr1",
    sourceID: "node2",

```

```

    targetID: "node4",
  },
  {
    id: "connectr3",
    sourceID: "node3",
    targetID: "node4",
  },
  {
    id: "connectr4",
    sourceID: "node4",
    targetID: "node5",
  },
];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={1000}
      height={1000}
      nodes={node}
      connectors={connector}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type
        type: "ComplexHierarchicalTree",
        orientation: "TopToBottom",
      }}
      //Sets the default properties for nodes and connectors
      getNodeDefaults=(obj: NodeModel) => {
        obj.shape = {
          type: "Text",
        };
        obj.style = {
          fill: "#6BA5D7",
          strokeColor: "none",
          strokeWidth: 2,
        };
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        (obj.shape as TextModel).margin = {
          left: 5,
          right: 5,
          top: 5,
          bottom: 5,
        };
        return obj;
      }}
      getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
    {
      connector.style = {
        strokeColor: "#6BA5D7",
        strokeWidth: 2,
      };
      connector.targetDecorator.style.fill = "#6BA5D7";
      connector.targetDecorator.style.strokeColor = "#6BA5D7";
      connector.type = "Orthogonal";
    }
  );
}

```

```

    return connector;
  }}
  >
  <Inject services={[ComplexHierarchicalTree, DataBinding]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Line Distribution

Line distribution is used to arrange the connectors without overlapping in automatic layout. In some cases, the automatic layout connectors connecting to the nodes will be overlapped with one another. So user can decide whether the segment of each connector from a single parent node should be same point or different point. The [connectionPointOrigin](#) property of layout is used to enable or disable the line distribution in layout. By default ConnectionPointOrigin will be SamePoint.

The following code example illustrates how to create a complex hierarchical tree with line distribution.

Note: If you want to use line distribution in diagram layout, you need to inject LineDistribution module in the diagram.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, ComplexHierarchicalTree,
ConnectionPointOrigin, LineDistribution, DataBinding, } from
"@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
let diagramInstance;
//Initializes data source
let data = [
  { Name: "node11" },
  { Name: "node12", ReportingPerson: ["node114"] },
  { Name: "node13", ReportingPerson: ["node12"] },
  { Name: "node14", ReportingPerson: ["node12"] },
  { Name: "node15", ReportingPerson: ["node12"] },
  { Name: "node116", ReportingPerson: ["node22", "node12"] },
  { Name: "node16", ReportingPerson: [] },
  { Name: "node18", ReportingPerson: [] },
  { Name: "node21" },
  { Name: "node22", ReportingPerson: ["node114"] },
  { Name: "node23", ReportingPerson: ["node22"] },
  { Name: "node24", ReportingPerson: ["node22"] },
  { Name: "node25", ReportingPerson: ["node22"] },
  { Name: "node26", ReportingPerson: [] },
  { Name: "node28", ReportingPerson: [] },
  { Name: "node31" },
  { Name: "node114", ReportingPerson: ["node11", "node21", "node31"] },
];
let items = new DataManager(data, new Query().take(7));
function App() {

```

```

    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={"100%"} height={"590px"} snapSettings={{
    constraints: 0,
  }} created={(args) => {
    diagramInstance.fitToPage({ mode: "Width" });
  }} layout={{
    type: "ComplexHierarchicalTree",
    connectionPointOrigin: ConnectionPointOrigin.DifferentPoint,
    horizontalSpacing: 40,
    verticalSpacing: 40,
    horizontalAlignment: "Left",
    verticalAlignment: "Top",
    margin: { left: 0, right: 0, top: 0, bottom: 0 },
    orientation: "TopToBottom",
  }} dataSourceSettings={{
    id: "Name",
    parentId: "ReportingPerson",
    dataManager: items,
  }} getNodeDefaults={(obj) => {
    obj.width = 40;
    obj.height = 40;
    obj.shape = { type: "Basic", shape: "Rectangle" };
    obj.style = { fill: "#6BA5D7", strokeColor: "none", strokeWidth:
2  };

    obj.borderWidth = 1;
    obj.backgroundColor = "#6BA5D7";
    return obj;
  }} getConnectorDefaults={(connector, diagram) => {
    connector.type = "Orthogonal";
    connector.cornerRadius = 7;
    connector.targetDecorator.height = 7;
    connector.targetDecorator.width = 7;
    connector.style = { strokeColor: "#6BA5D7", strokeWidth: 1 };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    return connector;
  }}>
    <Inject services={[DataBinding, ComplexHierarchicalTree,
LineDistribution]}/>
    </DiagramComponent>;
  }
  // Initializes the diagram
  const root = ReactDOM.createRoot(document.getElementById("diagram"));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,

```

```

NodeModel,
DiagramConstraints,
ComplexHierarchicalTree,
ConnectionPointOrigin,
LineDistribution,
DataBinding,
Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
let diagramInstance: Diagram;
//Initializes data source
let data: object[] = [
  { Name: "node11" },
  { Name: "node12", ReportingPerson: ["node114"] },
  { Name: "node13", ReportingPerson: ["node12"] },
  { Name: "node14", ReportingPerson: ["node12"] },
  { Name: "node15", ReportingPerson: ["node12"] },
  { Name: "node116", ReportingPerson: ["node22", "node12"] },
  { Name: "node16", ReportingPerson: [] },
  { Name: "node18", ReportingPerson: [] },
  { Name: "node21" },
  { Name: "node22", ReportingPerson: ["node114"] },
  { Name: "node23", ReportingPerson: ["node22"] },
  { Name: "node24", ReportingPerson: ["node22"] },
  { Name: "node25", ReportingPerson: ["node22"] },
  { Name: "node26", ReportingPerson: [] },
  { Name: "node28", ReportingPerson: [] },
  { Name: "node31" },
  { Name: "node114", ReportingPerson: ["node11", "node21", "node31"] },
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={"100%"}
      height={"590px"}
      snapSettings={{
        constraints: 0,
      }}
      created={(args) => {
        diagramInstance.fitToPage({ mode: "Width" });
      }}
      layout={{
        type: "ComplexHierarchicalTree",
        connectionPointOrigin: ConnectionPointOrigin.DifferentPoint,
        horizontalSpacing: 40,
        verticalSpacing: 40,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
        margin: { left: 0, right: 0, top: 0, bottom: 0 },
        orientation: "TopToBottom",
      }}
      dataSourceSettings={{
        id: "Name",

```

```

    parentId: "ReportingPerson",
    dataManager: items,
  })
  getNodeDefaults=(obj: Node) => {
    obj.width = 40;
    obj.height = 40;
    obj.shape = { type: "Basic", shape: "Rectangle" };
    obj.style = { fill: "#6BA5D7", strokeColor: "none", strokeWidth: 2
  };

    obj.borderWidth = 1;
    obj.backgroundColor = "#6BA5D7";
    return obj;
  })
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
  {
    connector.type = "Orthogonal";
    connector.cornerRadius = 7;
    connector.targetDecorator.height = 7;
    connector.targetDecorator.width = 7;
    connector.style = { strokeColor: "#6BA5D7", strokeWidth: 1 };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    return connector;
  })
  >
  <Inject
    services={[DataBinding, ComplexHierarchicalTree, LineDistribution]}
  />
  </DiagramComponent>
);
}
// Initializes the diagram
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Linear Arrangement

Linear arrangement is used to linearly arrange the child nodes in layout, which means the parent node is placed in the center corresponding to its children. When line distribution is enabled, linear arrangement is also activated by default. The [arrangement](#) property of layout is used to enable or disable the linear arrangement in layout. By default childArrangement will be **Nonlinear**.

Note: If you want to use linear arrangement in diagram layout, you need to inject LineDistribution module in the diagram. Linear arrangement is applicable only for complex hierarchical tree layout.

```
{% raw %}
```

```
`ts
```

```
function App() {
```

```
  return (
```

```
    <DiagramComponent
```

```
      id="container"
```

```

width={"100%"}
height={"590px"}
layout={{
  type: "ComplexHierarchicalTree",
  //To arrange a child nodes in a linear manner
  arrangement: ChildArrangement.Linear,
  horizontalSpacing: 40,
  verticalSpacing: 40,
  orientation: "TopToBottom",
}}

<Inject
  services={[DataBinding, ComplexHierarchicalTree, LineDistribution]}
/>
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`

```

```
{% endraw %}
```

Prevent connectors overlay

The below constraints prevents the connector segments overlapping nodes with a complex hierarchical layout.

```

{% raw %}
`ts
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"590px"}
      layout={{
        //this prevents connector segments overlapping

```



```
enableRouting: true,
}}
```

```
<Inject
services={[DataBinding, ComplexHierarchicalTree, LineDistribution]}
/>
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`
```

```
{% endraw %}
```

Customize layout

Orientation, spacings, and position of the layout can be customized with a set of properties.

To explore layout properties, refer to [Layout Properties](#).

Layout bounds

Diagram provides support to align the layout within any custom rectangular area. For more information about bounds, refer to [bounds](#).

Layout alignment

The layout can be aligned anywhere over the layout bounds/viewport using the [horizontalAlignment](#) and [verticalAlignment](#) properties of the layout.

The following code illustrates how to align the layout at the top-left of the layout bounds.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, Rect, }
from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Kevin-Manager",
  },
];
```

```

    },
    {
      Name: "John- Manager",
      ReportingPerson: "Peter-Manager",
    },
    {
      Name: "Mary-CSE ",
      ReportingPerson: "Peter-Manager",
    },
  ],
];
let items = new DataManager(data, new Query().take(7));
function App() {
  <DiagramComponent id="container" width={"100%"} height={"550px"}
    //Uses layout to auto-arrange nodes on the diagram page
    layout={
      {
        //Sets layout type
        type: "HierarchicalTree",
        //set layout alignment
        bounds: new Rect(0, 0, 500, 500),
        horizontalSpacing: 25,
        verticalSpacing: 30,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
      }
    }
    dataSourceSettings={
      {
        id: "Name",
        parentId: "ReportingPerson",
        dataManager: items,
      }
    }
    getNodeDefaults={
      (obj) => {
        obj.shape = {
          type: "Text",
          content: obj.data.Name,
        };
        obj.style = {
          fill: "None",
          strokeColor: "none",
          strokeWidth: 2,
          bold: true,
          color: "white",
        };
        obj.width = 100;
        obj.height = 40;
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        return obj;
      }
    }
    getConnectorDefaults={
      (connector, diagram) => {
        connector.style = {
          strokeColor: "#6BA5D7",
          strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.type = "Orthogonal";
        return connector;
      }
    }
  </DiagramComponent>
  <Inject services={[DataBinding, HierarchicalTree]}/>
  </DiagramComponent>;

```

```

}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  DiagramConstraints,
  HierarchicalTree,
  TextModel,
  DataBinding,
  Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
],
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type

```

```

    type: "HierarchicalTree",
    //set layout alignment
    bounds: new Rect(0, 0, 500, 500),
    horizontalSpacing: 25,
    verticalSpacing: 30,
    horizontalAlignment: "Left",
    verticalAlignment: "Top",
  }}
  dataSourceSettings={{
    id: "Name",
    parentId: "ReportingPerson",
    dataManager: items,
  }}
  getNodeDefaults=(obj: NodeModel) => {
    obj.shape = {
      type: "Text",
      content: (
        obj.data as {
          Name: "string";
        }
      ).Name,
    };
    obj.style = {
      fill: "None",
      strokeColor: "none",
      strokeWidth: 2,
      bold: true,
      color: "white",
    };
    obj.width = 100;
    obj.height = 40;
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    return obj;
  }}
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) => {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Layout spacing

Layout provides support to add space horizontally and vertically between the nodes. The [horizontalSpacing](#) and [verticalSpacing](#) properties of the layout allows you to set the space between the nodes in horizontally and vertically.

Layout margin

Layout provides support to add some blank space between the layout bounds/viewport and the layout. The [margin](#) property of the layout allows you to set the blank space.

The following code illustrates how to set the layout margin.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, Rect, }
from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
];
let items = new DataManager(data, new Query().take(7));
function App() {
  return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    //Uses layout to auto-arrange nodes on the diagram page
    layout={{
      //Sets layout type
      type: "HierarchicalTree",
      bounds: new Rect(0, 0, 500, 500),
      horizontalSpacing: 25,
      verticalSpacing: 30,
      horizontalAlignment: "Left",
      verticalAlignment: "Top",
    }}
    //Sets the default properties for nodes and connectors
    dataSourceSettings={{
      id: "Name",
    }}
  >);
}
```

```

        parentId: "ReportingPerson",
        dataManager: items,
    }}
    //Configures data source for diagram
    getNodeDefaults=(obj) => {
        obj.shape = {
            type: "Text",
            content: obj.data.Name,
        };
        obj.style = {
            fill: "None",
            strokeColor: "none",
            strokeWidth: 2,
            bold: true,
            color: "white",
        };
        obj.width = 50;
        obj.height = 40;
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        obj.shape.margin = {
            left: 25,
            right: 25,
            top: 25,
            bottom: 25,
        };
        return obj;
    }}
    getConnectorDefaults=(connector, diagram) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.type = "Orthogonal";
        return connector;
    }}
    <Inject services={[DataBinding, HierarchicalTree]}/>
    </DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    Inject,
    ConnectorModel,
    NodeModel,

```

```

DiagramConstraints,
HierarchicalTree,
TextModel,
DataBinding,
Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type
        type: "HierarchicalTree",
        bounds: new Rect(0, 0, 500, 500),
        horizontalSpacing: 25,
        verticalSpacing: 30,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
      }}
      //Sets the default properties for nodes and connectors
      dataSourceSettings={{
        id: "Name",
        parentId: "ReportingPerson",
        dataManager: items,
      }}
      //Configures data source for diagram
      getNodeDefaults=(obj: NodeModel) => {
        obj.shape = {
          type: "Text",

```

```

        content: (
          obj.data as {
            Name: "string";
          }
        ).Name,
      );
    obj.style = {
      fill: "None",
      strokeColor: "none",
      strokeWidth: 2,
      bold: true,
      color: "white",
    };
    obj.width = 50;
    obj.height = 40;
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    (obj.shape as TextModel).margin = {
      left: 25,
      right: 25,
      top: 25,
      bottom: 25,
    };
    return obj;
  })
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.type = "Orthogonal";
  return connector;
}
>
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Layout orientation

The layout orientation can be used to arrange the layout based on the direction. There are different orientation types that are defined in the following table.

Orientation	Description
TopToBottom	Aligns the layout from top to bottom. All the roots are placed at top of diagram.

| LeftToRight | Aligns the layout from left to right. All the roots are placed at left of diagram. |

| BottomToTop | Aligns the layout from bottom to top. All the roots are placed at bottom of the diagram. |

| RightToLeft | Aligns the layout from right to left. All the roots are placed at right of the diagram. |

Diagram provides support to customize the [orientation](#) of layout. You can set the desired orientation using layout.orientation.

Note: In the diagram the default orientation is TopToBottom.

The following code illustrates how to arrange the nodes in a BottomToTop orientation.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, Rect, }
from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
    {
        Name: "Steve-Ceo",
    },
    {
        Name: "Kevin-Manager",
        ReportingPerson: "Steve-Ceo",
    },
    {
        Name: "Peter-Manager",
        ReportingPerson: "Steve-Ceo",
    },
    {
        Name: "John- Manager",
        ReportingPerson: "Peter-Manager",
    },
    {
        Name: "Mary-CSE ",
        ReportingPerson: "Peter-Manager",
    },
    {
        Name: "Jim-CSE ",
        ReportingPerson: "Kevin-Manager",
    },
    {
        Name: "Martin-CSE",
        ReportingPerson: "Kevin-Manager",
    },
];
let items = new DataManager(data, new Query().take(7));
function App() {
    return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    //Uses layout to auto-arrange nodes on the diagram page
    layout={{
        //Sets layout type
```

```

        type: "HierarchicalTree",
        bounds: new Rect(0, 0, 500, 500),
        horizontalSpacing: 25,
        verticalSpacing: 30,
        horizontalAlignment: "Left",
        verticalAlignment: "Top",
        orientation: "BottomToTop",
    }} dataSourceSettings={{
        id: "Name",
        parentId: "ReportingPerson",
        dataManager: items,
    }}
    //Sets the default properties for nodes and connectors
    getNodeDefaults=(obj) => {
        obj.shape = {
            type: "Text",
            content: obj.data.Name,
        };
        obj.style = {
            fill: "None",
            strokeColor: "none",
            strokeWidth: 2,
            bold: true,
            color: "white",
        };
        obj.width = 50;
        obj.height = 40;
        obj.borderColor = "white";
        obj.backgroundColor = "#6BA5D7";
        obj.borderWidth = 1;
        obj.shape.margin = {
            left: 25,
            right: 25,
            top: 25,
            bottom: 25,
        };
        return obj;
    }} getConnectorDefaults=(connector, diagram) => {
        connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,
        };
        connector.targetDecorator.style.fill = "#6BA5D7";
        connector.targetDecorator.style.strokeColor = "#6BA5D7";
        connector.type = "Orthogonal";
        return connector;
    }}>
    <Inject services={[DataBinding, HierarchicalTree]}/>
    </DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  DiagramConstraints,
  HierarchicalTree,
  TextModel,
  DataBinding,
  Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
],
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"
      width={"100%"}
      height={"550px"}
      //Uses layout to auto-arrange nodes on the diagram page
      layout={{
        //Sets layout type

```

```

    type: "HierarchicalTree",
    bounds: new Rect(0, 0, 500, 500),
    horizontalSpacing: 25,
    verticalSpacing: 30,
    horizontalAlignment: "Left",
    verticalAlignment: "Top",
    orientation: "BottomToTop",
  })
  dataSourceSettings={
    id: "Name",
    parentId: "ReportingPerson",
    dataManager: items,
  }
  //Sets the default properties for nodes and connectors
  getNodeDefaults=(obj: NodeModel) => {
    obj.shape = {
      type: "Text",
      content: (
        obj.data as {
          Name: "string";
        }
      ).Name,
    };
    obj.style = {
      fill: "None",
      strokeColor: "none",
      strokeWidth: 2,
      bold: true,
      color: "white",
    };
    obj.width = 50;
    obj.height = 40;
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    (obj.shape as TextModel).margin = {
      left: 25,
      right: 25,
      top: 25,
      bottom: 25,
    };
    return obj;
  }
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.type = "Orthogonal";
  return connector;
}
}
<Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById("diagram"));
  root.render(<App />);
  {% enddraw %}

```

Fixed node

Layout provides support to arrange the nodes with reference to the position of a fixed node and set it to the [fixedNode](#) of the layout property. This is helpful when you try to expand/collapse a node. It might be expected that the position of the double-clicked node should not be changed.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, Rect, }
  from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Name: "Steve-Ceo",
    //set the offsetX and offsetY for the parent node
    offsetX: 250,
    offsetY: 50,
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
];
let items = new DataManager(data, new Query().take(7));
function App() {
  return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    //Defines default layout

```

```

layout={{
  type: "HierarchicalTree",
  bounds: new Rect(0, 0, 500, 500),
  horizontalSpacing: 25,
  verticalSpacing: 30,
  horizontalAlignment: "Left",
  verticalAlignment: "Top",
}} dataSourceSettings={{
  id: "Name",
  parentId: "ReportingPerson",
  dataManager: items,
}}
//Sets the default properties of the node.
getNodeDefaults=(obj) => {
  obj.shape = {
    type: "Text",
    content: obj.data.Name,
  };
  obj.style = {
    fill: "None",
    strokeColor: "none",
    strokeWidth: 2,
    bold: true,
    color: "white",
  };
  obj.width = 50;
  obj.height = 40;
  obj.borderColor = "white";
  obj.backgroundColor = "#6BA5D7";
  obj.borderWidth = 1;
  obj.shape.margin = {
    left: 25,
    right: 25,
    top: 25,
    bottom: 25,
  };
  return obj;
}
//Sets the default properties of the connector.
getConnectorDefaults=(connector, diagram) => {
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.type = "Orthogonal";
  return connector;
}>
  <Inject services={[DataBinding, HierarchicalTree]}/>
</DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  DiagramConstraints,
  HierarchicalTree,
  TextModel,
  DataBinding,
  Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
    //set the offsetX and offsetY for the parent node
    offsetX: 250,
    offsetY: 50,
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
];
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
function App() {
  return (
    <DiagramComponent
      id="container"

```

```

width={"100%"}
height={"550px"}
//Defines default layout
layout={({
  type: "HierarchicalTree",
  bounds: new Rect(0, 0, 500, 500),
  horizontalSpacing: 25,
  verticalSpacing: 30,
  horizontalAlignment: "Left",
  verticalAlignment: "Top",
})}
dataSourceSettings={{
  id: "Name",
  parentId: "ReportingPerson",
  dataManager: items,
}}
//Sets the default properties of the node.
getNodeDefaults=(obj: NodeModel) => {
  obj.shape = {
    type: "Text",
    content: (
      obj.data as {
        Name: "string";
      }
    ).Name,
  };
  obj.style = {
    fill: "None",
    strokeColor: "none",
    strokeWidth: 2,
    bold: true,
    color: "white",
  };
  obj.width = 50;
  obj.height = 40;
  obj.borderColor = "white";
  obj.backgroundColor = "#6BA5D7";
  obj.borderWidth = 1;
  (obj.shape as TextModel).margin = {
    left: 25,
    right: 25,
    top: 25,
    bottom: 25,
  };
  return obj;
}
//Sets the default properties of the connector.
getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
{
  connector.style = {
    strokeColor: "#6BA5D7",
    strokeWidth: 2,
  };
  connector.targetDecorator.style.fill = "#6BA5D7";
  connector.targetDecorator.style.strokeColor = "#6BA5D7";
  connector.type = "Orthogonal";
  return connector;
}

```



```

    }}
  >
    <Inject services={[DataBinding, HierarchicalTree]} />
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Expand and collapse

Diagram allows to expand/collapse the subtrees of a layout. The node's `isExpanded` property allows you to expand/collapse its children. The following code example shows how to expand/collapse the children of a node.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, } from
"@syncfusion/ej2-react-diagrams";
//Initializes data source
//let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
let diagramInstance;
let nodes = [
  {
    id: "node1",
    width: 140,
    height: 50,
    isExpanded: false,
    annotations: [
      {
        content: "node1",
      },
    ],
    expandIcon: {
      shape: "Plus",
    },
    collapseIcon: {
      shape: "Minus",
    },
  },
  {
    id: "node2",
    width: 140,
    height: 50,
    annotations: [
      {
        content: "node2",
      },
    ],
  },
  {
    id: "node3",

```

```

        width: 140,
        height: 50,
        annotations: [
            {
                content: "node3",
            },
        ],
    },
];
let connectors = [
    {
        // Name of the connector
        id: "connector1",
        // ID of the source and target nodes
        sourceID: "node1",
        targetID: "node2",
    },
    {
        id: "connector2",
        // ID of the source and target nodes
        sourceID: "node2",
        targetID: "node3",
    },
];
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
        (diagramInstance = diagram)} width={"1250px"} height={"590px"} nodes={nodes}
        connectors={connectors} snapSettings={{
            constraints: 0,
        }} created={args => {
            diagramInstance.fitToPage();
        }} layout={{
            // set enableAnimation as true
            enableAnimation: true,
            type: "HierarchicalTree",
            margin: {
                top: 20,
            },
            getLayoutInfo: (node, tree) => {
                if (!tree.hasSubTree) {
                    tree.orientation = "Vertical";
                    tree.type = "Alternate";
                }
            },
        }}
        //Sets the default properties for nodes and connectors
        getNodeDefaults={obj => {
            obj.shape = {
                type: "Text",
                style: {
                    color: "white",
                },
            };
            obj.style = {
                fill: "#6BA5D7",
                strokeColor: "none",
                strokeWidth: 2,
            };
        }}
    />);
}

```

```

    };
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    obj.shape.margin = {
      left: 5,
      right: 5,
      top: 5,
      bottom: 5,
    };
    return obj;
  }} getConnectorDefaults=({connector, diagram}) => {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }}>
  <Inject services={[HierarchicalTree]}/>
</DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  Node,
  SelectorConstraints,
  HierarchicalTree,
} from "@syncfusion/ej2-react-diagrams";
import { NodeModel } from "@syncfusion/ej2-react-diagrams";
//Initializes data source
//let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
let diagramInstance: Diagram;
let nodes: NodeModel[] = [
  {
    id: "node1",
    width: 140,
    height: 50,
    isExpanded: false,
    annotations: [
      {
        content: "node1",

```

```

    },
  ],
  expandIcon: {
    shape: "Plus",
  },
  collapseIcon: {
    shape: "Minus",
  },
},
{
  id: "node2",
  width: 140,
  height: 50,
  annotations: [
    {
      content: "node2",
    },
  ],
},
{
  id: "node3",
  width: 140,
  height: 50,
  annotations: [
    {
      content: "node3",
    },
  ],
},
];
let connectors: ConnectorModel[] = [
  {
    // Name of the connector
    id: "connector1",
    // ID of the source and target nodes
    sourceID: "node1",
    targetID: "node2",
  },
  {
    id: "connector2",
    // ID of the source and target nodes
    sourceID: "node2",
    targetID: "node3",
  },
];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={"1250px"}
      height={"590px"}
      nodes={nodes}
      connectors={connectors}
      snapSettings={{
        constraints: 0,
      }}
    />
  );
}

```

```

    created=({args}) => {
      diagramInstance.fitToPage();
    }
  layout=({
    // set enableAnimation as true
    enableAnimation: true,
    type: "HierarchicalTree",
    margin: {
      top: 20,
    }, // define the getLayoutInfo
    getLayoutInfo: (node: Node, tree: TreeInfo) => {
      if (!tree.hasSubTree) {
        tree.orientation = "Vertical";
        tree.type = "Alternate";
      }
    },
  })
  //Sets the default properties for nodes and connectors
  getNodeDefaults=(obj: Node) => {
    obj.shape = {
      type: "Text",
      style: {
        color: "white",
      },
    };
    obj.style = {
      fill: "#6BA5D7",
      strokeColor: "none",
      strokeWidth: 2,
    };
    obj.borderColor = "white";
    obj.backgroundColor = "#6BA5D7";
    obj.borderWidth = 1;
    (obj.shape as TextModel).margin = {
      left: 5,
      right: 5,
      top: 5,
      bottom: 5,
    };
    return obj;
  }
  getConnectorDefaults=(connector: ConnectorModel, diagram: Diagram) =>
  {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.type = "Orthogonal";
    return connector;
  }
}
<Inject services={[HierarchicalTree]} />
</DiagramComponent>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}
```

In the previous example, while expanding/collapsing a node, it is set as fixed node in order to prevent it from repositioning.

[Refresh layout](#)

Diagram allows to refresh the layout at runtime. To refresh the layout, refer to [Refresh layout](#).

[setNodeTemplate](#)

The `setNodeTemplate` function is provided for the purpose of customizing nodes. It will be called for each node on node initialization. In this function, the node style and its properties can be customized and can bind the custom JSON with node.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, HierarchicalTree, DataBinding, Rect, }
from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
    color: "darkcyan",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
    color: "white",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
    color: "darkcyan",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
    color: "white",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
    color: "darkcyan",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
  },
];
```

```

        color: "white",
      },
    ],
  ];
  let items = new DataManager(data, new Query().take(7));
  function App() {
    return (<DiagramComponent id="container" width={"100%"} height={"550px"}
    layout={{
      type: "HierarchicalTree",
      bounds: new Rect(0, 0, 500, 500),
      horizontalSpacing: 25,
      verticalSpacing: 30,
      horizontalAlignment: "Left",
      verticalAlignment: "Top",
    }} dataSourceSettings={{
      id: "Name",
      parentId: "ReportingPerson",
      dataManager: items,
    }} getNodeDefaults=(obj) => {
      obj.shape = {
        type: "Text",
        content: obj.data.Name,
      };
      obj.style = {
        fill: "None",
        strokeColor: "none",
        strokeWidth: 2,
        bold: true,
        color: "white",
      };
      obj.width = 50;
      obj.height = 40;
      obj.borderColor = "white";
      obj.backgroundColor = "#6BA5D7";
      obj.borderWidth = 1;
      obj.shape.margin = {
        left: 25,
        right: 25,
        top: 25,
        bottom: 25,
      };
    };
    return obj;
  }} getConnectorDefaults=(connector, diagram) => {
    connector.style = {
      strokeColor: "#6BA5D7",
      strokeWidth: 2,
    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.targetDecorator.shape = "None";
    connector.type = "Orthogonal";
    return connector;
  }} setNodeTemplate=function (obj, diagram) {
    obj.borderColor = obj.data.color;
  }}>
  <Inject services={[DataBinding, HierarchicalTree]}/>
</DiagramComponent>;
}

```

```
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  ConnectorModel,
  NodeModel,
  DiagramConstraints,
  HierarchicalTree,
  TextModel,
  DataBinding,
  Rect,
} from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from "@syncfusion/ej2-data";
//Initializes data source
let data: object[] = [
  {
    Name: "Steve-Ceo",
  },
  {
    Name: "Kevin-Manager",
    ReportingPerson: "Steve-Ceo",
    color: "darkcyan",
  },
  {
    Name: "Peter-Manager",
    ReportingPerson: "Steve-Ceo",
    color: "white",
  },
  {
    Name: "John- Manager",
    ReportingPerson: "Peter-Manager",
    color: "darkcyan",
  },
  {
    Name: "Mary-CSE ",
    ReportingPerson: "Peter-Manager",
    color: "white",
  },
  {
    Name: "Jim-CSE ",
    ReportingPerson: "Kevin-Manager",
    color: "darkcyan",
  },
  {
    Name: "Martin-CSE",
    ReportingPerson: "Kevin-Manager",
    color: "white",
  },
];
```



```

    },
  ];
  let items: DataManager = new DataManager(data as JSON[], new
  Query().take(7));
  function App() {
    return (
      <DiagramComponent
        id="container"
        width={"100%"}
        height={"550px"}
        layout={{
          type: "HierarchicalTree",
          bounds: new Rect(0, 0, 500, 500),
          horizontalSpacing: 25,
          verticalSpacing: 30,
          horizontalAlignment: "Left",
          verticalAlignment: "Top",
        }}
        dataSourceSettings={{
          id: "Name",
          parentId: "ReportingPerson",
          dataManager: items,
        }}
        getNodeDefaults={(obj: NodeModel) => {
          obj.shape = {
            type: "Text",
            content: (
              obj.data as {
                Name: "string";
              }
            ).Name,
          };
          obj.style = {
            fill: "None",
            strokeColor: "none",
            strokeWidth: 2,
            bold: true,
            color: "white",
          };
          obj.width = 50;
          obj.height = 40;
          obj.borderColor = "white";
          obj.backgroundColor = "#6BA5D7";
          obj.borderWidth = 1;
          (obj.shape as TextModel).margin = {
            left: 25,
            right: 25,
            top: 25,
            bottom: 25,
          };
          return obj;
        }}
        getConnectorDefaults={(connector: ConnectorModel, diagram: Diagram) =>
        {
          connector.style = {
            strokeColor: "#6BA5D7",
            strokeWidth: 2,

```

```

    };
    connector.targetDecorator.style.fill = "#6BA5D7";
    connector.targetDecorator.style.strokeColor = "#6BA5D7";
    connector.targetDecorator.shape = "None";
    connector.type = "Orthogonal";
    return connector;
  }}
  setNodeTemplate={function (obj: NodeModel, diagram: Diagram) {
    obj.borderColor = (
      obj.data as {
        color: "string";
      }
    ).color;
  }}
  >
  <Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Accessibility in React Diagram component

Diagram provides built-in compliance with the [WAI-ARIA](#) specifications. WAI-ARIA Accessibility supports are achieved through the attributes like [aria-label](#). It helps to provides information about elements in a document for assistive technology.

The accessibility compliance for the diagram component is outlined below.

Accessibility Criteria	Compatibility
-----	-----
WCAG 2.2 Support	
Section 508 Support	
Screen Reader Support	
Right-To-Left Support	
Color Contrast	
Mobile Device Support	
Keyboard Navigation Support	

```
| Accessibility Checker Validation |  |  
  
| Axe-core Accessibility Validation |  |  
  
<style>  
.post .post-content img {  
display: inline-block;  
margin: 0.5em 0;  
}  
</style>  
  
<div> - All  
features of the component meet the requirement.</div>  
  
<div> - Some features of the component do not meet the requirement.</div>  
  
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

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

| Attributes | Purpose |

| --- | --- |

| **aria-label** | Provides an accessible name for the Diagram Objects. |

Aria-label

Attribute provides the text label with some default description for below elements in diagram.

<!-- markdownlint-disable MD033 -->

Element	Default description
ResizeNorthWest	Thumb to resize the selected object on the top-left corner.
ResizeNorthEast	Thumb to resize the selected object on the top-right side direction.
ResizeSouthWest	Thumb to resize the selected object on the bottom-left side direction.
ResizeSouthEast	Thumb to resize the selected object on the bottom-right side direction.
ResizeNorth	Thumb to resize the selected object on the top side direction.
ResizeSouth	Thumb to resize the selected object on the bottom side direction.
ResizeWest	Thumb to resize the selected object on the left side direction.
ResizeEast	Thumb to resize the selected object on the right side direction.

ConnectorSourceThumb	Thumb to move the source point of the connector.
ConnectorTargetThumb	Thumb to move the target point of the connector.
RotateThumb	Thumb to rotate the selected object.

Mobile device support

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

Screen Reader Support

The Diagram component supports and its information was dictated properly by the screen readers based on the ARIA attributes and content.

Keyboard navigation support

Syncfusion Diagram component support keyboard navigation, allowing users who rely on alternate methods to effortlessly navigate and interact with the component.

Keyboard interaction

The Diagram component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Diagram component.

| **Command** | **Action** |

| --- | --- |

| Ctrl + A | Select All |

| Ctrl + X | Cut |

| Ctrl + C | Copy |

| Ctrl + V | Paste |

| Ctrl + Z | Undo |

| Ctrl + Y | Redo |

| Delete | Delete |

| Up Arrow | Move selected object to up |

| Down Arrow | Move selected object to down |

| Left Arrow | Move selected object to left |

| Right Arrow | Move selected object to right |

| Enter | Start Annotation Edit |

| Escape | End Annotation Edit |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Commands in React Diagram component

<!-- markdownlint-disable MD010 -->

There are several commands available in the diagram as follows.

- Alignment commands
- Spacing commands
- Sizing commands
- Clipboard commands
- Grouping commands
- Z-order commands
- Zoom commands
- Nudge commands
- FitToPage commands
- Undo/Redo commands

Align

Alignment commands enable you to align the selected or defined objects such as nodes and connectors with respect to the selection boundary. Refer to [align](#) commands which shows how to use align methods in the diagram.

<!-- markdownlint-disable MD033 -->

Parameters	Description
[:----- :-----:	
[[:Alignment	
Options`](https://ej2.syncfusion.com/react/documentation/api/diagram/alignmentOptions#AlignmentOptions)	

Defines the specific direction, with respect to which the objects to be aligned.

The accepted values of the argument "alignment options" are as follows.

Left Aligns all the selected objects at the left of the selection boundary.

Right Aligns all the selected objects at the right of the selection boundary.

Center Aligns all the selected objects at the center of the selection boundary.

Top Aligns all the selected objects at the top of the selection boundary.

Bottom Aligns all the selected objects at the bottom of the selection boundary.

Middle Aligns all the selected objects at the middle of the selection boundary.

|

| Objects | `<p align="left">`Defines the objects to be aligned. This is an optional parameter. By default, all the nodes and connectors in the selected region of the diagram gets aligned.`</p>` |

[`Alignment

Mode`](<https://ej2.syncfusion.com/react/documentation/api/diagram/alignmentMode#AlignmentMode>) |

Defines the specific mode, with respect to which the objects to be aligned. This is an optional parameter. The default alignment mode is `Object`.

The accepted values of the argument "alignment mode" are as follows.

Object Aligns the objects based on the first object in the selected list.

Selector Aligns the objects based on the selection boundary.

|

The following code example illustrates how to align all the selected objects at the left side of the selection boundary.

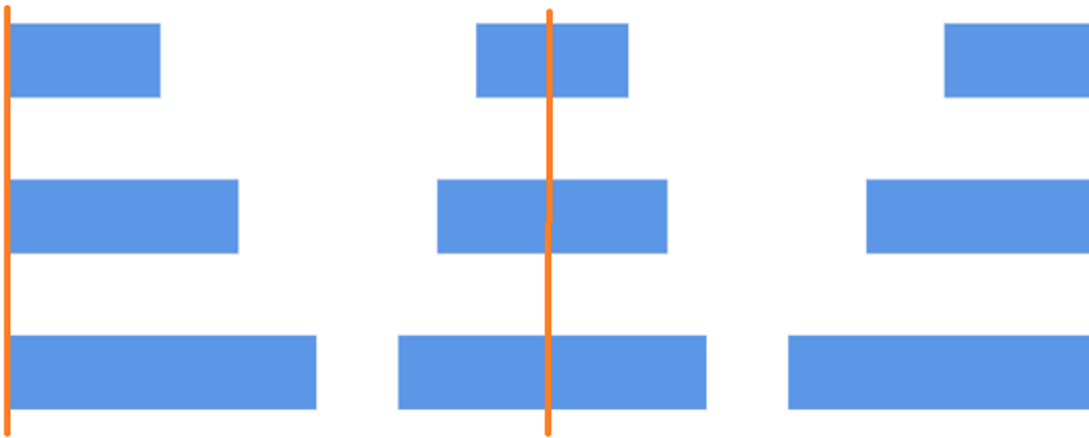
```
`ts
```

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the node
let node: NodeModel[] = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
```

```
strokeWidth: 1
},
}, {
id: 'node2',
width: 100,
height: 60,
offsetX: 100,
offsetY: 170,
style: {
fill: '#6BA5D7',
strokeColor: 'white',
strokeWidth: 1
},
}, {
id: 'node3',
width: 140,
height: 60,
offsetX: 100,
offsetY: 240,
style: {
fill: '#6BA5D7',
strokeColor: 'white',
strokeWidth: 1
},
}];
//Initializes the Diagram Component
function App() {
return (
<DiagramComponent
id="container"
ref={({diagram}) => (diagramInstance = diagram)}
width={'650px'}
height={'350px'}
```

```
nodes={node}
created={() => {
  let selArray = [];
  selArray.push(
    diagramInstance.nodes[0],
    diagramInstance.nodes[1],
    diagramInstance.nodes[2]
  );
  //Selects the nodes
  diagramInstance.select(selArray);
  //Sets direction as left
  diagramInstance.align(
    'Left',
    diagramInstance.selectedItems.nodes,
    'Selector'
  );
  diagramInstance.dataBind();
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Distribute

The [Distribute](#) commands enable to place the selected objects on the page at equal intervals from each other. The selected objects are equally spaced within the selection boundary.

The factor to distribute the shapes [DistributeOptions](#) are listed as follows:

- RightToLeft: Distributes the objects based on the distance between the right and left sides of the adjacent objects.
- Left: Distributes the objects based on the distance between the left sides of the adjacent objects.
- Right: Distributes the objects based on the distance between the right sides of the adjacent objects.
- Center: Distributes the objects based on the distance between the center of the adjacent objects.
- BottomToTop: Distributes the objects based on the distance between the bottom and top sides of the adjacent objects.
- Top: Distributes the objects based on the distance between the top sides of the adjacent objects.
- Bottom: Distributes the objects based on the distance between the bottom sides of the adjacent objects.
- Middle: Distributes the objects based on the distance between the vertical center of the adjacent objects.

The following code example illustrates how to execute the space commands.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
```

```
} from "@syncfusion/ej2-react-diagrams";  
let diagramInstance: DiagramComponent;  
//Initializes the node  
let node: NodeModel[] = [{  
  id: 'node1',  
  width: 90,  
  height: 60,  
  offsetX: 100,  
  offsetY: 100,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
}, {  
  id: 'node2',  
  width: 90,  
  height: 60,  
  offsetX: 240,  
  offsetY: 100,  
  style: {  
    fill: '#6BA5D7',  
    strokeColor: 'white',  
    strokeWidth: 1  
  },  
}, {  
  id: 'node3',  
  width: 90,  
  height: 60,  
  offsetX: 170,  
  offsetY: 150,  
  style: {  
    fill: '#6BA5D7',
```

```
strokeColor: 'white',
strokeWidth: 1
},
});
//Initializes the Diagram Component
function App() {
return (
<DiagramComponent
id="container"
ref={(diagram) => (diagramInstance = diagram)}
width={'650px'}
height={'350px'}
nodes={node}
created={() => {
let selArray: (NodeModel | ConnectorModel)[] = []; selArray.push(diagramInstance.nodes[0],
diagramInstance.nodes[1], diagramInstance.nodes[2]);
//Selects the nodes
diagramInstance.select(selArray);
//Distributes space between the nodes
diagramInstance.distribute('RightToLeft', diagramInstance.selectedItems.nodes);
}}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```



Sizing

Sizing [sameSize](#) commands enable to equally size the selected nodes with respect to the first selected object.

[SizingOptions](#) are as follows:

- Width: Scales the width of the selected objects.
- Height: Scales the height of the selected objects.
- Size: Scales the selected objects both vertically and horizontally.

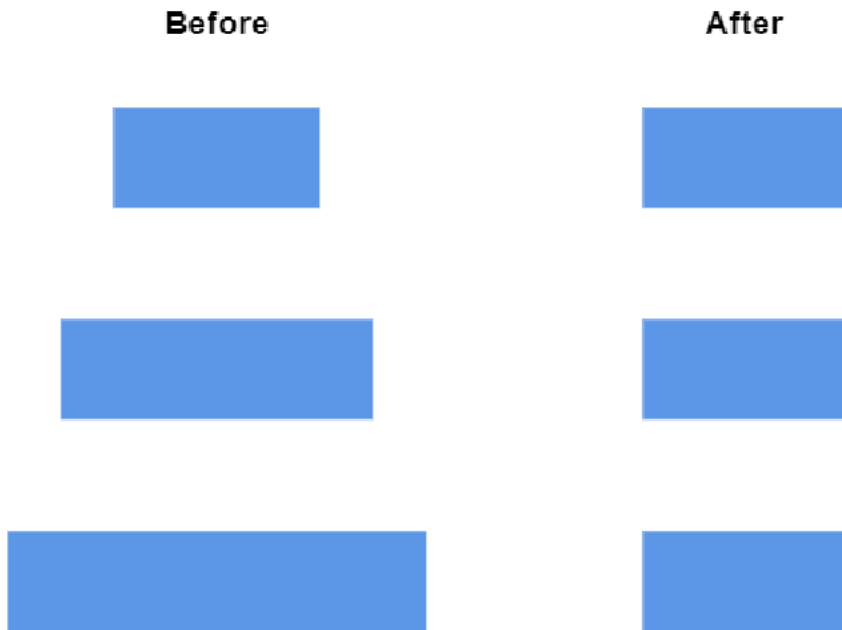
The following code example illustrates how to execute the size commands.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the nodes
let node: NodeModel[] = [{
```

```
id: 'node1',
width: 90,
height: 60,
offsetX: 100,
offsetY: 100,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white',
  strokeWidth: 1
},
}, {
id: 'node2',
width: 100,
height: 60,
offsetX: 100,
offsetY: 170,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white',
  strokeWidth: 1
},
}, {
id: 'node3',
width: 130,
height: 60,
offsetX: 100,
offsetY: 230,
style: {
  fill: '#6BA5D7',
  strokeColor: 'white',
  strokeWidth: 1
},
}];
```

```
//Initializes the Diagram Component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      nodes={node}
      created={() => {
        let selArray: (NodeModel)[] = []; selArray.push(diagramInstance.nodes[0], diagramInstance.nodes[1],
        diagramInstance.nodes[2]);
        //Selects the nodes
        diagramInstance.select(selArray);
        //Resizes the selected nodes with the same width
        diagramInstance.sameSize('Width', diagramInstance.selectedItems.nodes);
      }}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
,
```



Clipboard

Clipboard commands are used to cut, copy, or paste the selected elements. Refer to the following link which shows how to use clipboard methods in the diagram.

- Cuts the selected elements from the diagram to the diagram's clipboard, [cut](#).
- Copies the selected elements from the diagram to the diagram's clipboard, [copy](#).
- Pastes the diagram's clipboard data (nodes/connectors) into the diagram, [paste](#).

The following code illustrates how to execute the clipboard commands.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the connector
let connector = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2,
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
```

```

    }
  }
  });
  //Initializes the nodes
  let node = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
  }, {
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 240,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
  },
  ];
  //Initializes the Diagram component
  function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
      (diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
      connectors={connector} created={() => {
        diagramInstance.select([
          diagramInstance.nodes[0],
          diagramInstance.nodes[1],
          diagramInstance.connectors[0],
        ]);
        //copies the selected nodes
        diagramInstance.copy();
        //pastes the copied objects
        diagramInstance.paste(diagramInstance.copy());
      }}/>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,

```



```

    NodeModel
  } from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the connector
let connector: ConnectorModel[] = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
  },
  targetDecorator: {
    style: {
      fill : '#6BA5D7',
      strokeColor : '#6BA5D7'
    }
  }
}];
//Initializes the nodes
let node: NodeModel[] = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}, {
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 240,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}];
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      nodes={node}
      connectors={connector}
      created={() => {
        diagramInstance.select([
          diagramInstance.nodes[0],

```

```

        diagramInstance.nodes[1],
        diagramInstance.connectors[0],
    ]);
    //copies the selected nodes
    diagramInstance.copy();
    //pastes the copied objects
    diagramInstance.paste(diagramInstance.copy());
    }}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Grouping

Grouping commands are used to group/ungroup the selected elements on the diagram. Refer to the following link which shows how to use grouping commands in the diagram.

[Group](#) the selected nodes and connectors in the diagram.

[Ungroup](#) the selected nodes and connectors in the diagram.

The following code illustrates how to execute the grouping commands.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the nodes
let nodes = [{
  id: 'node1',
  width: 100,
  height: 70,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 100,
  height: 70,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
],

```

```

    {
      id: 'node3',
      width: 100,
      height: 70,
      offsetX: 200,
      offsetY: 200,
      style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
      },
    },
  ],
  {
    {
      id: 'group',
      children: ['node1', 'node2', 'connector1']
    },
    {
      id: 'group2',
      children: ['node3', 'group']
    }
  }
];
//Initializes the connector
let connector = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2,
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
}
]];
//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="container" ref={ (diagram) =>
    (diagramInstance = diagram) } width={ '650px' } height={ '350px' } nodes={nodes}
    connectors={connector} created={ () => {
      //Selects the diagram
      diagramInstance.selectAll();
      //Groups the selected elements.
      diagramInstance.group();
    } }/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```
{% raw %}
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the nodes
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 70,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node2',
  width: 100,
  height: 70,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'node3',
  width: 100,
  height: 70,
  offsetX: 200,
  offsetY: 200,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
{
  id: 'group',
  children: ['node1', 'node2', 'connector1']
},
{
  id: 'group2',
  children: ['node3', 'group']
}
];
//Initializes the connector
let connector: ConnectorModel[] = [{

```

```

    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    style: {
      strokeColor : '#6BA5D7',
      fill: '#6BA5D7',
      strokeWidth : 2,
    },
    targetDecorator: {
      style: {
        fill : '#6BA5D7',
        strokeColor : '#6BA5D7'
      }
    }
  }
}
});
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      nodes={nodes}
      connectors={connector}
      created={() => {
        //Selects the diagram
        diagramInstance.selectAll();
        //Groups the selected elements.
        diagramInstance.group();
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Z-Order command

Z-Order commands enable you to visually arrange the selected objects such as nodes and connectors on the page.

[bringToFront command](#)

The [bringToFront](#) command visually brings the selected element to front over all the other overlapped elements. The following code illustrates how to execute the [bringToFront](#) command.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the nodes
let node = [{
  id: 'node1',

```

```

        width: 90,
        height: 60,
        offsetX: 100,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
    }, {
        id: 'node2',
        width: 90,
        height: 60,
        offsetX: 240,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
    }, {
        id: 'node3',
        width: 90,
        height: 60,
        offsetX: 160,
        offsetY: 90,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
    },
    ]];
let selArray = [];
//Initializes the Diagram component
function App() {
    return (<DiagramComponent id="diagram1" ref={(diagram) =>
(diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
created={() => {
        selArray.push(diagramInstance.nodes[2]);
        //Selects the nodes
        diagramInstance.select(selArray);
        //Brings to front
        diagramInstance.bringToFront();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,

```

```

    DiagramComponent,
    ConnectorModel,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the nodes
let node: NodeModel[] = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 240,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node3',
    width: 90,
    height: 60,
    offsetX: 160,
    offsetY: 90,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
},
];
let selArray: (NodeModel)[] = [];
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="diagram1"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'650px'}
            height={'350px'}
            nodes={node}
            created={() => {
                selArray.push(diagramInstance.nodes[2]);
                //Selects the nodes
                diagramInstance.select(selArray);
                //Brings to front
                diagramInstance.bringToFront();
            }}

```

```

    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

sendToBack command

The [sendToBack](#) command visually moves the selected element behind all the other overlapped elements. The following code illustrates how to execute the `sendToBack` command.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the nodes
let node = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}, {
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 240,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}, {
  id: 'node3',
  width: 90,
  height: 60,
  offsetX: 160,
  offsetY: 90,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
];
let selArray = [];
//Initializes the Diagram component
function App() {

```



```

    return (<DiagramComponent id="diagram1" ref={(diagram) =>
(diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
created={() => {
    selArray.push(diagramInstance.nodes[2]);
    //Selects the nodes
    diagramInstance.select(selArray);
    //Brings to front
    diagramInstance.sendToBack();
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the nodes
let node: NodeModel[] = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 240,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node3',
    width: 90,
    height: 60,
    offsetX: 160,
    offsetY: 90,
    style: {
        fill: '#6BA5D7',

```

```

        strokeColor: 'white',
        strokeWidth: 1
    },
  ]];
let selArray: (NodeModel)[] = [];
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="diagram1"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      nodes={node}
      created={() => {
        selArray.push(diagramInstance.nodes[2]);
        //Selects the nodes
        diagramInstance.select(selArray);
        //Brings to front
        diagramInstance.sendToBack();
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

moveForward command

The [moveForward](#) command visually moves the selected element over the nearest overlapping element. The following code illustrates how to execute the `moveForward` command.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let node = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}, {
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 180,
  offsetY: 100,

```

```

        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
    ]];
let selArray = [];
//Initializes the Diagram component
function App() {
    return (<DiagramComponent id="diagram1" ref={ (diagram) =>
    (diagramInstance = diagram) } width={ '650px' } height={ '350px' } nodes={node}
    created={ () => {
        selArray.push(diagramInstance.nodes[2]);
        //Selects the nodes
        diagramInstance.select(selArray);
        //Brings to front
        diagramInstance.moveForward();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let node: NodeModel[] = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 180,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
},

```

```

    }];
    let selArray: (NodeModel)[] = [];
    //Initializes the Diagram component
    function App() {
        return (
            <DiagramComponent
                id="diagram1"
                ref={(diagram) => (diagramInstance = diagram)}
                width={'650px'}
                height={'350px'}
                nodes={node}
                created={() => {
                    selArray.push(diagramInstance.nodes[2]);
                    //Selects the nodes
                    diagramInstance.select(selArray);
                    //Brings to front
                    diagramInstance.moveForward();
                }}
            />
        );
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);
    {% endraw %}

```

sendBackward command

The [sendBackward](#) command visually moves the selected element behind the underlying element. The following code illustrates how to execute the `sendBackward` command.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let node = [{
    id: 'node1',
    width: 90,
    height: 60,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
}, {
    id: 'node2',
    width: 90,
    height: 60,
    offsetX: 180,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',

```

```

        strokeWidth: 1
      },
    ]];
let selArray = [];
//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="diagram1" ref={(diagram) =>
    (diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
    created={() => {
      selArray.push(diagramInstance.nodes[2]);
      //Selects the nodes
      diagramInstance.select(selArray);
      //Brings to front
      diagramInstance.sendBackward();
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let node: NodeModel[] = [{
  id: 'node1',
  width: 90,
  height: 60,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
}, {
  id: 'node2',
  width: 90,
  height: 60,
  offsetX: 180,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
},
];
let selArray: (NodeModel)[] = [];

```

```

//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="diagram1"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      nodes={node}
      created={() => {
        selArray.push(diagramInstance.nodes[2]);
        //Selects the nodes
        diagramInstance.select(selArray);
        //Brings to front
        diagramInstance.sendBackward();
      }}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Zoom

The [zoom](#) command is used to zoom-in and zoom-out the diagram view.

The following code illustrates how to zoom-in/zoom out the diagram.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      created={() => {

```

```
// Sets the zoomFactor
//Defines the focusPoint to zoom the Diagram with respect to any point
//When you do not set focus point, zooming is performed with reference to the center of current
Diagram view.
diagramInstance.zoom(1.2, {
x: 100,
y: 100,
});
}}
/>
);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Nudge command

The [nudge](#) commands move the selected elements towards up, down, left, or right by 1 pixel.

[NudgeDirection](#) nudge command moves the selected elements towards the specified direction by 1 pixel, by default.

The accepted values of the argument "direction" are as follows:

- Up: Moves the selected elements towards up by the specified delta value.
- Down: Moves the selected elements towards down by the specified delta value.
- Left: Moves the selected elements towards left by the specified delta value.
- Right: Moves the selected elements towards right by the specified delta value.

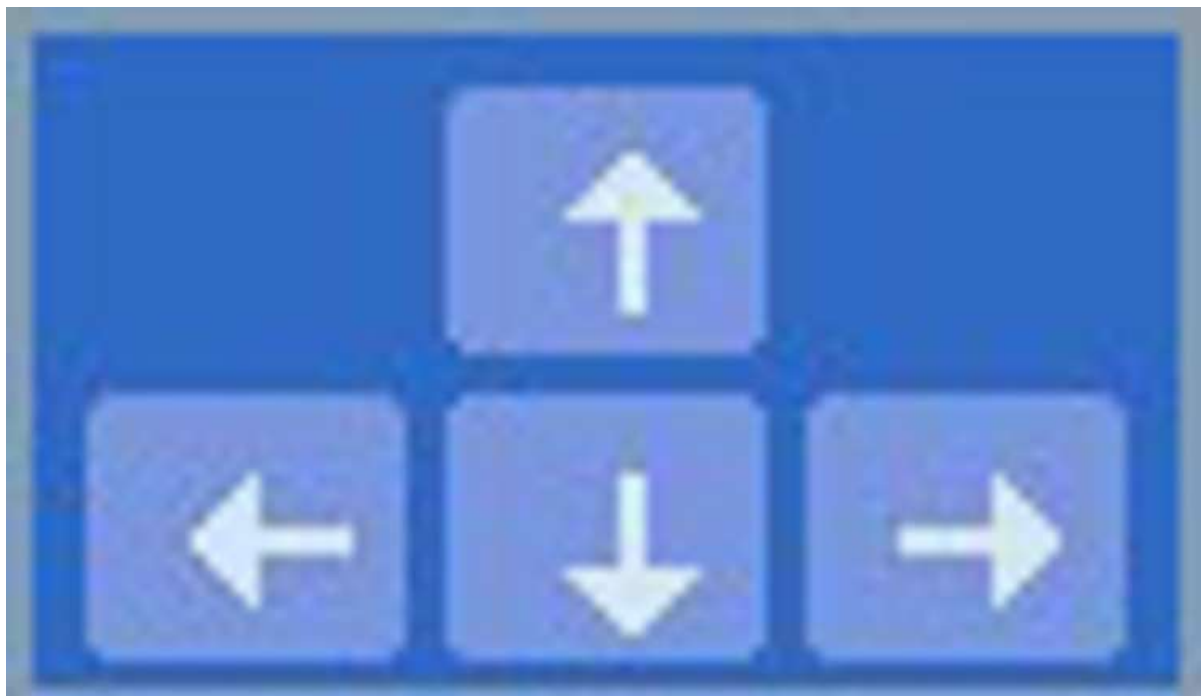
The following code illustrates how to execute nudge command.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
Diagram,
DiagramComponent,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the Diagram component
```

```
function App() {  
  return (  
    <DiagramComponent  
      id="container"  
      ref={(diagram) => (diagramInstance = diagram)}  
      width={'650px'}  
      height={'350px'}  
      created={() => {  
        //Nudges to right  
        diagramInstance.nudge('Right');  
      }}  
    />  
  );  
}  
  
const root = ReactDOM.createRoot(document.getElementById('diagram'));  
root.render(<App />);  
`
```

Nudge by using arrow keys

The corresponding arrow keys are used to move the selected elements towards up, down, left, or right direction by 1 pixel.



Nudge commands are particularly useful for accurate placement of elements.

[BringIntoView](#)

The [bringIntoView](#) command brings the specified rectangular region into the viewport of the diagram.

The following code illustrates how to execute the `bringIntoView` command.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Rect
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={{(diagram) => (diagramInstance = diagram)}}
      width={'650px'}
      height={'350px'}
      created={() => {
        //Brings the specified rectangular region of the Diagram content to the viewport of the page.
        let bound: Rect = new Rect(200, 400, 500, 400);
        diagramInstance.bringIntoView(bound);
      }}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

BringToCenter

The [bringToCenter](#) command brings the specified rectangular region of the diagram content to the center of the viewport.

The following code illustrates how to execute the `bringToCenter` command.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Rect
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      created={() => {
        //Brings the specified rectangular region of the Diagram content to the viewport of the page.
        let bound: Rect = new Rect(200, 400, 500, 400);
        diagramInstance.bringToCenter(bound);
      }}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

FitToPage command

The [fitToPage](#) command helps to fit the diagram content into the view with respect to either width, height, or at the whole.

The [mode](#) parameter defines whether the diagram has to be horizontally/vertically fit into the viewport with respect to width, height, or entire bounds of the diagram.

The [region](#) parameter defines the region that has to be drawn as an image.

The [margin](#) parameter defines the region/bounds of the diagram content that is to be fit into the view.

The [canZoomIn](#) parameter enables/disables zooming to fit the smaller content into a larger viewport.

The [customBounds](#) parameter the custom region that has to be fit into the viewport.

The following code illustrates how to execute `FitToPage` command.

```
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'650px'}
      height={'350px'}
      created={() => {
        //fit the diagram to the page with respect to mode and region
        diagramInstance.fitToPage({
          mode: 'Page',
          region: 'Content',
          margin: {
            bottom: 50
          }
        }
      }
    ),
```

```

canZoomIn: false
});
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Command manager

Diagram provides support to map/bind command execution with desired combination of key gestures. Diagram provides some built-in commands.

[CommandManager](#) provides support to define custom commands. The custom commands are executed, when the specified key gesture is recognized.

Custom command

To define a custom command, specify the following properties:

- [execute](#): A method to be executed.
- [canExecute](#): A method to define whether the command can be executed at the moment.
- [gesture](#): A combination of [keys](#) and [KeyModifiers](#).
- [parameter](#): Defines any additional parameters that are required at runtime.
- [name](#): Defines the name of the command.

To explore the properties of custom commands, refer to [Commands](#).

The following code example illustrates how to define a custom command.

```

`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import {
  Diagram,
  DiagramComponent,
  CommandManagerModel,
  Keys,
  KeyModifiers
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Custom command for Diagraming elements.

```

```
public getCommandManagerSettings(): CommandManagerModel {
  let commandManager: CommandManagerModel = {
    commands: [{
      name: 'customCopy',
      parameter: 'node',
      //Method to define whether the command can be executed at the current moment
      canExecute: function() {
        //Defines that the clone command can be executed, if and only if the selection list is not empty.
        if (diagramInstance.selectedItems.nodes.length > 0 || diagramInstance.selectedItems.connectors.length > 0) {
          return true;
        }
        return false;
      },
      //Command handler
      execute: function() {
        //Logic to clone the selected element
        diagramInstance.copy();
        diagramInstance.paste();
        diagramInstance.dataBind();
      },
      //Defines that the clone command has to be executed on the recognition of key press.
      gesture: {
        key: Keys.G,
        keyModifiers: KeyModifiers.Shift | KeyModifiers.Alt
      }
    }
  ]
};
return commandManager;
}

//Initializes the Diagram component
function App() {
```

```

return (
  <DiagramComponent
    id="container"
    ref={(diagram) => (diagramInstance = diagram)}
    width={'650px'}
    height={'350px'}
    commandManager={this.getCommandManagerSettings()}
  />
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Modify the existing command

When any one of the default commands is not desired, they can be disabled. To change the functionality of a specific command, the command can be completely modified.

The following code example illustrates how to disable a command and how to modify the built-in commands.

```

{% raw %}
`ts
import * as React from "react";
import * as ReactDOM from "react-dom";

import {
  Diagram,
  DiagramComponent,
  CommandManagerModel,
  Keys,
  KeyModifiers
} from "@syncfusion/ej2-react-diagrams";

let diagramInstance: DiagramComponent;

//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent

```

```
id="container"
ref={({diagram}) => (diagramInstance = diagram)}
width={'650px'}
height={'350px'}
//Disables the nudging commands
commandManager={{
  commands: {
    //Assigns null value to an existing command and disables its execution
    nudgeUp: null,
    nudgeDown: null,
    nudgeRight: null,
    //Modifies the existing command - nudgeLeft
    nudgeLeft: {
      canExecute: function (args) {
        if (args.model.selectedItems.length) {
          return true;
        }
      },
    },
  },
  //Command handler
  execute: function (args) {
    diagramInstance.nudge('left');
  },
  gesture: {
    key: Keys.Left,
  },
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

{% endraw %}

See Also

- [How to create the custom context menu items](#)

Undo redo in React Diagram component

Diagram tracks the history of actions that are performed after initializing the diagram and provides support to reverse and restore those changes.

Undo and redo

Diagram provides built-in support to track the changes that are made through interaction and through public APIs. The changes can be reverted or restored either through shortcut keys or through commands.

Note: If you want to use Undo-Redo in diagram, you need to inject UndoRedo in the diagram.

Undo/redo through shortcut keys

Undo/redo commands can be executed through shortcut keys. Shortcut key for undo is Ctrl+z and shortcut key for redo is Ctrl+y.

Undo/redo through public APIs

The client-side methods [undo](#) and [redo](#) help you to revert/restore the changes. The following code example illustrates how to undo/redo the changes through script.

The following code example illustrates how to undo/redo the changes through script.

```
`ts
// initialize Diagram component
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={({diagram}) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
// Reverts the last action performed
```



```

diagramInstance.undo();
// Restores the last undone action
diagramInstance.redo();
`

```

When a change in the diagram is reverted or restored (undo/redo), the historyChange event gets triggered.

Group multiple changes

History list allows to revert or restore multiple changes through a single undo/redo command. For example, revert/restore the fill color change of multiple elements at a time.

The client-side method [startGroupAction](#) is used to notify the diagram to start grouping the changes. The client-side method [endGroupAction](#) is used to notify to stop grouping the changes. The following code illustrates how to undo/redo fillColor change of multiple elements at a time.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, UndoRedo, Inject } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'label1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
created={() => {
        //Start to group the changes
        diagramInstance.startGroupAction();
        //Makes the changes
        let color = ['black', 'red', 'green', 'yellow'];
        for (var i = 0; i < color.length; i++) {
            // Updates the fillColor for all the child elements.
            diagramInstance.nodes[0].style.fill = color[i];
            diagramInstance.dataBind();
        }
        //Ends grouping the changes
        diagramInstance.endGroupAction();
    }} getNodeDefaults={(node) => {

```

```

        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
    }>
    <Inject services={[UndoRedo]}/>
  </DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  ConnectorModel,
  UndoRedo,
  Inject
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      created={() => {
        //Start to group the changes
        diagramInstance.startGroupAction();
        //Makes the changes
        let color = ['black', 'red', 'green', 'yellow'];
        for (var i = 0; i < color.length; i++) {

```

```

        // Updates the fillColor for all the child elements.
        diagramInstance.nodes[0].style.fill = color[i];
        diagramInstance.dataBind();
    }
    //Ends grouping the changes
    diagramInstance.endGroupAction();
  }}
  getNodeDefaults=(node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }}
  >
  <Inject services={[UndoRedo]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Track custom changes

Diagram provides options to track the changes that are made to custom properties. For example, in case of an employee relationship diagram, track the changes in the employee information. The historyManager of the diagram enables you to track such changes. The following example illustrates how to track such custom property changes.

Before changing the employee information, save the existing information to historyManager by using the client-side method push of historyManager. The historyManager canLog method can be used which takes a history entry as argument and returns whether the specific entry can be added or not.

The following code example illustrates how to save the existing property values.

```

`ts
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}

      <Inject services={[UndoRedo]} />
    </DiagramComponent>
  );
}

```

```

);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

//Creates a custom entry
let entry: HistoryEntry = {
  undoObject: diagramInstance.nodes[0];
};

// adds that to history list
diagramInstance.historyManager.push(entry);

diagramInstance.dataBind();
`

```

canLog

canLog in the history list, which takes a history entry as argument and returns whether the specific entry can be added or not.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, UndoRedo, Inject } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
created={() => {
  // canLog decide whether the entry add or not in history List
  diagramInstance.historyList.canLog = function (entry) {
    entry.cancel = true;
    return entry;
  };
}});

```

```

    }} getNodeDefaults=(node) => {
      node.height = 100;
      node.width = 100;
      node.style.fill = '#6BA5D7';
      node.style.strokeColor = 'white';
      return node;
    }}>
    <Inject services={[UndoRedo]}/>
  </DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  ConnectorModel,
  UndoRedo,
  Inject,
  HistoryEntry
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
  id: 'Start',
  width: 140,
  height: 50,
  offsetX: 300,
  offsetY: 50,
  annotations: [{
    id: 'label1',
    content: 'Start'
  }],
  shape: {
    type: 'Flow',
    shape: 'Terminator'
  }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      created={() => {
        // canLog decide whether the entry add or not in history List
        diagramInstance.historyList.canLog = function (entry: HistoryEntry)
      }
    >

```

```

        entry.cancel = true;
        return entry;
    };
  }}
  getNodeDefaults=(node: NodeModel) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = 'white';
    return node;
  }}
  >
  <Inject services={[UndoRedo]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Track undo/redo actions

The historyManager undoStack property is used to get the collection of undo actions which should be performed in the diagram.

The undoStack/redoStack is the read-only property.

`ts

```
let diagramInstance: DiagramComponent;
```

```
function App() {
```

```
  return (
```

```
    <DiagramComponent
```

```
      id="container"
```

```
      ref={(diagram) => (diagramInstance = diagram)}
```

```
      width={'100%'}
```

```
      height={'600px'}
```

```
      nodes={nodes}
```

```
    <Inject services={[UndoRedo]} />
```

```
  </DiagramComponent>
```

```
);
```

```
}
```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
```

```
root.render(<App />);
```

```
//get the collection of undoStack objects
```

```
let undoStack = diagramInstance.historyManager.undoStack;
//get the collection of redoStack objects
let redoStack = diagramInstance.historyManager.redoStack;
`
```

History change event

The [historyChange](#) event triggers, whenever the interaction of the node and connector is take place.

```
`ts
```

```
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={({diagram}) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}

      <Inject services={[UndoRedo]} />
    </DiagramComponent>
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
// history change event
diagramInstance.historyChange = (arg: IHistoryChangeArgs) => {
  //causes of history change
  let cause: string = arg.cause;
}
`
```

Stack Limit

The [stackLimit](#) property of history manager is used to limits the number of actions to be stored on the history manager.

INDEX.JSX

```
{% raw %}
```

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, UndoRedo, Inject } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let nodes = [{
    id: 'Start',
    width: 140,
    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
        id: 'label1',
        content: 'Start'
    }],
    shape: {
        type: 'Flow',
        shape: 'Terminator'
    }
}];
function App() {
    return (<DiagramComponent id="container" ref={(diagram) =>
(diagramInstance = diagram)} width={'100%'} height={'600px'} nodes={nodes}
created={() => {
        diagramInstance.historyManager = { stackLimit: 3 };
        getDefaults=(node) => {
            node.height = 100;
            node.width = 100;
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = 'white';
            return node;
        }
    })>
    <Inject services={[UndoRedo]} />
    </DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    ConnectorModel,
    UndoRedo,
    Inject
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let nodes: NodeModel[] = [{
    id: 'Start',
    width: 140,

```



```

    height: 50,
    offsetX: 300,
    offsetY: 50,
    annotations: [{
      id: 'label1',
      content: 'Start'
    }],
    shape: {
      type: 'Flow',
      shape: 'Terminator'
    }
  }];
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      created={() => {
        diagramInstance.historyManager = { stackLimit: 3 };
      }}
      getNodeDefaults={(node: NodeModel) => {
        node.height = 100;
        node.width = 100;
        node.style.fill = '#6BA5D7';
        node.style.strokeColor = 'white';
        return node;
      }}
    >
    <Inject services={[UndoRedo]} />
    </DiagramComponent>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Retain Selection

You can retain a selection at undo/redo operation by using the client-side API Method called **updateSelection**. Using this method, we can select a diagram objects.

`ts

```

let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}

```

```

width={'100%'}
height={'600px'}
nodes={nodes}

<Inject services={[UndoRedo]} />
</DiagramComponent>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
// Update the previous selection
diagramInstance.updateSelection: (object: NodeModel, diagram: Diagram) => {
let selArr = [];
selArr.push(object)
diagram.select(selArr);
}
`

```

Virtualization in React Diagram component

Virtualization in Diagram

Virtualization is the process of loading the diagramming objects available in the visible area of the Diagram control, that is, only the diagramming objects that lie within the ViewPort of the Scroll Viewer are loaded (remaining objects are loaded only when they come into view).

This feature gives an optimized performance while loading and dragging items to the Diagram that consists of many Nodes and Connectors.

The following code illustrates how to enable Virtualization mode in the diagram.

```

`ts
function App() {
return (
<DiagramComponent
id="container"
width={700}
height={600}
//Enable virtualization in diagram
constraints={

```

```
DiagramConstraints.Default | DiagramConstraints.Virtualization
```

```
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`
```

Serialization in React Diagram component

Serialization is the process of saving and loading for state persistence of the diagram.

Save

The diagram is serialized as string while saving. The client-side method, [saveDiagram](#) helps to serialize the diagram as a string. The following code illustrates how to save the diagram.

```
`ts
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
let saveData: string;
//returns serialized string of the Diagram
saveData = diagram.saveDiagram();
`
```

This string can be converted to JSON data and stored for future use. The following snippet illustrates how to save the serialized string into local storage.

```
`ts
//Saves the string in to local storage
localStorage.setItem('fileName', saveData);
saveData = localStorage.getItem('fileName');
`
```

Diagram can also be saved as raster or vector image files. For more information about saving the diagram as images, refer to [Print and Export](#).

Load

Diagram is loaded from the serialized string data by client-side method, [loadDiagram](#).

The following code illustrates how to load the diagram from serialized string data.

```
`ts
let diagramElement = document.getElementById('element');
let diagram: Object[] = diagramElement.ej2_instances[0];
```

```
//Loads the diagram from saved json data
diagram.loadDiagram(saveData);
`
```

Note: Before loading a new diagram, existing diagram is cleared.

Prevent Default Values

The diagram provides supports to simplifying the saved JSON object without adding the default properties that are presented in the diagram.

The following code illustrates how to simplify the JSON object.

```
`ts
let diagram: Diagram = new Diagram({
  serializationSettings: { preventDefaults: true },
});
`
```

Export in React Diagram component

Diagram provides support to export its content as image/svg files. The client-side method [exportDiagram](#) helps to export the diagram. The following code illustrates how to export the diagram as image.

Note: To use Print and Export, you need to inject `PrintAndExport` in the diagram.

```
<!-- markdownlint-disable MD033 -->
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

let options: IExportOptions = {};
options.mode = 'Download';
diagram.exportDiagram(options);
`
```

Exporting options

Diagram provides support to export the desired region of the diagram to desired formats.

File Name

[FileName](#) is the name of the file to be downloaded. By default, the file name is set to **Diagram**.

Format

[Format](#) is to specify the type/format of the exported file. By default, the diagram is exported as .jpg format. You can export diagram to the following formats:

- JPG
- PNG
- BMP
- SVG

```
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
let options: IExportOptions = {};
options.mode = 'Download';
options.format = 'SVG';
diagram.exportDiagram(options);
`
```

Margin

[Margin](#) specifies the amount of space that has to be left around the diagram.

<!-- markdownlint-disable MD033 -->

```
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
let options: IExportOptions = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
diagram.exportDiagram(options);
`
```

Mode

[Mode](#) specifies whether the diagram will be exported as files or get base64 data (ImageURL/SVG). The export options are as follows:

- Download: Exports and downloads the diagram as image/SVG.
- Data: return a base64 string.

The following code example illustrates how to export the diagram as raw data.

```
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

let options: IExportOptions = {};
options.mode = 'Data';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
let base64data = diagram.exportDiagram(options);
`
```

Region

You can export any particular [region](#) of the diagram and it is categorized into three types as follows.

- PageSettings
- Content
- CustomBounds

PageSettings

Diagram is exported based on the given PageSettings width and height. The Properties available in page settings are as follows.

- width
- height
- margin
- orientation
- boundaryConstraints
- background
- multiplePage
- showPageBreaks
- fitOptions

boundaryConstraints

Defines the editable region of the diagram.

- Infinity - Allow the interactions to take place at infinite height and width.
- Diagram - Allow the interactions to take place around the diagram's height and width.
- Page - Allow the interactions to take place around the page's height and width.

multiplePage

While setting multiple pages as false, the diagram is exported as a single image and while setting multiple pages as true, the diagram is exported as a separate image based on width and height.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`ts
function App() {
return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
let options: IExportOptions = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
options.format = 'SVG';
options.region = 'PageSettings';
diagram.exportDiagram(options);
`
```

Content

The diagram content alone will be exported as an image.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`javascript
var diagram = new ej.diagrams.Diagram({
width: 1500, height: 1500
}, '#element');
var options = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'format';
```

```
options.format = 'SVG';
options.region = 'Content';
diagram.exportDiagram(options);
`
```

Custom bounds

Diagram provides support to export any specific region of the diagram by using [bounds](#).

The following code example illustrates how to export the region occupied by the diagram elements.

```
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

let options: IExportOptions = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'CustomBounds';
options.bounds.x = 10;
options.bounds.y = 10;
options.bounds.height = 100;
options.bounds.width = 100;
diagram.exportDiagram(options);
`
```

Export diagram with stretch option

Diagram provides support to export the diagram as image for [stretch](#) option. The exported images will be clearer but larger in file size.

The following code example illustrates how to export the region occupied by the diagram elements.

```
`ts
function App() {
  return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
```



```

root.render(<App />);
let options: IExportOptions = {};
options.mode = 'Download';
options.margin = { left: 10, right: 10, top: 10, bottom: 10};
options.fileName = 'region';
options.format = 'SVG';
options.region = 'Content';
options.stretch = 'Stretch';
diagram.exportDiagram(options);
`

```

Print

The client-side method [print](#) helps to print the diagram as image.

Name	Type	Description
region	enum	Sets the region of the diagram to be printed.
bounds	object	Prints any custom region of diagram.
stretch	enum	Resizes the diagram content to fill its allocated space and printed.
multiplePage	boolean	Prints the diagram into multiple pages.
pageWidth	number	Sets the page width of the diagram while printing the diagram into multiple pages.
pageHeight	number	Sets the page height of the diagram while printing the diagram into multiple pages.
pageOrientation	enum	Sets the orientation of the page.

The following code example illustrates how to export the region occupied by the diagram elements.

```

`ts
function App() {
return <DiagramComponent id="container" width={'1500'} height={'1500'} />;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
let options: IExportOptions = {};
options.mode = 'Download';
options.region = 'PageSettings';
options.multiplePage = true;

```

```
options.pageHeight = 300;  
options.pageWidth = 300;  
diagram.print(options);  
,
```

Limitations

We have a limitation in exporting the image with HTML and Native node. So, Syncfusion Essential PDF library is used, which supports HTML Content to Image conversion by using the advanced Qt WebKit rendering engine. You can refer to the following KB link for more details.

[<https://www.syncfusion.com/kb/13298/how-to-print-or-export-the-html-and-native-node-into-image-format>]

Tooltip in React Diagram component

<!-- markdownlint-disable MD010 -->

In Graphical User Interface (GUI), the tooltip is a message that is displayed when mouse hovers over an element. The diagram provides tooltip support while dragging, resizing, rotating a node, and when the mouse hovers any diagram element.

Default tooltip

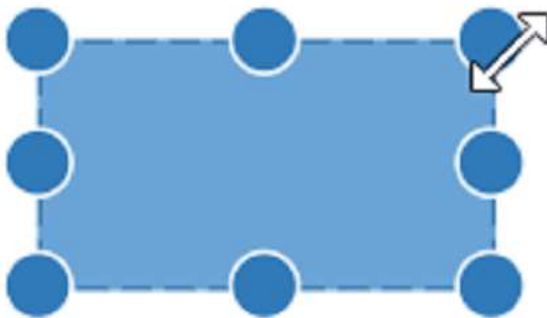
By default, diagram displays a tooltip to provide the size, position, and angle related information while dragging, resizing, and rotating. The following images illustrate how the diagram displays the node information during an interaction.

| Drag | Resize | Rotate |

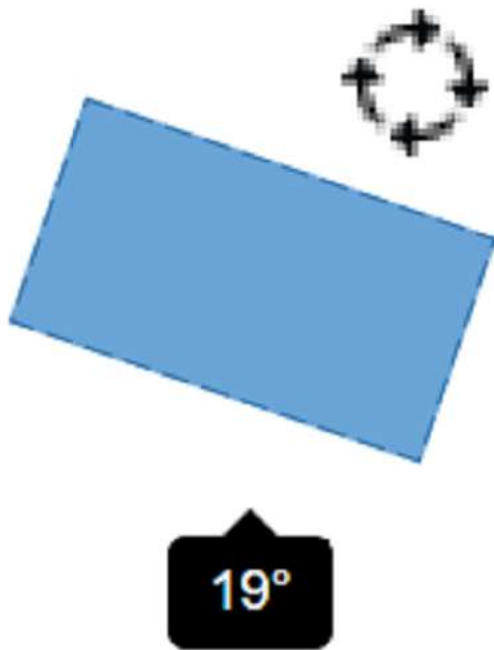
|---|---|---|



X:123 Y:48



W:94 H:51



Common tooltip for all nodes and connectors

The diagram provides support to show tooltip when the mouse hovers over any node/connector.

To show tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#) as shown in the following example.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white',
    },
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
```

```

        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
    },
    constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
  }];
//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="container" width={'650px'}
    height={'350px'} constraints={DiagramConstraints.Default |
    DiagramConstraints.Tooltip}
    //Defines nodes
    nodes={node}
    //Defines mouse over tooltip
    tooltip={{
      content: 'Nodes',
      position: 'TopLeft',
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  DiagramConstraints,
  NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import { TooltipComponent, TooltipEventArgs } from '@syncfusion/ej2-react-
popups';
import {
  NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white',
    },
  }],
  offsetX: 200,

```

```

        offsetY: 200,
        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7',
        },
        constraints: NodeConstraints.Default |
NodeConstraints.Tooltip,
    }];
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'650px'}
            height={'350px'}
            constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
            //Defines nodes
            nodes={node}
            //Defines mouse over tooltip
            tooltip={{
                content: 'Nodes',
                position: 'TopLeft',
            }}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Disable tooltip at runtime

The tooltip on mouse over can be disabled by assigning the [tooltip](#) property as `null`. The following code example illustrates how to disable the mouse over tooltip at runtime.

```

`ts
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'650px'}
            height={'350px'}
            constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
            //Defines nodes
            nodes={node}
            //Disables mouse over tooltip at runtime
            tooltip = {

```

```

null
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Tooltip for a specific node/connector

The tooltip can be customized for each node and connector. Remove the **InheritTooltip** option from the [constraints](#) of that node/connector. The following code example illustrates how to customize the tooltip for individual elements.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    //Sets the content of the Tooltip
    content: 'Node1',
    //Sets the position of the Tooltip
    position: 'BottomRight',
    //Sets the tooltip position relative to the node
    relativeMode: 'Object'
  },
},

```

```

    }];
    //Initializes the Diagram component
    function App() {
        return (<DiagramComponent id="container" width={'650px'}
            height={'350px'} constraints={DiagramConstraints.Default |
            DiagramConstraints.Tooltip}
            //Defines nodes
            nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    DiagramComponent,
    DiagramConstraints,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        },
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    constraints: NodeConstraints.Default |
    NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
        //Sets the content of the Tooltip
        content: 'Node1',
        //Sets the position of the Tooltip
        position: 'BottomRight',
    }

```



```

        //Sets the tooltip position relative to the node
        relativeMode: 'Object'
    },
    ]];
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={ '650px' }
            height={ '350px' }
            constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
            //Defines nodes
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Tooltip for Ports

The tooltip feature has been implemented to support Ports, providing the ability to display information or descriptions when the mouse hovers over them.

To display tooltips on mouseover, set the desired tooltip [content](#) by utilizing the [tooltip](#) property.

Tooltips for Ports can be enabled or disabled using the [PortConstraints](#) Tooltip property.

```

`js
let ports: [{
  offset: {x: 1,y: 0.5},
  tooltip: {content: 'Port Tootip'},
  //enable Port Tooltip Constraints
  constraints: PortConstraints.Default | PortConstraints.ToolTip,
  //disable Port Tooltip Constraints
  constraints: PortConstraints.Default ~& PortConstraints.ToolTip
}]
`

```

- Dynamic modification of tooltip content is supported, allowing you to change the displayed tooltip content during runtime.

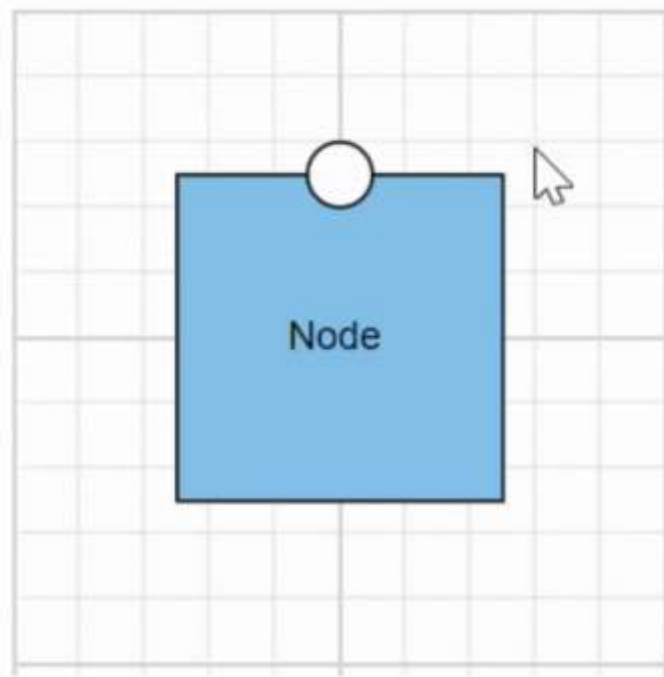
```

`js
{
  //change tooltip content at run time
  diagram.nodes[0].ports[0].tooltip.content = 'New Tooltip Content';
}

```

```
diagram.databind;  
}  
`
```

The following image illustrates how the diagram displays tooltips during an interaction with ports:



Here, the code provided below demonstrates the port tooltip Interaction.

INDEX.JSX

```
import * as React from "react";  
import * as ReactDOM from "react-dom";  
import { DiagramComponent, DiagramConstraints,  
NodeConstraints, PortConstraints, PortVisibility } from "@syncfusion/ej2-  
react-diagrams";  
// A node is created and stored in nodes array.  
let node = [{  
  id: "node1",  
  width: 100,  
  height: 100,  
  offsetX: 200,  
  offsetY: 200,  
  style: {  
    strokeColor: '#6BA5D7',  
    fill: '#6BA5D7'  
  },  
  ports: [{  
    offset: {  
      x: 0.5,  
      y: 0  
    },  
    visibility: PortVisibility.Visible,  
  }],  
}],
```

```

        //Set the style for the port
        style: {
            fill: '#FFFFFF',
            strokeWidth: 1,
            strokeColor: 'black'
        },
        //Defines mouse over tooltip for a node
        tooltip: {
            //Sets the content of the Tooltip
            content: 'Port Tooltip',
            //Sets the tooltip position relative to the node
            relativeMode: 'Object'
        },
        constraints: PortConstraints.Default | PortConstraints.Tooltip
    }
}
});
//Initializes the Diagram component
function App() {
    return (<DiagramComponent id="container" width={'650px'}
height={'350px'}
    //Defines nodes
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    DiagramComponent,
    DiagramConstraints,
    NodeConstraints,
    PortVisibility,
    PortConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
    id: "node1",
    width: 100,
    height: 100,
    offsetX: 200,
    offsetY: 200,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    ports:[{
        offset: {
            x: 0.5,

```

```

        y: 0
      },
      visibility: PortVisibility.Visible,
      //Set the style for the port
      style: {
        fill: '#FFFFFF',
        strokeWidth: 1,
        strokeColor: 'black'
      },
      //Defines mouse over tooltip for a node
      tooltip: {
        //Sets the content of the Tooltip
        content: 'Port Tooltip',
        //Sets the tooltip position relative to the node
        relativeMode: 'Object'
      },
      constraints: PortConstraints.Default |
PortConstraints.Tooltip
    ]}
  });
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'650px'}
      height={'350px'}
      //Defines nodes
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Tooltip template content

Any text or image can be added to the tooltip, by default. To customize the tooltip layout or to create your own visualized element on the tooltip, template can be used.

The following code example illustrates how to add formatted HTML content to the tooltip.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {

```

```

        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      },
    ]],
    offsetX: 200,
    offsetY: 200,
    style: {
      strokeColor: '#6BA5D7',
      fill: '#6BA5D7'
    },
    constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
      //Sets the content of the Tooltip
      content: getContent(),
      //Sets the position of the Tooltip
      position: 'TopLeft',
      //Sets the tooltip position relative to the node
      relativeMode: 'Object'
    }
  }
  });
function App() {
  return (<DiagramComponent id="container" width={'650px'}
    height={'350px'} constraints={DiagramConstraints.Default |
    DiagramConstraints.Tooltip}
    //Defines nodes
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
function getContent() {
  let tooltipContent = document.createElement('div');
  tooltipContent.innerHTML = '<div style="background-color: #f4f4f4;
color: black; border-width:1px;border-style: solid;border-color: #d3d3d3;
border-radius: 8px;white-space: nowrap;"> <span style="margin: 10px;">
Tooltip !!! </span> </div>';
  return tooltipContent;
}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  DiagramConstraints,
  NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
  NodeAnimationSettings
} from "@syncfusion/ej2-navigations";

```

```

// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    },
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: NodeConstraints.Default |
NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    //Sets the content of the Tooltip
    content: getContent(),
    //Sets the position of the Tooltip
    position: 'TopLeft',
    //Sets the tooltip position relative to the node
    relativeMode: 'Object'
  }
}];

function App() {
  return (
    <DiagramComponent
      id="container"
      width={'650px'}
      height={'350px'}
      constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
      //Defines nodes
      nodes={node}
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

function getContent(): HTMLElement {
  let tooltipContent: HTMLElement = document.createElement('div');
  tooltipContent.innerHTML = '<div style="background-color: #f4f4f4;
color: black; border-width:1px;border-style: solid;border-color: #d3d3d3;
border-radius: 8px;white-space: nowrap;"> <span style="margin: 10px;">
Tooltip !!! </span> </div>';
  return tooltipContent;
}

```

Tooltip alignments

Tooltip relative to object

The diagram provides support to show tooltip around the node/connector that is hovered by the mouse. The tooltip can be aligned by using the [position](#) property of the tooltip. The [relativeMode](#) property of the tooltip defines whether the tooltip has to be displayed around the object or at the mouse position.

The following code example illustrates how to position the tooltip around object.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
// A node is created and stored in nodes array.
let node = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    content: 'Node1',
    //Sets the alignment properties
    position: 'BottomRight',
    //Sets to show tooltip around the element
    relativeMode: 'Object',
  },
}],
};
//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="container" width={'650px'}
height={'350px'} constraints={DiagramConstraints.Default |
DiagramConstraints.Tooltip}
//Defines nodes
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
```

```
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  DiagramComponent,
  DiagramConstraints,
  NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
  NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
// A node is created and stored in nodes array.
let node: NodeModel[] = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: NodeConstraints.Default |
NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    content: 'Node1',
    //Sets the alignment properties
    position: 'BottomRight',
    //Sets to show tooltip around the element
    relativeMode: 'Object',
  },
}];
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'650px'}
      height={'350px'}
    />
```



```

        constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
        //Defines nodes
        nodes={node}
    />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Tooltip relative to mouse position

To display the tooltip at mouse position, need to set **mouse** option to the [relativeMode](#) property of the tooltip.

The following code example illustrates how to show tooltip at mouse position.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
let node = [{
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
    //Defines mouse over tooltip for a node
    tooltip: {
        content: 'Node1',
        //Sets to show tooltip at mouse position
        relativeMode: 'Mouse',
    },
}],
//Initializes the Diagram component
function App() {
    return (<DiagramComponent id="container" width={'650px'}
height={'350px'} constraints={DiagramConstraints.Default |
DiagramConstraints.Tooltip}
        //Defines nodes

```

```

        nodes={node}/>);
    }
    const root = ReactDOM.createRoot(document.getElementById('diagram'));
    root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    DiagramComponent,
    DiagramConstraints,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
    NodeAnimationSettings
} from "@syncfusion/ej2-navigations";
let node: NodeModel[] = [{
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }],
    offsetX: 200,
    offsetY: 200,
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7'
    },
    constraints: NodeConstraints.Default |
NodeConstraints.Tooltip,
//Defines mouse over tooltip for a node
    tooltip: {
        content: 'Node1',
        //Sets to show tooltip at mouse position
        relativeMode: 'Mouse',
    },
}];
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'650px'}
            height={'350px'}

```

```

        constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
        //Defines nodes
        nodes={node}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Tooltip animation

To animate the tooltip, a set of specific animation effects are available, and it can be controlled by using the [animation](#) property. The animation property also allows you to set delay, duration, and various other effects of your choice.

INDEX.JSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
let node = [{
  id: "node1",
  width: 100,
  height: 100,
  annotations: [{
    id: 'label',
    content: 'Rectangle',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    },
  }],
  offsetX: 200,
  offsetY: 200,
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7'
  },
  constraints: NodeConstraints.Default | NodeConstraints.Tooltip,
  //Defines mouse over tooltip for a node
  tooltip: {
    content: 'Node1',
    position: 'BottomCenter',
    relativeMode: 'Object',
    animation: {
      //Animation settings to be applied on the Tooltip, while it
      //is being shown over the target.
      open: {
        //Animation effect on the Tooltip is applied during open
        //and close actions.
        effect: 'ZoomIn',
        //Duration of the animation that is completed per
        //animation cycle.

```

```

        duration: 1000,
        //Indicating the waiting time before animation begins.
        delay: 0
    },
    //Animation settings to be applied on the Tooltip, when it
    is closed.
    close: {
        effect: 'ZoomOut',
        duration: 500,
        delay: 0
    },
    },
    },
    });
//Initializes the Diagram component
function App() {
    return (<DiagramComponent id="container" width={'650px'}
    height={'350px'} constraints={DiagramConstraints.Default |
    DiagramConstraints.Tooltip}
    //Defines nodes
    nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    NodeModel,
    DiagramConstraints,
    NodeConstraints
} from "@syncfusion/ej2-react-diagrams";
import {
    NodeAnimationSettings
} from '@syncfusion/ej2-navigations';
let node: NodeModel[] = [{
    id: "node1",
    width: 100,
    height: 100,
    annotations: [{
        id: 'label',
        content: 'Rectangle',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    },
    ],
    offsetX: 200,
    offsetY: 200,
}

```

```

        style: {
            strokeColor: '#6BA5D7',
            fill: '#6BA5D7'
        },
        constraints: NodeConstraints.Default |
NodeConstraints.Tooltip,
        //Defines mouse over tooltip for a node
        tooltip: {
            content: 'Node1',
            position: 'BottomCenter',
            relativeMode: 'Object',
            animation: {
                //Animation settings to be applied on the
                Tooltip, while it is being shown over the target.
                open: {
                    //Animation effect on the Tooltip is applied
                    during open and close actions.
                    effect: 'ZoomIn',
                    //Duration of the animation that is
                    completed per animation cycle.
                    duration: 1000,
                    //Indicating the waiting time before
                    animation begins.
                    delay: 0
                },
                //Animation settings to be applied on the
                Tooltip, when it is closed.
                close: {
                    effect: 'ZoomOut',
                    duration: 500,
                    delay: 0
                }
            },
        },
    },
    });
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'650px'}
            height={'350px'}
            constraints={DiagramConstraints.Default | DiagramConstraints.Tooltip}
            //Defines nodes
            nodes={node}
        />
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

User handle in React Diagram component

- User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the [userHandles](#) collection of the [selectedItems](#) property.

- The name property of user handle is used to define the name of the user handle and its further used to find the user handle at runtime and do any customization.

Alignment

User handles can be aligned relative to the node boundaries. It has [margin](#), [offset](#), [side](#), [horizontalAlignment](#), and [verticalAlignment](#) settings. It is quite tricky when all four alignments are used together but gives more control over alignment.

Offset for user handle

The [offset](#) property of [userHandles](#) is used to align the user handle based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

Side

The [side](#) property of [userHandles](#) is used to align the user handle by using the [Top](#), [Bottom](#), [Left](#), and [Right](#) options.

Horizontal and vertical alignments

The [horizontalAlignment](#) property of [userHandles](#) is used to set how the user handle is horizontally aligned at the position based on the [offset](#). The [verticalAlignment](#) property is used to set how user handle is vertically aligned at the position.

Margin for the user handle

Margin is an absolute value used to add some blank space in any one of its four sides. The [userHandles](#) can be displaced with the [margin](#) property.

Appearance

The appearance of the user handle can be customized by using the [size](#), [borderColor](#), [backgroundColor](#), [visible](#), [pathData](#), and [pathColor](#) properties of the [userHandles](#).

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, MoveTool, SelectorConstraints, randomId,
cloneObject } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let shape = {
  type: 'Basic',
  shape: 'Rectangle'
};
let node1 = [{
  id: 'node',
  offsetX: 100,
  offsetY: 100,
  shape: shape
}];
let handles = [{
  name: 'clone',
  pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z
M68.5,28.9h-30c-3, 0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-
2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-
30V34.4h30V72.5z',
  visible: true,
  offset: 0,
```

```

        side: 'Bottom',
        pathColor: "white",
        margin: {
            top: 0,
            bottom: 0,
            left: 0,
            right: 0
        }
    }
    });
ReactDOM.render(<DiagramComponent id="container" ref={diagram =>
(diagramInstance = diagram)} width={"100%"} height={"600px"} nodes={node1}
selectedItems={{
    constraints: SelectorConstraints.UserHandle,
    userHandles: handles
}}
//set Node default value
getNodeDefaults=(node) => {
    node.height = 100;
    node.width = 100;
    node.style.fill = '#6BA5D7';
    node.style.strokeColor = '#6BA5D7';
    return node;
}
//set CustomTool
getCustomTool={getTool}/>, document.getElementById("diagram"));
function getTool(action) {
    let tool;
    if (action === 'clone') {
        tool = new CloneTool(diagramInstance.commandHandler);
    }
    return tool;
}
//Defines the clone tool used to copy Node/Connector
class CloneTool extends MoveTool {
    mouseDown(args) {
        let newObject;
        if (diagramInstance.selectedItems.nodes.length > 0) {
            newObject = cloneObject(diagramInstance.selectedItems.nodes[0]);
        }
        else {
            newObject =
cloneObject(diagramInstance.selectedItems.connectors[0]);
        }
        newObject.id += randomId();
        diagramInstance.paste([newObject]);
        args.source = diagramInstance.nodes[diagramInstance.nodes.length -
1];
        args.sourceWrapper = args.source.wrapper;
        super.mouseDown(args);
        this.inAction = true;
    }
}
{% enddraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  DiagramComponent,
  NodeModel,
  ConnectorModel,
  Diagram,
  NodeModel,
  BasicShapeModel,
  MoveTool,
  MouseEventArgs,
  IElement,
  UserHandleModel,
  ToolBase,
  SelectorConstraints,
  Actions,
  randomId,
  cloneObject,
  Node,
  Side,
  SnapConstraints
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let shape: BasicShapeModel = {
  type: 'Basic',
  shape: 'Rectangle'
};
let nodes: NodeModel[] = [{
  id: 'node',
  offsetX: 100,
  offsetY: 100,
  shape: shape
}];
let handles: UserHandleModel[] = [{
  name: 'clone',
  pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z
M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-
2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-
30V34.4h30V72.5z',
  visible: true,
  offset: 0,
  side: 'Bottom',
  pathColor: "white",
  margin: {
    top: 0,
    bottom: 0,
    left: 0,
    right: 0
  }
}];
ReactDOM.render(
  <DiagramComponent id="container" ref={diagram => (diagramInstance =
diagram)}
  width = {
    "100%"
  }
  height = {

```



```

        "600px"
    }
    nodes = {
        node1
    }
    selectedItems = {
        {
            constraints: SelectorConstraints.UserHandle,
            userHandles: handles
        }
    }
    //set Node default value
    getNodeDefaults = {
        (node: NodeModel): NodeModel => {
            node.height = 100;
            node.width = 100;
            node.style.fill = '#6BA5D7';
            node.style.strokeColor = '#6BA5D7';
            return node;
        }
    }
    //set CustomTool
    getCustomTool = {
        getTool
    }
/> , document.getElementById("diagram")
);
function getTool(action: string): ToolBase {
    let tool: ToolBase;
    if (action === 'clone') {
        tool = new CloneTool(diagramInstance.commandHandler);
    }
    return tool;
}
//Defines the clone tool used to copy Node/Connector
class CloneTool extends MoveTool {
    public mouseDown(args: MouseEventArgs): void {
        let newObject: any;
        if (diagramInstance.selectedItems.nodes.length > 0) {
            newObject = cloneObject(diagramInstance.selectedItems.nodes[0])
as NodeModel;
        } else {
            newObject =
cloneObject(diagramInstance.selectedItems.connectors[0]) as ConnectorModel;
        }
        newObject.id += randomId();
        diagramInstance.paste([newObject]);
        args.source = diagramInstance.nodes[diagramInstance.nodes.length -
1] as IElement;
        args.sourceWrapper = args.source.wrapper;
        super.mouseDown(args);
        this.inAction = true;
    }
}
}

```

Fixed user handles

The fixed user handles are used to add some frequently used commands around the node and connector even without selecting it.

Initialization an fixed user handles

To create the fixed user handles, define and add them to the collection of nodes and connectors property. The following code example used to create an fixed user handles for the nodes and connectors.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let node = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Node.
  fixedUserHandles: [{ margin: { right: 20 }, pathData:
'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-
3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let node: NodeModel[] = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
},
```

```
// A fixed user handle is created and stored in fixed user handle
collection of Node.
fixedUserHandles: [{ margin: { right: 20 }, pathData: 'M60.3,18H27.5c-
3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={ '100%' }
      height={ '600px' }
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Customization

- The id property of fixed user handle is used to define the unique identification of the fixed user handle and it is further used to add custom events to the fixed user handle.
- The fixed user handle can be positioned relative to the node and connector boundaries. It has offset, padding and cornerRadius settings. It is used to position and customize the fixed user handle.
- The **Padding** is used to leave the space that is inside the fixed user handle between the icon and border.
- The corner radius allows to create a fixed user handles with rounded corners. The radius of the rounded corner is set with the **cornerRadius** property.

Note: The PathData needs to be provided to render fixed user handle.

Size

Diagram allows you set size for the fixed user handles by using the **width** and **height** property. The default value of the width and height property is 10.

Style

- You can change the style of the fixed user handles with the specific properties of **borderColor**, **borderWidth**, and background color using the **handleStrokeColor**, **handleStrokeWidth**, and **fill** properties, and the icon **borderColor**, and **borderWidth** using the **iconStrokeColor** and **iconStrokeWidth**.
- The fixed user handle's **iconStrokeColor** and **iconStrokeWidth** property used to change the stroke color and stroke width of the given **pathData**.
- The fixed user handle **handleStrokeColor** and **fill** properties are used to define the background color and border color of the userhandle and the **handleStrokeWidth** property is used to define the border width of the fixed user handle.

- The `visible` property of the fixed user handle enables or disables the visibility of fixed user handle.

The following code explains how to customize the appearance of the fixed user handles.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let diagramInstance;
let node = [{
  offsetX: 150,
  offsetY: 150,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Node.
  fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
padding: { left: 3, right: 3, top: 3, bottom: 3 }, iconStrokeColor: 'white',
fill: 'black', height: 20, width: 20, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
}];
let connector = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
  targetPoint: {
    x: 400,
    y: 200
  },
  type: 'Orthogonal',
  style: {
    strokeColor: '#6BA5D7'
  },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Connector.
  fixedUserHandles: [{ offset: 0.5, width: 20, alignment: 'Before',
padding: { left: 3, right: 3, top: 3, bottom: 3 }, iconStrokeColor: 'white',
fill: 'black', height: 20, id: 'usercon1', displacement: { x: 10, y: 10 },
pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z
M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-
2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-
30V34.4h30V72.5z' }]
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node} connectors={connector}/>);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

INDEX.TSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
let node: NodeModel[] = [{
  offsetX: 150,
  offsetY: 150,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
},
// A fixed user handle is created and stored in fixed user handle collection of Node.
  {
    fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20
},padding:{left:3,right:3,top:3,bottom:3},iconStrokeColor:'white',fill:'black', height: 20, width: 20, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
}],
let connector: ConnectorModel[] = [{
  sourcePoint: {
    x: 300,
    y: 100
  },
},
  {
    targetPoint: {
      x: 400,
      y: 200
    },
  },
  {
    type: 'Orthogonal',
    style: {
      strokeColor: '#6BA5D7'
    },
  },
},
// A fixed user handle is created and stored in fixed user handle collection of Connector.
  {
    fixedUserHandles: [{ offset: 0.5, width: 20, alignment: 'Before',padding:{left:3,right:3,top:3,bottom:3},iconStrokeColor:'white',fill:'black', height: 20, id: 'usercon1', displacement:{x:10,y:10}, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
}],
function App() {
  return (
    <DiagramComponent
```

```

        id="container"
        width={ '100%' }
        height={ '600px' }
        nodes={node}
        connectors={connector}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);

```

Note: The fixed user handle id need to be unique.

Customizing the node fixed user handle

- The node fixed user handle can be aligned relative to the node boundaries. It has **margin** and **offset** settings. It is quite useful to position the node fixed userhandle and used together and gives you more control over the node fixed user handle positioning.

Margin for the node fixed user handle

Margin is an absolute value used to add some blank space in any one of its four sides. The fixed user handle can be displaced with the **margin** property.

Offset for the node fixed user handle

The **offset** property of fixed user handle is used to align the user handle based on the **x** and **y** points. (0,0) represents the top or left corner and (1,1) represents the bottom or right corner.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Margin	Output
-----	-----	-----

| (0,0) | Right = 20 |



|



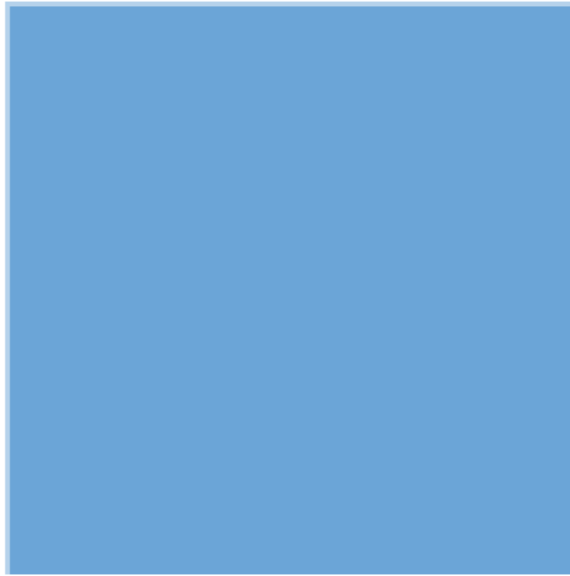
| (0.5,0) | Bottom = 20 |

|



| (1,0) | Left = 20 |

|



| (0,0.5) | Right = 20 |

|



| (0,1) | Left = 20 |

|

| (0,1) | Right = 20 |





| (0.5,1) | Top = 20 |

|



| (1,1) | Left = 20 |

|

The following code explains how to customize the node fixed user handle.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let node = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Node.
  fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
width: 20, height: 20, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
  }];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node}/>);
```

```

}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let node: NodeModel[] = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  // A fixed user handle is created and stored in fixed user handle
  // collection of Node.
  fixedUserHandles: [{ offset: { x: 0, y: 0 }, margin: { right: 20 },
width: 20, height: 20, pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3,0-5.5,2.4-
5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' } ]
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

Customizing the connector fixed user handle

- The connector fixed user handle can be aligned relative to the connector boundaries. It has alignment, displacement and offset settings. It is useful to position the connector fixed userhandle and used together and gives you more control over the connector fixed user handle positioning.
- The `offset` and `alignment` properties of fixed user handle allows you to align the connector fixed user handles to the segments.

Offset for the connector fixed user handle

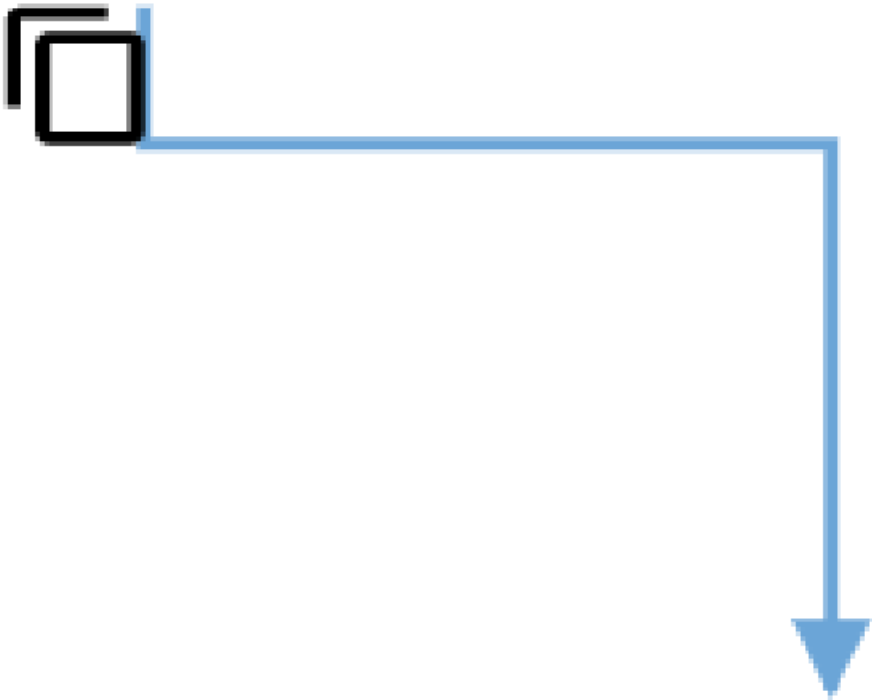
The `offset` property of connector fixed user handle is used to align the user handle based on fractions. 0 represents the connector source point, 1 represents the connector target point, and 0.5 represents the center point of the connector segment.

Alignment

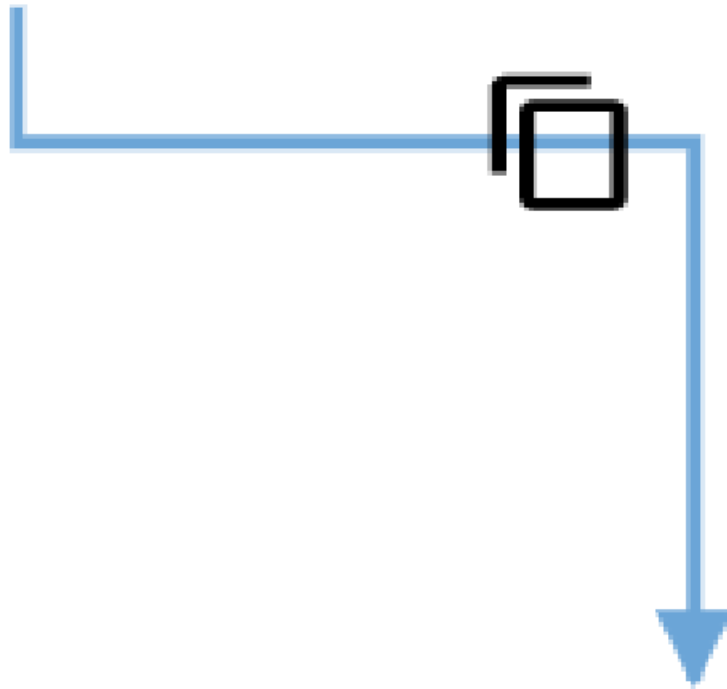
The connector's fixed user handle can be aligned over its segment path using the `alignment` property of fixed user handle.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Offset	Alignment	Output
-----	-----	-----



| 0 | Before |





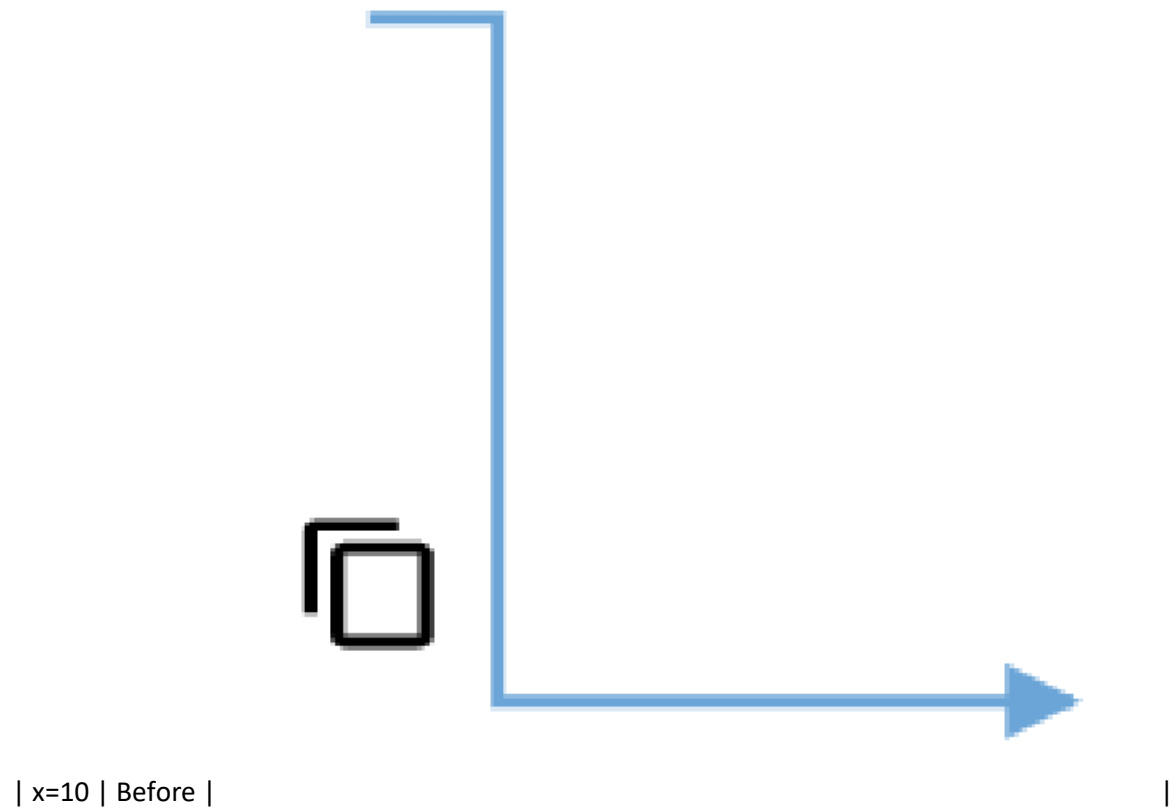
| 1 | After | |

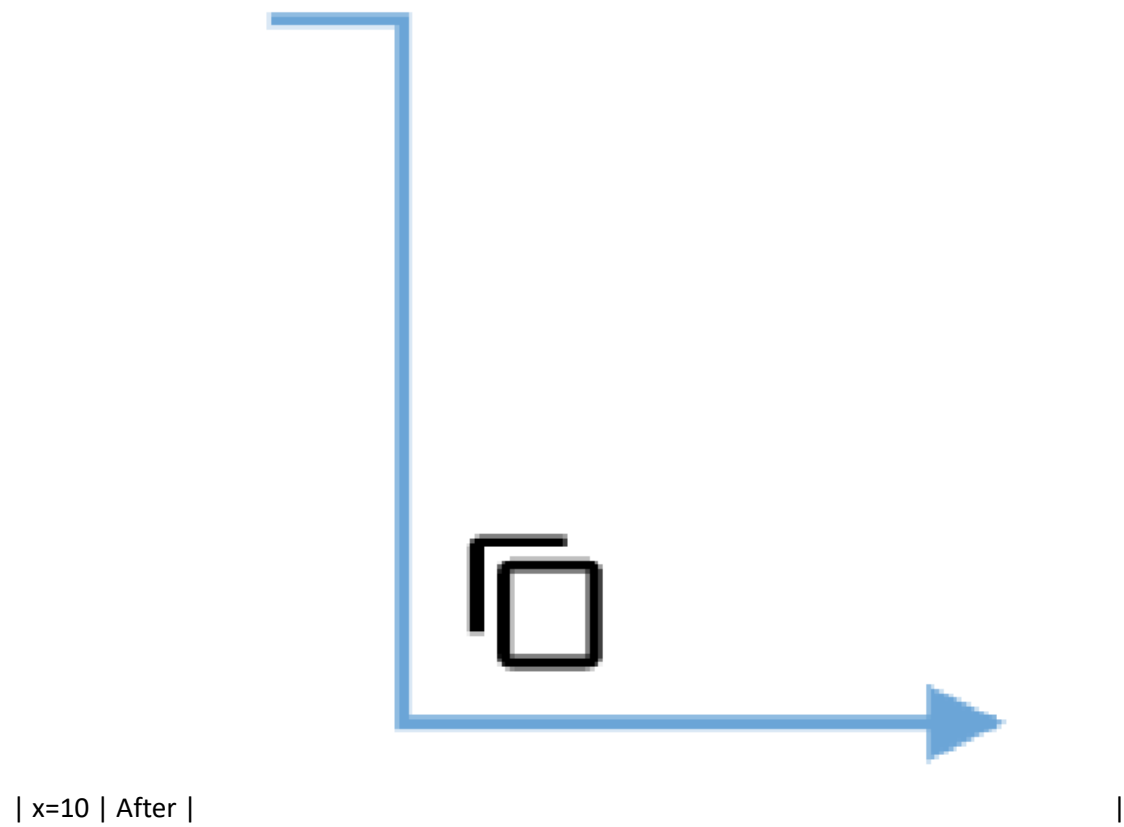
Displacement

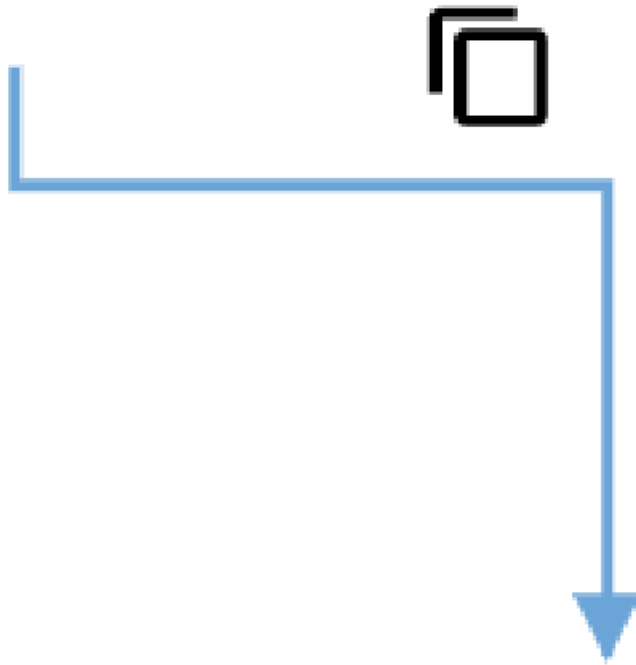
- The `displacement` property allows you to specify the space to be left from the connector segment based on the x and y value provided.

The following table shows all the possible alignments visually shows the fixed user handle positions.

Displacement	Alignment	Output
-----	-----	-----

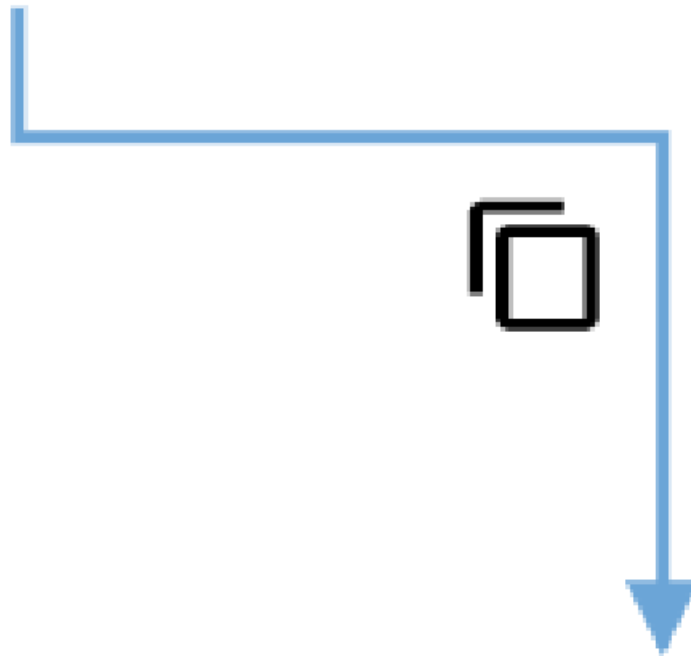






| y=10 | Before |

|



| y=10 | After |

Note: Displacement will not be done if the alignment is set to be center.

The following code explains how to customize the connector fixed user handle.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
let connectors = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
```

```

        x: 200,
        y: 200
      }, type: 'Orthogonal',
      // A fixed user handle is created and stored in fixed user handle
      collection of Connector.
      fixedUserHandles: [{ offset: 0.5, width: 20, alignment: 'Before',
height: 20, id: 'usercon1', displacement: { x: 10, y: 10 }, pathData:
'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-
3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
    ]];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
connectors={connectors}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel
} from "@syncfusion/ej2-react-diagrams";
let connectors: ConnectorModel = [{
  id: "connector1",
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  },
  sourcePoint: {
    x: 100,
    y: 100
  },
  targetPoint: {
    x: 200,
    y: 200
  },
  type: 'Orthogonal',
  // A fixed user handle is created and stored in fixed user handle
  collection of Connector.
  fixedUserHandles: [{ offset: 0.5, width: 20, alignment: 'Before',
height: 20, id: 'usercon1', displacement:{x:10,y:10}, pathData:
'M60.3,18H27.5c-3,0-5.5,2.4-5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-
3,0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z M68.5,72.5h-30V34.4h30V72.5z' }]
    ]];

```

```
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      connectors={connectors}
    />
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

Tooltip support for User Handle

The diagram provides support to show tooltip when the mouse hovers over any user handle.

To show tooltip on mouse over, the [tooltip](#) property of diagram model needs to be set with the tooltip [content](#) and [position](#) as shown in the following example.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, SelectorConstraints, NodeConstraints } from
"@syncfusion/ej2-react-diagrams";
let node = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  tooltip: { content: 'node1', position: 'BottomRight', relativeMode:
'Object' },
  constraints: NodeConstraints.Default | NodeConstraints.Tooltip
}];
let handles = [{
  name: 'clone', pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3, ' +
'0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z ' +
'M68.5,72.5h-30V34.4h30V72.5z',
  visible: true, offset: 0, side: 'Bottom', margin: { top: 0, bottom:
0, left: 0, right: 0 },
  tooltip: { content: 'handle1', position: 'BottomRight',
relativeMode: 'Object' }
}];
function App() {
  return (<DiagramComponent id="container" width={'100%'} height={'600px'}
nodes={node} selectedItems={{
  constraints: SelectorConstraints.All,
  userHandles: handles,
}}/>);
```



```

}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  NodeModel,
  SelectorConstraints,
  NodeConstraints,
  UserHandleModel
} from "@syncfusion/ej2-react-diagrams";
let node: NodeModel[] = [{
  offsetX: 250,
  offsetY: 250,
  width: 100,
  height: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white'
  },
  tooltip: { content: 'node1', position: 'BottomRight', relativeMode:
'Object' },
  constraints: NodeConstraints.Default | NodeConstraints.Tooltip
}];
let handles: UserHandleModel[] = [{
  name: 'clone', pathData: 'M60.3,18H27.5c-3,0-5.5,2.4-
5.5,5.5v38.2h5.5V23.5h32.7V18z M68.5,28.9h-30c-3, ' +
'0-5.5,2.4-5.5,5.5v38.2c0,3,2.4,5.5,5.5,5.5h30c3,0,5.5-2.4,5.5-
5.5V34.4C73.9,31.4,71.5,28.9,68.5,28.9z ' +
'M68.5,72.5h-30V34.4h30V72.5z',
  visible: true, offset: 0, side: 'Bottom', margin: { top: 0, bottom:
0, left: 0, right: 0 },
  tooltip: { content: 'handle1', position: 'BottomRight',
relativeMode: 'Object' }
}];
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={node}
      selectedItems={{
        constraints: SelectorConstraints.All,
        userHandles: handles,
      }}
    />
  );
}

```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Style in React Diagram component

Customizing the connector end point handle

Use the following CSS to customize the connector end point handle.

```
`scss
```

```
.e-diagram-endpoint-handle {
```

```
fill: red;
```

```
stroke: green;
```

```
}
```

```
`
```

Customizing the connector end point handle when connected

Use the following CSS to customize the connector end point handle when connected.

```
`scss
```

```
.e-diagram-endpoint-handle.e-connected {
```

```
fill: red;
```

```
stroke: green;
```

```
}
```

```
`
```

Customizing the connector end point handle when disabled

Use the following CSS to customize the connector end point handle when disabled.

```
`scss
```

```
.e-diagram-endpoint-handle.e-disabled {
```

```
fill: red;
```

```
opacity: 1;
```

```
stroke: green;
```

```
}
```

```
`
```

Customizing the bezier connector handle

Use the following CSS to customize the bezier handle properties.

```
`scss
```

```
.e-diagram-bezier-handle {
```

```
fill: red;
```

```
stroke: green;  
}
```

```
,
```

Customizing the bezier connector line

Use the following CSS to customize the bezier line properties.

```
`scss  
.e-diagram-bezier-line {  
stroke: black;  
}
```

```
,
```

Customizing the resize handle

Use the following CSS to customize the resize handle.

```
`scss  
.e-diagram-resize-handle {  
fill: white;  
opacity: 1;  
stroke: white;  
}
```

```
,
```

Customizing the selector pivot line

Use the following CSS to customize the line between the selector and rotate handle.

```
`scss  
.e-diagram-pivot-line {  
stroke: red;  
}
```

```
,
```

Customizing the selector border

Use the following CSS to customize the selector border.

```
`scss  
.e-diagram-border {  
stroke: red;  
}
```

```
,
```

Customizing the rotate handle

Use the following CSS to customize the rotate handle properties.

```
`scss
.e-diagram-rotate-handle {
  fill: red;
  stroke: green;
}
`
```

Customizing the symbolpalette while hovering

Use the following CSS to customize the symbolpalette while hovering.

```
`scss
.e-symbolpalette .e-symbol-hover:hover {
  background: red;
}
`
```

Customizing the symbolpalette when selected

Use the following CSS to customize the symbolpalette when selected.

```
`scss
.e-symbolpalette .e-symbol-selected {
  background: white;
}
`
```

Customizing the ruler

Use the following CSS to customize the ruler properties.

```
`scss
.e-diagram .e-ruler {
  background-color: red;
  font-size: 13px;
}
`
```

Customizing the ruler overlap

Use the following CSS to ruler overlap properties.

```
`scss
.e-diagram .e-ruler-overlap {
```

```
background-color: red;
}
```

Customizing the text edit

Use the following CSS to customize the text edit properties.

```
`scss
.e-diagram .e-diagram-text-edit {
background: white;
border-color: red;
border-style: dashed;
border-width: 1px;
box-sizing: content-box;
color: black;
min-width: 50px;
}
```

Customizing the text edit on selection

Use the following CSS to customize the text edit on selection properties.

```
`scss
.e-diagram-text-edit::selection {
background: red;
color: green;
}
```

Ruler in React Diagram component

The Ruler provides a horizontal and vertical guide for measuring in the Diagram control. The Ruler can be used to measure the diagram objects, indicate positions, and align diagram elements. This is especially useful in creating scale models.

Adding Rulers to the Diagram

- The [rulerSettings](#) property is used to control the visibility and appearance of the ruler in the diagram.
- The RulerSettings [showRulers](#) property is used to show or hide the rulers in the diagram.
- The RulerSettings [horizontalRuler](#) and [verticalRuler](#) properties are used to customize the rulers appearance in the diagram.

The following code shows how to add a ruler to the diagram.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    rulerSettings={{ showRulers: true }}></DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            rulerSettings={{ showRulers: true }}
        ></DiagramComponent>
    );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Customizing the Ruler

By default, the ruler segments are arranged based on pixel values.

- The HorizontalRuler's [interval](#) property allows you to define the interval between ruler segments and the [segmentWidth](#) property allows you to define the segment width of the ruler. Similarly, you can use the VerticalRuler's [interval](#) and [segmentWidth](#) properties are used to define the interval and segment width of the vertical ruler
- The HorizontalRuler's [tickAlignment](#) property is used to align the ruler tick either left or right side of the ruler. The VerticalRuler's [tickAlignment](#) property is used to align the ruler tick either top or bottom side of the ruler.
- The HorizontalRuler's [arrangeTick](#) and VerticalRuler's [arrangeTick](#) function is provided for the purpose of customizing the appearance of ruler ticks. It will be called for each tick rendering.
- The HorizontalRuler's [markerColor](#) and VerticalRuler's [markerColor](#) properties are used to define the ruler marker color and marker will be shown when performing the interaction in the diagram.

The following code shows how the diagram ruler can be customized.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent } from "@syncfusion/ej2-react-diagrams";
function App() {
    return (<DiagramComponent id="container" width={'100%'} height={'600px'}
    rulerSettings={{
        showRulers: true,
        horizontalRuler: {
            interval: 8,
            segmentWidth: 100,
            thickness: 25,
            tickAlignment: 'LeftOrTop',
        },
        verticalRuler: {
            interval: 10,
            segmentWidth: 150,
            thickness: 35,
            tickAlignment: 'RightOrBottom',
        },
    }}></DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent
} from "@syncfusion/ej2-react-diagrams";
function App() {
    return (
        <DiagramComponent
            id="container"
            width={'100%'}
            height={'600px'}
            rulerSettings={{
                showRulers: true,
                horizontalRuler: {
                    interval: 8,
                    segmentWidth: 100,
                    thickness: 25,
                    tickAlignment: 'LeftOrTop',
                },
                verticalRuler: {
                    interval: 10,
                    segmentWidth: 150,
                    thickness: 35,

```

```

        tickAlignment: 'RightOrBottom',
      },
    ]},
  </DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Note : The MarkerColor property can be customized using the [marker](#) CSS style.

Layers in React Diagram component

Layer is used to organize related shapes on a diagram control. A layer is a named category of shapes. By assigning shapes to different layers, you can selectively view, remove, and lock different categories of shapes.

In diagram, [Layers](#) provide a way to change the properties of all shapes that have been assigned to that layer. The following properties can be set.

- Visible
- Lock
- Objects
- AddInfo

Visible

The layer's [visible](#) property is used to control the visibility of the elements assigned to the layer.

`ts

// A node is created and stored in nodes array.

```

let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Default Shape'
  }]
},
{
  id: 'node2',
  width: 100,

```



```
height: 100,
offsetX: 300,
offsetY: 100,
shape: {
  type: 'Path',
  data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
  content: 'Path Element'
}]
}
];

let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];

// initialize diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
```

```

height={'600px'}
nodes={nodes}
connectors={connectors}
// Add layer
layers={[
  {
    id: 'layer1',
    visible: true,
    objects: ['node1'],
  },
]}
// render initialized Diagram
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Lock

The layer's [lock](#) property is used to prevent or allow changes to the elements dimension and position.

```
`ts
```

```
// A node is created and stored in nodes array.
```

```

let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Default Shape'
  }]
},
{

```

```

id: 'node2',
width: 100,
height: 100,
offsetX: 300,
offsetY: 100,
shape: {
  type: 'Path',
  data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
},
annotations: [{
  content: 'Path Element'
}]
}
];
let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];
// initialize diagram component
function App() {
  return (
<DiagramComponent

```

```
id="container"
width={'100%'}
height={'600px'}
nodes={nodes}
connectors={connectors}
// Add layer
// Add layer
layers = {
[
{
id: 'layer1',
visible: true,
objects: ['node1'],
lock: true
},
{
id: 'layer2',
visible: true,
objects: ['node2'],
lock: false
}
]
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Objects

The layer's [objects](#) property defines the diagram elements to the layer.

`ts

// A node is created and stored in nodes array.

```
let nodes: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  annotations: [{
    content: 'Default Shape'
  }]
},
{
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  shape: {
    type: 'Path',
    data:
'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L
540.3643,' +
'179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1
37.9336z'
  },
  annotations: [{
    content: 'Path Element'
  }]
}
];

let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
```

```
y: 300
},
targetPoint: {
x: 200,
y: 400
},
}];
// initialize diagram component
function App() {
return (
<DiagramComponent
id="container"
width={'100%'}
height={'600px'}
nodes={nodes}
connectors={connectors}
// Add layer
layers = {
[
{
id: 'layer1',
visible: true,
objects: ['node1', 'node2']
},
{
id: 'layer2',
visible: true,
objects: ['node2'],
}
]
}
/>
);
```

```
}  
  
const root = ReactDOM.createRoot(document.getElementById('diagram'));  
root.render(<App />);  
,
```

AddInfo

The [addInfo](#) property of layers allow you to maintain additional information to the layers.

The following code illustrates how to add additional information to the layers.

```
`ts  
  
// A node is created and stored in nodes array.  
let nodes: NodeModel[] = [{  
  id: 'node1',  
  width: 100,  
  height: 100,  
  offsetX: 100,  
  offsetY: 100,  
  annotations: [{  
    content: 'Default Shape'  
  }]  
},  
{  
  id: 'node2',  
  width: 100,  
  height: 100,  
  offsetX: 300,  
  offsetY: 100,  
  shape: {  
    type: 'Path',  
    data:  
      'M540.3643,137.9336L546.7973,159.7016L570.3633,159.7296L550.7723,171.9366L558.9053,194.9966L  
540.3643,' +  
      '179.4996L521.8223,194.9966L529.9553,171.9366L510.3633,159.7296L533.9313,159.7016L540.3643,1  
37.9336z'  
  },  
  annotations: [{
```

```
content: 'Path Element'
}}
}
];

let connectors: ConnectorModel[] = [{
  id: 'connector1',
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300
  },
  targetPoint: {
    x: 200,
    y: 400
  },
}];

let addInfo: Object = { Description: 'Layer1' };
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
    // Add layer
    layers = {
      [
        {
          id: 'layer1',
          visible: true,
          objects: ['node1', 'node2'],
```



```

addInfo: addInfo
},
{
  id: 'layer2',
  visible: true,
  objects: ['node2'],
}
]
}
// render initialized Diagram
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Add layer at runtime

Layers can be added at runtime by using the [addLayer](#) public method.

The layer's [ID](#) property defines the ID of the layer, and its further used to find the layer at runtime and do any customization.

The following code illustrates how to add a layer.

```

`ts
let diagramInstance: DiagramComponent;

function App() {
  return (
    <DiagramComponent
      id="container"
      ref={{(diagram) => (diagramInstance = diagram)}}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      // Add layer
      created={() => {

```

```
// add the layers to the existing diagram layer collection
diagramInstance.addLayer(
{
  id: 'newlayer',
  objects: [],
  visible: true,
  lock: false,
  zIndex: -1,
},
[
{
  type: 'Straight',
  sourcePoint: {
    x: 100,
    y: 300,
  },
  targetPoint: {
    x: 200,
    y: 400,
  },
},
]
);
}}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

Remove layer at runtime

Layers can be removed at runtime by using the [removeLayer](#) public method.

The following code illustrates how to remove a layer.

```

`ts
let diagramInstance: DiagramComponent;
// initialize Diagram component
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={{(diagram) => (diagramInstance = diagram)}}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
    // Add layer
    created = {
      () => {
        // remove the diagram layers
        diagram.removeLayer([diagram.model.layers[i]]);
      }
    }
  />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

moveObjects

Objects of the layers can be moved by using the [moveObjects](#) public method.

The following code illustrates how to move objects from one layer to another layer from the diagram.

```

`ts
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent

```

```

id="container"
ref={({diagram}) => (diagramInstance = diagram)}
width={'100%'}
height={'600px'}
nodes={nodes}
connectors={connectors}
// Add layer
created = {
  () => {
    // move the objects of diagram layers
    diagram.moveObjects(['connector1'], 'layer2');
  }
}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

bringLayerForward

Layers can be moved forward at runtime by using the [bringLayerForward](#) public method.

The following code illustrates how to bring forward to layer.

```

`ts
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={({diagram}) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}

```

```
// Add layer
created = {
  () => {
    // move the layer forward
    diagram.bringLayerForward('layer1');
  }
}
/>
);
}
```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`
```

sendLayerBackward

Layers can be moved backward at runtime by using the [sendLayerBackward](#) public method.

The following code illustrates how to send backward to layer.

```
`ts
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      // Add layer
      created = {
        () => {
          // Send the layer backward
          diagram.sendLayerBackward('layer1');
        }
      }
    />
  );
}
```

```

}
/>
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

cloneLayer

Layers can be cloned with its object by using the [cloneLayer](#) public method.

The following code illustrates how to bring forward to layer.

```

`ts
let diagramInstance: DiagramComponent;
function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      // Add layer
      created = {
        () => {
          // clone a layer with its object
          diagram.cloneLayer('layer2');
        }
      }
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);

```

`

[getActiveLayer](#)

To get the active layers back in diagram, use the [getActiveLayer](#) public method.

The following code illustrates how to bring forward to layer.

`ts

```
let diagramInstance: DiagramComponent;

function App() {
  return (
    <DiagramComponent
      id="container"
      ref={(diagram) => (diagramInstance = diagram)}
      width={'100%'}
      height={'600px'}
      nodes={nodes}
      connectors={connectors}
      // Add layer
      created = {
        () => {
          // gets the active layer back
          diagram.getActiveLayer();
        }
      }
    />
  );
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
```

`

[setActiveLayer](#)

Set the active layer by using the [setActiveLayer](#) public method.

The following code illustrates how to bring forward to layer.

`ts

```
let diagramInstance: DiagramComponent;

function App() {
```

```

return (
  <DiagramComponent
    id="container"
    ref={({diagram}) => (diagramInstance = diagram)}
    width={'100%'}
    height={'600px'}
    nodes={nodes}
    connectors={connectors}
    // Add layer
    created = {
      () => {
        // set the active layer
        // @param layerName defines the name of the layer which is to be active layer
        diagram.setActiveLayer('layer2');
      }
    }
  />
);
}

const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
`

```

Context menu in React Diagram component

<!-- markdownlint-disable MD010 -->

In graphical user interface (GUI), a context menu is a type of menu that appears when you perform right-click operation. Nested level of context menu items can be created.

Diagram provides some in-built context menu items and allows to define custom menu items through the [contextMenuSettings](#) property.

Customize context menu

The [show](#) property helps you to enable/disable the context menu. Diagram provides some default context menu items to ease the execution of some frequently used commands.

The following code illustrates how to enable the default context menu items.

INDEX.JSX

```

{% raw %}
import * as React from "react";

```



```

import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramContextMenu, Inject } from
"@syncfusion/ej2-react-diagrams";
//Initializes the connector
let connector = [{
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    }
}];
//Initializes the nodes
let node = [{
    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label1',
        content: 'Rectangle1',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }],
}, {
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label2',

```

```

        content: 'Rectangle2',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }],
    });
//Initializes the Diagram component
ReactDOM.render(<DiagramComponent id="diagram_contextmenu" width={'650px'}
height={'350px'} nodes={node} connectors={connector}
//Enables the context menu
contextMenuSettings={{
    show: true
}}><Inject services={[DiagramContextMenu]}/>
</DiagramComponent>, document.getElementById("diagram"));
{% endraw %}

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    DiagramContextMenu,
    Inject,
    ConnectorModel,
    NodeModel,
    DiagramBeforeMenuOpenEventArgs
} from "@syncfusion/ej2-react-diagrams";
import {
    MenuEventArgs
} from '@syncfusion/ej2-navigations';
//Initializes the connector
let connector: ConnectorModel[] = [{
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor : '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth : 2,
    },
    targetDecorator: {
        style: {
            fill : '#6BA5D7',
            strokeColor : '#6BA5D7'
        }
    }
}
]];
//Initializes the nodes
let node: NodeModel[] = [{

```

```

    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
    annotations: [{
      id: 'label1',
      content: 'Rectangle1',
      offset: {
        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      }
    }],
  }, {
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
    annotations: [{
      id: 'label2',
      content: 'Rectangle2',
      offset: {
        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      }
    }],
  }],
  });
//Initializes the Diagram component
ReactDOM.render( <DiagramComponent id = "diagram_contextmenu"
  width = {
    '650px'
  }
  height = {
    '350px'
  }
  nodes = {
    node
  }
  connectors = {
    connector

```

```

    }
    //Enables the context menu
    contextMenuSettings = {
      {
        show: true
      }
    }
    <Inject services = {[DiagramContextMenu]}/>
  </DiagramComponent>, document.getElementById("diagram")
);

```

Context menu can be defined for individual node with the desired context menu items.

- Apart from the default context menu items, define some additional context menu items. Those additional items have to be defined and added to the [items](#) property of the context menu.
- Set text and ID for context menu item using the context menu [text](#) and [ID](#) properties respectively.
- Set an image for the context menu item using the context menu [url](#) property.
- The [iconCss](#) property defines the class/multiple classes separated by a space for the menu item that is used to include an icon. Menu item can include font icon and sprite image.
- The [target](#) property used to set the target to show the menu item.
- The [separator](#) property defines the horizontal lines that are used to separate the menu items. You cannot select the separators. You can enable separators to group the menu items using the [separator](#) property.

The following code example illustrates how to add custom context menu items.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, DiagramContextMenu, Inject } from
"@syncfusion/ej2-react-diagrams";
//Initializes the connector
let connector = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  type: 'Straight',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2,
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
}];
//Initializes the nodes
let node = [{

```

```

    id: 'node1',
    width: 100,
    height: 100,
    offsetX: 100,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
    annotations: [{
      id: 'label1',
      content: 'Rectangle1',
      offset: {
        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      }
    }]
  }, {
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
      fill: '#6BA5D7',
      strokeColor: 'white',
      strokeWidth: 1
    },
    annotations: [{
      id: 'label2',
      content: 'Rectangle2',
      offset: {
        x: 0.5,
        y: 0.5
      },
      style: {
        color: 'white'
      }
    }]
  }
];

//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="diagram_contextmenu" width={'650px'}
    height={'350px'} nodes={node} connectors={connector} contextMenuSettings={{
      //Enables the context menu
      show: true,
      // Defines the custom context menu items
      items: [
        {
          // Text to be displayed
          text: 'Save',
          //Sets the id for the item
          id: 'save',

```

```

        //ContextMenu can be visible based on the target in
        which you open the ContextMenu.
        target: '.e-elementcontent',
        // Sets the css icons for the item
        iconCss: 'e-save',
    },
    {
        text: 'Load',
        id: 'load',
        target: '.e-elementcontent',
        iconCss: 'e-load',
    },
    {
        text: 'Clear',
        id: 'clear',
        target: '.e-elementcontent',
        iconCss: 'e-clear',
    },
],
// Hides the default context menu items
showCustomMenuOnly: false,
}}>
<Inject services={[DiagramContextMenu]}/>
</DiagramComponent>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    DiagramComponent,
    DiagramContextMenu,
    Inject,
    ConnectorModel,
    NodeModel,
    DiagramBeforeMenuOpenEventArgs
} from "@syncfusion/ej2-react-diagrams";
import {
    MenuEventArgs
} from '@syncfusion/ej2-navigations';
//Initializes the connector
let connector: ConnectorModel[] = [{
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    type: 'Straight',
    style: {
        strokeColor : '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth : 2,
    }
}

```

```

    },
    targetDecorator: {
      style: {
        fill : '#6BA5D7',
        strokeColor : '#6BA5D7'
      }
    }
  }
];
//Initializes the nodes
let node: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label2',
    content: 'Rectangle2',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}
];
//Initializes the Diagram component
function App() {
  return (

```

```

<DiagramComponent
  id="diagram_contextmenu"
  width={'650px'}
  height={'350px'}
  nodes={node}
  connectors={connector}
  contextMenuSettings={{
    //Enables the context menu
    show: true,
    // Defines the custom context menu items
    items: [
      {
        // Text to be displayed
        text: 'Save',
        //Sets the id for the item
        id: 'save',
        //ContextMenu can be visible based on the target in which you
open the ContextMenu.
        target: '.e-elementcontent',
        // Sets the css icons for the item
        iconCss: 'e-save',
      },
      {
        text: 'Load',
        id: 'load',
        target: '.e-elementcontent',
        iconCss: 'e-load',
      },
      {
        text: 'Clear',
        id: 'clear',
        target: '.e-elementcontent',
        iconCss: 'e-clear',
      },
    ],
    // Hides the default context menu items
    showCustomMenuOnly: false,
  }}
>
  <Inject services={[DiagramContextMenu]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

To display the custom context menu items alone, set the [showCustomMenuOnly](#) property to true.

Template Support for Context menu

- Diagram provides template support for context menu. The context menu items can be customized by using the `contextMenuBeforeItemRender` event. The `contextMenuBeforeItemRender` event triggers while rendering each menu item.

- In the following sample, the menu item is rendered with key code for specified action in Context Menu using the template. Here, the key code is specified for the cut and copy at right corner of the menu items by adding a span element in the `contextMenuBeforeItemRender` event.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DiagramContextMenu } from
"@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the connector
let connector = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor: '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth: 2,
  },
  targetDecorator: {
    style: {
      fill: '#6BA5D7',
      strokeColor: '#6BA5D7'
    }
  }
}];
//Initializes the nodes
let node = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
```

```

        offsetX: 300,
        offsetY: 100,
        style: {
            fill: '#6BA5D7',
            strokeColor: 'white',
            strokeWidth: 1
        },
        annotations: [{
            id: 'label2',
            content: 'Rectangle2',
            offset: {
                x: 0.5,
                y: 0.5
            },
            style: {
                color: 'white'
            }
        }]
    }];
    //Initializes the Diagram component
    function App() {
        return (<DiagramComponent id="diagram_contextmenu" ref={(diagram) =>
        (diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
        connectors={connector} contextMenuSettings={{
            //Enables the context menu
            show: true,
            items: [
                {
                    text: 'Cut',
                    id: 'Cut',
                    target: '.e-diagramcontent',
                    iconCss: 'e-Cut',
                },
                {
                    text: 'Copy',
                    id: 'Copy',
                    target: '.e-diagramcontent',
                    iconCss: 'e-Copy',
                },
            ],
            //Shows the custom context menu items
            showCustomMenuOnly: true,
            contextMenuBeforeItemRender: function (args) {
                // To render template in li.
                let shortCutSpan = createElement('span');
                let text = args.item.text;
                let shortCutText = text === 'Cut'
                    ? 'Ctrl + S'
                    : text === 'Copy'
                    ? 'Ctrl + U'
                    : 'Ctrl + Shift + I';
                shortCutSpan.textContent = shortCutText;
                args.element.appendChild(shortCutSpan);
                shortCutSpan.setAttribute('class', 'shortcut');
            },
        }}>
        <Inject services={[DiagramContextMenu]}/>

```

```

    </DiagramComponent>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  DiagramContextMenu,
  MenuEventArgs,
  ConnectorModel,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
let diagramInstance: DiagramComponent;
//Initializes the connector
let connector: ConnectorModel[] = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
  },
  targetDecorator: {
    style: {
      fill : '#6BA5D7',
      strokeColor : '#6BA5D7'
    }
  }
}
]];
//Initializes the nodes
let node: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    }
  }
]
}

```

```

        },
        style: {
            color: 'white'
        }
    }
}
}, {
    id: 'node2',
    width: 100,
    height: 100,
    offsetX: 300,
    offsetY: 100,
    style: {
        fill: '#6BA5D7',
        strokeColor: 'white',
        strokeWidth: 1
    },
    annotations: [{
        id: 'label2',
        content: 'Rectangle2',
        offset: {
            x: 0.5,
            y: 0.5
        },
        style: {
            color: 'white'
        }
    }
    ]
}
}];
//Initializes the Diagram component
function App() {
    return (
        <DiagramComponent
            id="diagram_contextmenu"
            ref={(diagram) => (diagramInstance = diagram)}
            width={'650px'}
            height={'350px'}
            nodes={node}
            connectors={connector}
            contextMenuSettings={{
                //Enables the context menu
                show: true,
                items: [
                    {
                        text: 'Cut',
                        id: 'Cut',
                        target: '.e-diagramcontent',
                        iconCss: 'e-Cut',
                    },
                    {
                        text: 'Copy',
                        id: 'Copy',
                        target: '.e-diagramcontent',
                        iconCss: 'e-Copy',
                    },
                ],
            }},
            //Shows the custom context menu items
            showCustomMenuOnly: true,

```

```

    contextMenuBeforeItemRender: function (args: MenuEventArgs) {
        // To render template in li.
        let shortCutSpan: HTMLElement = createElement('span');
        let text: string = args.item.text;
        let shortCutText: string =
            text === 'Cut'
                ? 'Ctrl + S'
                : text === 'Copy'
                ? 'Ctrl + U'
                : 'Ctrl + Shift + I';
        shortCutSpan.textContent = shortCutText;
        args.element.appendChild(shortCutSpan);
        shortCutSpan.setAttribute('class', 'shortcut');
    },
    }}
>
    <Inject services={[DiagramContextMenu]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Context menu events

You would be notified with events, when you try to open the context menu items [contextMenuOpen](#) and when you click the menu items [contextMenuClick](#).

The following code example illustrates how to define those events.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject, DiagramContextMenu } from
"@syncfusion/ej2-react-diagrams";
let diagramInstance;
//Initializes the connector
let connector = [{
    id: 'connector1',
    sourceID: 'node1',
    targetID: 'node2',
    style: {
        strokeColor: '#6BA5D7',
        fill: '#6BA5D7',
        strokeWidth: 2,
    },
    targetDecorator: {
        style: {
            fill: '#6BA5D7',
            strokeColor: '#6BA5D7'
        }
    }
}
    ]];
//Initializes the nodes

```

```

let node = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label2',
    content: 'Rectangle2',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}];

//Initializes the Diagram component
function App() {
  return (<DiagramComponent id="diagram_contextmenu" ref={(diagram) =>
    (diagramInstance = diagram)} width={'650px'} height={'350px'} nodes={node}
    connectors={connector} contextMenuSettings={{
      //Enables the context menu
      show: true,
      items: [
        {
          text: 'delete',
          id: 'delete',
        }
      ]
    })
  );
}

```

```

    ],
    // Hides the default context menu items
    showContextMenuOnly: false,
  }} contextMenuOpen={ (args) => {
    //do your custom action here.
    for (let item of args.items) {
      if (item.text === 'delete') {
        if (!diagramInstance.selectedItems.nodes.length &&
          !diagramInstance.selectedItems.connectors.length) {
          args.hiddenItems.push(item.id);
        }
      }
    }
  }} contextMenuClick={ (args) => {
    //do your custom action here.
    if (args.item.text === 'delete') {
      if (diagramInstance.selectedItems.nodes.length +
        diagramInstance.selectedItems.connectors.length >
        0) {
        diagramInstance.cut();
      }
    }
  }}>
  <Inject services={[DiagramContextMenu]}/>
</DiagramComponent>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  Inject,
  DiagramContextMenu,
  DiagramBeforeMenuOpenEventArgs,
  ConnectorModel,
  NodeModel
} from "@syncfusion/ej2-react-diagrams";
import { MenuEventArgs } from '@syncfusion/ej2-navigations';
let diagramInstance: DiagramComponent;
//Initializes the connector
let connector: ConnectorModel[] = [{
  id: 'connector1',
  sourceID: 'node1',
  targetID: 'node2',
  style: {
    strokeColor : '#6BA5D7',
    fill: '#6BA5D7',
    strokeWidth : 2,
  },
},

```

```

    targetDecorator: {
      style: {
        fill : '#6BA5D7',
        strokeColor : '#6BA5D7'
      }
    }
  }];
//Initializes the nodes
let node: NodeModel[] = [{
  id: 'node1',
  width: 100,
  height: 100,
  offsetX: 100,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label1',
    content: 'Rectangle1',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}, {
  id: 'node2',
  width: 100,
  height: 100,
  offsetX: 300,
  offsetY: 100,
  style: {
    fill: '#6BA5D7',
    strokeColor: 'white',
    strokeWidth: 1
  },
  annotations: [{
    id: 'label2',
    content: 'Rectangle2',
    offset: {
      x: 0.5,
      y: 0.5
    },
    style: {
      color: 'white'
    }
  }]
}];
//Initializes the Diagram component
function App() {
  return (
    <DiagramComponent

```



```

id="diagram_contextmenu"
ref={ (diagram) => (diagramInstance = diagram)}
width={'650px'}
height={'350px'}
nodes={node}
connectors={connector}
contextMenuSettings={{
  //Enables the context menu
  show: true,
  items: [
    {
      text: 'delete',
      id: 'delete',
    },
  ],
  // Hides the default context menu items
  showCustomMenuOnly: false,
}}
contextMenuOpen={ (args) => {
  //do your custom action here.
  for (let item of args.items) {
    if (item.text === 'delete') {
      if (
        !diagramInstance.selectedItems.nodes.length &&
        !diagramInstance.selectedItems.connectors.length
      ) {
        args.hiddenItems.push(item.id);
      }
    }
  }
}}
contextMenuClick={ (args) => {
  //do your custom action here.
  if (args.item.text === 'delete') {
    if (
      diagramInstance.selectedItems.nodes.length +
      diagramInstance.selectedItems.connectors.length >
      0
    ) {
      diagramInstance.cut();
    }
  }
}}
>
<Inject services={[DiagramContextMenu]} />
</DiagramComponent>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Symbol palette in React Diagram component

The **SymbolPalette** displays a collection of palettes. The palette shows a set of nodes and connectors. It allows to drag and drop the nodes and connectors into the diagram.

Create symbol palette

The [width](#) and [height](#) properties of the symbol palette allows to define the size of the symbol palette.

`ts

```
import {
  SymbolPalette,
  SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent id="container" width={"100%"} height={"700px"} />
  );
}

const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
`
```

<!-- markdownlint-disable MD010 -->

Add palettes to SymbolPalette

A palette allows to display a group of related symbols and it textually annotates the group with its header.

A [Palettes](#) can be added as a collection of symbol groups.

The collection of predefined symbols can be added in palettes using the [symbols](#) property.

To initialize a palette, define a JSON object with the property [ID](#) that is unique ID is set to the palettes.

The following code example illustrates how to define a palette and how its added to symbol palette.

INDEX.JSX

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
```

```

        shape: {
          type: "Basic",
          shape: "Ellipse",
        },
      },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ],
];
return basicShapes;
}
export function getSvgShapes() {
  let basicShapes = [
    {
      id: "node2",
      style: { fill: "none" },
      annotations: [{ content: "Start \n Text Editing" }],
      shape: {
        type: "Native",
        content: '<g xmlns="http://www.w3.org/2000/svg">      <g
transform="translate(1 1)">          <g>          <path
style="fill:#61443C;" d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-
1.109c-2.01,1.33-4.472,1.791-6.827,1.28
C62.726,435.13,62.354,435.072,61.979,435.057z"/>          <path
style="fill:#61443C;" d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304      c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296      c0.377-
47.729-34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z"/>          </g>
      <path style="fill:#8B685A;" d="M476.869,502.471H7.536c-0.191-
32.558,22.574-60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685      c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264      c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296      c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389c456.696,444.61,477.033,471.713,4
76.869,502.471      L476.869,502.471z"/>          <path style="fill:#66993E;"
d="M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-
5.59-59.392-12.032      c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>          <path
style="fill:#9ACA5C;" d="M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6      c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144

```

```

C238.363,206.279,219.931,41.329,476.784,10.183z"/>          <path
style="fill:#66993E;" d="M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-21.505,7.427-
44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z"/>          <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-13.351,4.412-
27.142,7.359-41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/>    </g>    <g>
    <path d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.2981-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671L162.136,426.671z"/>
    <path d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>          <path
d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-
3.335,8.671-0.054,12.01218.533,8.533      c3.341,3.281,8.701,3.256,12.012-
0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z"/>
    <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
    <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-
2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-
3.335-12.012-0.0541-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C198.241,407.524,201.352,409.603,204.803,409.604z"/>          <path
d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C326.241,441.658,329.352,443.737,332.803,443.737z"/>          <path
d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C334.774,364.858,337.885,366.937,341.336,366.937z"/>          <path
d="M164.636,454.7711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
    <path d="M232.903,429.1711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
    <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      c-3.311-3.311-8.671-3.335-
12.012-0.0541-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298
C377.441,407.524,380.552,409.603,384.003,409.604z"/>          <path
d="M70.77,463.3041-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271s3.1,5.28,6.065,6.065      c2.965,0.785,6.122-0.082,8.271-
2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/>          <path
d="M121.97,446.2381-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
    <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-
4.433,0.899-6.033,2.51-8.533,8.533      c-2.178,2.151-3.037,5.304-

```

```
2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-
0.073,8.262-2.25118.533-8.533      c1.601-1.6,2.5-3.77,2.5-
6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z"/>
      <path d="M210.836,463.304c-3.341-3.281-8.701-3.256-
12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-
2.965-0.082-6.122-2.27-8.271L210.836,463.304z"/>      <path
d="M343.836,454.7711-8.533,8.533c-2.188,2.149-3.055,5.307-
2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065      c2.965,0.785,6.122-
0.082,8.271-2.2718.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012
C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
      <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-
8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>      <path
d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-
3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.0541-8.533,8.533c-
2.44,2.441-3.169,6.11-
1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>
      <path d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012      s-8.671-3.335-12.012-0.0541-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>
      <path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796      c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308      C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881      c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196      c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058      c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035      c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045      c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092      c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125      s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874      C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138      c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892      c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
```

```

2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661    c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467    c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z"/>    </g></g>',
    },
  },
  {
    id: "syncfusion",
    style: { fill: "none" },
    shape: {
      type: "Native",
      content: '<g xmlns="http://www.w3.org/2000/svg">' +
        '<rect height="256" width="256" fill="#34353F"/>' +
        '<path id="path1" transform="rotate(0,128,128)'
translate(59,61.2230899333954) scale(4.3125,4.3125)  " fill="#FFFFFF"
d="M18.88501,23.042998L26.804993,23.042998 26.804993,30.969001
18.88501,30.969001z M9.4360352,23.042998L17.358032,23.042998
17.358032,30.969001 9.4360352,30.969001z
M0.014038086,23.042998L7.9360352,23.042998 7.9360352,30.969001
0.014038086,30.969001z M18.871033,13.609001L26.791016,13.609001
26.791016,21.535994 18.871033,21.535994z
M9.4219971,13.609001L17.342041,13.609001 17.342041,21.535994
9.4219971,21.535994z M0,13.609001L7.9219971,13.609001 7.9219971,21.535994
0,21.535994z M9.4219971,4.1859968L17.342041,4.1859968 17.342041,12.113998
9.4219971,12.113998z M0,4.1859968L7.9219971,4.1859968 7.9219971,12.113998
0,12.113998z M25.846008,0L32,5.2310026 26.773987,11.382995
20.619019,6.155998z"/>' +
        "</g>",
    },
  },
  {
    id: "network",
    style: { fill: "none" },
    shape: {
      type: "Native",
      content: '<g xmlns="http://www.w3.org/2000/svg">' +
        '<rect height="256" width="256" fill="#34353F"/>' +
        '<path id="path1" transform="rotate(0,128,128)'
translate(59.1078108549118,59) scale(4.3125,4.3125)  " fill="#FFFFFF"
d="M12.12701,24.294998C12.75201,24.294998 13.258998,24.803009
13.258998,25.428009 13.258998,26.056 12.75201,26.563004 12.12701,26.563004
11.499019,26.563004 10.993007,26.056 10.993007,25.428009 10.993007,24.803009
11.499019,24.294998 12.12701,24.294998z
M7.9750035,24.294998C8.6010101,24.294998 9.1090057,24.803009
9.1090057,25.428009 9.1090057,26.056 8.6010101,26.563004 7.9750035,26.563004
7.3480199,26.563004 6.8399942,26.056 6.8399942,25.428009 6.8399942,24.803009
7.3480199,24.294998 7.9750035,24.294998z
M7.9750035,20.286011C8.6010101,20.286011 9.1090057,20.792999
9.1090057,21.419006 9.1090057,22.044006 8.6010101,22.552002
7.9750035,22.552002 7.3500035,22.552002 6.8420084,22.044006
6.8420084,21.419006 6.8420084,20.792999 7.3500035,20.286011
7.9750035,20.286011z

```

M18.499994,19.317001C18.313013,19.317001,18.156,19.472,18.156,19.656006L18.156,27.01001C18.156,27.195007,18.313013,27.350006,18.499994,27.350006L29.521993,27.350006C29.707998,27.350006,29.865988,27.195007,29.865988,27.01001L29.865988,19.656006C29.865988,19.472,29.707998,19.317001,29.521993,19.317001z

M17.243006,17.443008L30.778003,17.443008C31.425007,17.445007,31.947986,17.962006,31.950001,18.602997L31.950001,28.542007C31.947986,29.182999,31.425007,29.702011,30.778003,29.703003L25.654012,29.703003C25.511007,29.703003,25.399008,29.824997 25.413992,29.964996 25.430013,30.13501 25.452993,30.360001 25.477011,30.559998 25.506002,30.809998 25.727987,30.980011

25.976003,31.033997L27.756002,31.419006C27.907003,31.452011 28.015005,31.58428.015005,31.738007 28.015005,31.883011 27.895986,32 27.74999,32L27.571005,32 20.450004,32 20.318016,32C20.171013,32 20.053001,31.883011 20.053001,31.738007 20.053001,31.585007 20.161003,31.452011

20.312004,31.419998L22.115989,31.033005C22.35601,30.98201 22.572014,30.815002 22.596,30.574005 22.616997,30.363007 22.636009,30.130997 22.648002,29.960007 22.658012,29.819 22.542015,29.70401 22.399986,29.70401L17.243006,29.703003C16.596002,29.702011,16.072992,29.182999,16.071008,28.542007L16.071008,18.602997C16.072992,17.962006,16.596002,17.445007,17.243006,17.443008z M7.9750035,16.133011C8.6020172,16.133011 9.1100128,16.641006 9.1100128,17.268005 9.1100128,17.893997 8.6020172,18.402008 7.9750035,18.402008 7.3489964,18.402008 6.8410013,17.893997 6.8410013,17.268005 6.8410013,16.641006 7.3489964,16.133011 7.9750035,16.133011z

M24.027,13.762009C24.654014,13.762009 25.16201,14.270004 25.16201,14.895996 25.16201,15.522003 24.654014,16.029999 24.027,16.029999 23.400993,16.029999 22.892998,15.522003 22.892998,14.895996 22.892998,14.270004 23.400993,13.762009 24.027,13.762009z M24.027,9.6110077C24.653007,9.6110077 25.161003,10.119003 25.161003,10.74501 25.161003,11.37001 24.653007,11.878006 24.027,11.878006 23.402,11.878006 22.894005,11.37001 22.894005,10.74501 22.894005,10.119003 23.402,9.6110077 24.027,9.6110077z

M24.027,5.6000061C24.654014,5.6000061 25.16201,6.1080017 25.16201,6.7350006 25.16201,7.3610077 24.654014,7.8690033 24.027,7.8690033 23.400993,7.8690033 22.892998,7.3610077 22.892998,6.7350006 22.892998,6.1080017 23.400993,5.6000061 24.027,5.6000061z

M19.876001,5.6000061C20.503013,5.6000061 21.011009,6.1080017 21.011009,6.7350006 21.011009,7.3610077 20.503013,7.8690033 19.876001,7.8690033 19.249994,7.8690033 18.743006,7.3610077 18.743006,6.7350006 18.743006,6.1080017 19.249994,5.6000061 19.876001,5.6000061z

M2.4290157,1.8740082C2.2420037,1.8740082,2.0850215,2.029007,2.0850215,2.2140045L2.0850215,9.5680084C2.0850215,9.753006,2.2420037,9.9069977,2.4290157,9.9069977L13.451014,9.9069977C13.637995,9.9069977,13.795008,9.753006,13.795008,9.5680084L13.795008,2.2140045C13.795008,2.029007,13.637995,1.8740082,13.451014,1.8740082z

M1.1730042,0L14.706996,0C15.353999,0.0019989014,15.877009,0.51899719,15.878993,1.1600037L15.878993,11.100006C15.877009,11.740005,15.353999,12.26001,14.706996,12.26001L9.5830047,12.26001C9.4399994,12.26001 9.3290069,12.382004 9.3420074,12.52301 9.3600128,12.692001 9.3829925,12.917999 9.4060028,13.117004 9.4349945,13.367004 9.6570099,13.53801 9.9049957,13.591003L11.684994,13.975998C11.835994,14.009003 11.945003,14.141998 11.945003,14.294998 11.945003,14.440002 11.826015,14.557007 11.679012,14.557007L11.499996,14.557007 4.3789966,14.557007 4.2470081,14.557007C4.1000049,14.557007 3.9819935,14.440002 3.9819937,14.294998 3.9819935,14.141998 4.0899952,14.009003

```

4.2409961,13.977005L6.0450113,13.589996C6.2860086,13.539001
6.501005,13.373001 6.5249918,13.130997 6.5460184,12.921005
6.5650003,12.688004 6.5769937,12.516998 6.5870035,12.376999
6.4710062,12.262009
6.3290079,12.262009L1.1730042,12.26001C0.52499391,12.26001,0.0020143806,11.7
40005,0,11.100006L0,1.1600037C0.0020143806,0.51899719,0.52499391,0.001998901
4,1.1730042,0z"/>' +
    "</g>",
    },
  },
];
return basicShapes;
}
export function getHtmlShapes() {
  let basicShapes = [
    {
      id: "HTML",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content: '<div style="height:100%;width:100%;"><button
type="button" style="width:100%;overflow:hidden">HTML</button></div>',
      },
    },
    {
      id: "Checkbox",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content: '<div><div style="height:50%;width:100%;"><input
type="checkbox" value="Yes" style="width:25%">Yes</input></div>' +
          '<div style="height:50%;width:100%;"><input
type="checkbox" value="No" style="width:25%">No</input></div></div>',
      },
    },
    {
      id: "Dropdown",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content: '<div style="height:100%;width:100%;"><select
style="width:100%"><option>SVG</option><option>Canvas</option></select></div>',
      },
    },
  ];
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
  let flowShapes = [
    {
      id: "process",
      shape: {

```



```

        type: "Flow",
        shape: "Process",
    },
},
{
    id: "document",
    shape: {
        type: "Flow",
        shape: "Document",
    },
},
{
    id: "predefinedprocess",
    shape: {
        type: "Flow",
        shape: "PreDefinedProcess",
    },
},
},
];
return flowShapes;
}
//Initializes connector symbols for the symbol palette
export function getConnectors() {
    let connectorSymbols = [
        {
            id: "Link1",
            type: "Orthogonal",
            sourcePoint: {
                x: 0,
                y: 0,
            },
            targetPoint: {
                x: 40,
                y: 40,
            },
            targetDecorator: {
                shape: "Arrow",
            },
        },
        {
            id: "Link21",
            type: "Straight",
            sourcePoint: {
                x: 0,
                y: 0,
            },
            targetPoint: {
                x: 40,
                y: 40,
            },
            targetDecorator: {
                shape: "Arrow",
            },
        },
        {
            id: "link33",
            type: "Bezier",

```

```

        sourcePoint: {
            x: 0,
            y: 0,
        },
        targetPoint: {
            x: 40,
            y: 40,
        },
        style: {
            strokeWidth: 2,
        },
        targetDecorator: {
            shape: "None",
        },
    },
    ],
    ];
    return connectorSymbols;
}
//Initializes the symbol palette
function App() {
    return (<SymbolPaletteComponent id="palette1"
        //Defines how many palettes can be at expanded mode at a time
        expandMode={"Multiple"}
        //Defines the palette collection
        palettes=[
            {
                //Sets the id of the palette
                id: "flow",
                //Sets whether the palette expands/collapse its children
                expanded: true,
                //Adds the palette items to palette
                symbols: getFlowShapes(),
                //Sets the header text of the palette
                title: "Flow Shapes",
                iconCss: "e-ddb-icons e-flow",
            },
            {
                id: "basic",
                expanded: true,
                symbols: getBasicShapes(),
                title: "Basic Shapes",
                iconCss: "e-ddb-icons e-basic",
            },
            {
                id: "svg",
                expanded: true,
                symbols: getSvgShapes(),
                title: "SVG Shapes",
                iconCss: "e-ddb-icons e-basic",
            },
            {
                id: "html",
                expanded: true,
                symbols: getHtmlShapes(),
                title: "HTML Shapes",
                iconCss: "e-ddb-icons e-basic",
            },
        ],
    );
}

```

```

        {
            id: "connectors",
            expanded: true,
            symbols: getConnectors(),
            title: "Connectors",
            iconCss: "e-ddb-icons e-connector",
        },
    ]}
    //Specifies the default size to render symbols
    symbolHeight={80} symbolWidth={80}/>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);

```

INDEX.TSX

```

import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    SymbolPalette,
    ConnectorModel,
    Connector,
    SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
    let basicShapes: NodeModel[] = [
        {
            id: "Rectangle",
            shape: {
                type: "Basic",
                shape: "Rectangle",
            },
        },
        {
            id: "Ellipse",
            shape: {
                type: "Basic",
                shape: "Ellipse",
            },
        },
        {
            id: "Hexagon",
            shape: {
                type: "Basic",
                shape: "Hexagon",
            },
        },
    ];
    return basicShapes;
}
export function getSvgShapes(): NodeModel[] {
    let basicShapes: NodeModel[] = [
        {

```

```

id: "node2",
style: { fill: "none" },
annotations: [{ content: "Start \n Text Editing" }],
shape: {
  type: "Native",
  content:
    '<g xmlns="http://www.w3.org/2000/svg">      <g
transform="translate(1 1)">      <g>      <path
style="fill:#61443C;" d="M61.979,435.057c2.645-0.512,5.291-0.853,7.936-
1.109c-2.01,1.33-4.472,1.791-6.827,1.28
C62.726,435.13,62.354,435.072,61.979,435.057z"/>      <path
style="fill:#61443C;" d="M502.469,502.471h-25.6c0.163-30.757-20.173-57.861-
49.749-66.304      c-5.784-1.581-11.753-2.385-17.749-2.389c-2.425-0.028-
4.849,0.114-7.253,0.427c1.831-7.63,2.747-15.45,2.731-23.296      c0.377-
47.729-34.52-88.418-81.749-95.317c4.274-0.545,8.577-0.83,12.885-
0.853c25.285,0.211,49.448,10.466,67.167,28.504
c17.719,18.039,27.539,42.382,27.297,67.666c0.017,7.846-0.9,15.666-
2.731,23.296c2.405-0.312,4.829-0.455,7.253-0.427
C472.572,434.123,502.783,464.869,502.469,502.471z"/>      </g>
      <path style="fill:#8B685A;" d="M476.869,502.471H7.536c-0.191-
32.558,22.574-60.747,54.443-67.413
c0.375,0.015,0.747,0.072,1.109,0.171c2.355,0.511,4.817,0.05,6.827-
1.28c1.707-0.085,3.413-0.171,5.12-0.171
c4.59,0,9.166,0.486,13.653,1.451c2.324,0.559,4.775,0.147,6.787-1.141c2.013-
1.288,3.414-3.341,3.879-5.685      c7.68-39.706,39.605-70.228,79.616-
76.117c4.325-0.616,8.687-0.929,13.056-0.939c13.281-
0.016,26.409,2.837,38.485,8.363
c3.917,1.823,7.708,3.904,11.349,6.229c2.039,1.304,4.527,1.705,6.872,1.106c2.
345-0.598,4.337-2.142,5.502-4.264      c14.373-25.502,39.733-42.923,68.693-
47.189h0.171c47.229,6.899,82.127,47.588,81.749,95.317c0.017,7.846-
0.9,15.666-2.731,23.296      c2.405-0.312,4.829-0.455,7.253-
0.427c5.996,0.005,11.965,0.808,17.749,2.389C456.696,444.61,477.033,471.713,4
76.869,502.471      L476.869,502.471z"/>      <path style="fill:#66993E;"
d="M502.469,7.537c0,0-6.997,264.96-192.512,252.245c-20.217-1.549-40.166-
5.59-59.392-12.032      c-1.365-0.341-2.731-0.853-4.096-1.28c0,0-0.597-2.219-
1.451-6.144c-6.656-34.048-25.088-198.997,231.765-230.144
C485.061,9.159,493.595,8.22,502.469,7.537z"/>      <path
style="fill:#9ACA5C;" d="M476.784,10.183c-1.28,26.197-16.213,238.165-
166.827,249.6      c-20.217-1.549-40.166-5.59-59.392-12.032c-1.365-0.341-
2.731-0.853-4.096-1.28c0,0-0.597-2.219-1.451-6.144
C238.363,206.279,219.931,41.329,476.784,10.183z"/>      <path
style="fill:#66993E;" d="M206.192,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-21.505,7.427-
44.293,10.417-66.987,8.789C21.104,252.103,8.816,94.236,7.621,71.452c-0.085-
1.792-0.085-2.731-0.085-2.731
C222.747,86.129,211.653,216.689,206.192,246.727z"/>      <path
style="fill:#9ACA5C;" d="M180.336,246.727c-0.768,3.925-1.365,6.144-
1.365,6.144c-1.365,0.427-2.731,0.939-4.096,1.28      c-13.351,4.412-
27.142,7.359-41.131,8.789C21.104,252.103,8.816,94.236,7.621,71.452
C195.952,96.881,185.541,217.969,180.336,246.727z"/>      </g>      <g>
      <path d="M162.136,426.671c3.451-0.001,6.562-2.08,7.882-5.268s0.591-
6.858-1.849-9.2981-8.533-8.533      c-3.341-3.281-8.701-3.256-12.012,0.054c-
3.311,3.311-3.335,8.671-0.054,12.01218.533,8.533
C157.701,425.773,159.872,426.673,162.136,426.671L162.136,426.671z"/>
      <path d="M292.636,398.57c3.341,3.281,8.701,3.256,12.012-0.054c3.311-
3.311,3.335-8.671,0.054-12.0121-8.533-8.533      c-3.341-3.281-8.701-3.256-
12.012,0.054s-3.335,8.671-0.054,12.012L292.636,398.57z"/>      </path

```

```
d="M296.169,454.771c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-12.012L296.169,454.771z"/>
    <path d="M386.503,475.37c3.341,3.281,8.701,3.256,12.012-0.054c3.311-3.311,3.335-8.671,0.054-12.012l8.533-8.533c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L386.503,475.37z"/>
    <path d="M204.803,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C198.241,407.524,201.352,409.603,204.803,409.604z"/>
    <path d="M332.803,443.737c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C326.241,441.658,329.352,443.737,332.803,443.737z"/>
    <path d="M341.336,366.937c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C334.774,364.858,337.885,366.937,341.336,366.937z"/>
    <path d="M164.636,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C173.337,451.515,167.977,451.49,164.636,454.771L164.636,454.771z"/>
    <path d="M232.903,429.171l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C241.604,425.915,236.243,425.89,232.903,429.171L232.903,429.171z"/>
    <path d="M384.003,409.604c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012c-3.311-3.311-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-1.849,9.298C377.441,407.524,380.552,409.603,384.003,409.604z"/>
    <path d="M70.77,463.304l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271s3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C79.47,460.048,74.11,460.024,70.77,463.304L70.77,463.304z"/>
    <path d="M121.97,446.238l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C130.67,442.981,125.31,442.957,121.97,446.238L121.97,446.238z"/>
    <path d="M202.302,420.638c-1.6-1.601-3.77-2.5-6.033-2.5c-2.263,0-4.433,0.899-6.033,2.51-8.533,8.533c-2.178,2.151-3.037,5.304-2.251,8.262c0.786,2.958,3.097,5.269,6.055,6.055c2.958,0.786,6.111-0.073,8.262-2.251l8.533-8.533c1.601-1.6,2.5-3.77,2.5-6.033C204.802,424.408,203.903,422.237,202.302,420.638L202.302,420.638z"/>
    <path d="M210.836,463.304c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012l8.533,8.533c2.149,2.188,5.307,3.055,8.271,2.27c2.965-0.785,5.28-3.1,6.065-6.065c0.785-2.965-0.082-6.122-2.27-8.271L210.836,463.304z"/>
    <path d="M343.836,454.771l-8.533,8.533c-2.188,2.149-3.055,5.307-2.27,8.271c0.785,2.965,3.1,5.28,6.065,6.065c2.965,0.785,6.122-0.082,8.271-2.27l8.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012C352.537,451.515,347.177,451.49,343.836,454.771L343.836,454.771z"/>
    <path d="M429.17,483.904c3.341,3.281,8.701,3.256,12.012-0.054s3.335-8.671,0.054-12.012l8.533-8.533c-3.341-3.281-8.701-3.256-12.012,0.054c-3.311,3.311-3.335,8.671-0.054,12.012L429.17,483.904z"/>
    <path d="M341.336,401.071c2.264,0.003,4.435-0.897,6.033-2.518.533-8.533c3.281-3.341,3.256-8.701-0.054-12.012s-8.671-3.335-12.012-0.054l-8.533,8.533c-2.44,2.44-3.169,6.11-
```

```

1.849,9.298C334.774,398.991,337.885,401.07,341.336,401.071z"/>
    <path d="M273.069,435.204c2.264,0.003,4.435-0.897,6.033-2.518.533-
8.533c3.281-3.341,3.256-8.701-0.054-12.012    s-8.671-3.335-12.012-0.054l-
8.533,8.533c-2.44,2.44-3.169,6.11-
1.849,9.298C266.508,433.124,269.618,435.203,273.069,435.204z"/>
    <path
d="M253.318,258.138c22.738,7.382,46.448,11.338,70.351,11.737c31.602,0.543,62
.581-8.828,88.583-26.796    c94.225-65.725,99.567-227.462,99.75-
234.317c0.059-2.421-0.91-4.754-2.667-6.421c-1.751-1.679-4.141-2.52-6.558-
2.308    C387.311,9.396,307.586,44.542,265.819,104.5c-28.443,42.151-
38.198,94.184-26.956,143.776c-3.411,8.366-6.04,17.03-7.852,25.881    c-
4.581-7.691-9.996-14.854-16.147-21.358c8.023-38.158,0.241-77.939-21.57-
110.261C160.753,95.829,98.828,68.458,9.228,61.196    c-2.417-0.214-
4.808,0.628-6.558,2.308c-1.757,1.667-2.726,4-
2.667,6.421c0.142,5.321,4.292,130.929,77.717,182.142
c20.358,14.081,44.617,21.428,69.367,21.008c18.624-0.309,37.097-3.388,54.814-
9.138c11.69,12.508,20.523,27.407,25.889,43.665
c0.149,15.133,2.158,30.19,5.982,44.832c-12.842-5.666-26.723-8.595-40.759-
8.6c-49.449,0.497-91.788,35.567-101.483,84.058    c-5.094-1.093-10.29-1.641-
15.5-1.638c-42.295,0.38-76.303,34.921-76.025,77.217c-
0.001,2.263,0.898,4.434,2.499,6.035
c1.6,1.6,3.771,2.499,6.035,2.499h494.933c2.263,0.001,4.434-0.898,6.035-
2.499c1.6-1.6,2.499-3.771,2.499-6.035    c0.249-41.103-31.914-75.112-72.967-
77.154c0.65-4.78,0.975-9.598,0.975-14.421c0.914-45.674-28.469-86.455-72.083-
100.045    c-43.615-13.59-90.962,3.282-
116.154,41.391C242.252,322.17,242.793,288.884,253.318,258.138L253.318,258.13
8z M87.519,238.092    c-55.35-38.567-67.358-129.25-69.833-
158.996c78.8,7.921,133.092,32.454,161.458,72.992
c15.333,22.503,22.859,49.414,21.423,76.606c-23.253-35.362-77.83-105.726-
162.473-140.577c-2.82-1.165-6.048-0.736-8.466,1.125    s-3.658,4.873-
3.252,7.897c0.406,3.024,2.395,5.602,5.218,6.761c89.261,36.751,144.772,117.77
6,161.392,144.874    C150.795,260.908,115.29,257.451,87.519,238.092z
M279.969,114.046c37.6-53.788,109.708-86.113,214.408-96.138    c-2.65,35.375-
17.158,159.05-91.892,211.175c-37.438,26.116-85.311,30.57-142.305,13.433
c19.284-32.09,92.484-142.574,212.405-191.954c2.819-1.161,4.805-3.738,5.209-
6.76c0.404-3.022-0.835-6.031-3.25-7.892    c-2.415-1.861-5.64-2.292-8.459-
1.131C351.388,82.01,279.465,179.805,252.231,222.711
C248.573,184.367,258.381,145.945,279.969,114.046L279.969,114.046z
M262.694,368.017c15.097-26.883,43.468-43.587,74.3-43.746
c47.906,0.521,86.353,39.717,85.95,87.625c-0.001,7.188-0.857,14.351-
2.55,21.337c-0.67,2.763,0.08,5.677,1.999,7.774
c1.919,2.097,4.757,3.1,7.568,2.676c1.994-0.272,4.005-0.393,6.017-
0.362c29.59,0.283,54.467,22.284,58.367,51.617H17.661    c3.899-
29.333,28.777-51.334,58.367-51.617c4-
0.004,7.989,0.416,11.9,1.254c4.622,0.985,9.447,0.098,13.417-2.467    c3.858-
2.519,6.531-6.493,7.408-11.017c7.793-40.473,43.043-69.838,84.258-
70.192c16.045-0.002,31.757,4.582,45.283,13.212
c4.01,2.561,8.897,3.358,13.512,2.205C256.422,375.165,260.36,372.163,262.694,
368.017L262.694,368.017z"/>    </g></g>',
    },
  },
  {
    id: "syncfusion",
    style: { fill: "none" },
    shape: {
      type: "Native",
      content:

```

```

    '<g xmlns="http://www.w3.org/2000/svg">' +
    '<rect height="256" width="256" fill="#34353F"/>' +
    '<path id="path1" transform="rotate(0,128,128)
translate(59,61.2230899333954) scale(4.3125,4.3125) " fill="#FFFFFF"
d="M18.88501,23.042998L26.804993,23.042998 26.804993,30.969001
18.88501,30.969001z M9.4360352,23.042998L17.358032,23.042998
17.358032,30.969001 9.4360352,30.969001z
M0.014038086,23.042998L7.9360352,23.042998 7.9360352,30.969001
0.014038086,30.969001z M18.871033,13.609001L26.791016,13.609001
26.791016,21.535994 18.871033,21.535994z
M9.4219971,13.609001L17.342041,13.609001 17.342041,21.535994
9.4219971,21.535994z M0,13.609001L7.9219971,13.609001 7.9219971,21.535994
0,21.535994z M9.4219971,4.1859968L17.342041,4.1859968 17.342041,12.113998
9.4219971,12.113998z M0,4.1859968L7.9219971,4.1859968 7.9219971,12.113998
0,12.113998z M25.846008,0L32,5.2310026 26.773987,11.382995
20.619019,6.155998z"/>' +
    "</g>",
  },
},
{
  id: "network",
  style: { fill: "none" },
  shape: {
    type: "Native",
    content:
      '<g xmlns="http://www.w3.org/2000/svg">' +
      '<rect height="256" width="256" fill="#34353F"/>' +
      '<path id="path1" transform="rotate(0,128,128)
translate(59.1078108549118,59) scale(4.3125,4.3125) " fill="#FFFFFF"
d="M12.12701,24.294998C12.75201,24.294998 13.258998,24.803009
13.258998,25.428009 13.258998,26.056 12.75201,26.563004 12.12701,26.563004
11.499019,26.563004 10.993007,26.056 10.993007,25.428009 10.993007,24.803009
11.499019,24.294998 12.12701,24.294998z
M7.9750035,24.294998C8.6010101,24.294998 9.1090057,24.803009
9.1090057,25.428009 9.1090057,26.056 8.6010101,26.563004 7.9750035,26.563004
7.3480199,26.563004 6.8399942,26.056 6.8399942,25.428009 6.8399942,24.803009
7.3480199,24.294998 7.9750035,24.294998z
M7.9750035,20.286011C8.6010101,20.286011 9.1090057,20.792999
9.1090057,21.419006 9.1090057,22.044006 8.6010101,22.552002
7.9750035,22.552002 7.3500035,22.552002 6.8420084,22.044006
6.8420084,21.419006 6.8420084,20.792999 7.3500035,20.286011
7.9750035,20.286011z
M18.499994,19.317001C18.313013,19.317001,18.156,19.472,18.156,19.656006L18.1
56,27.01001C18.156,27.195007,18.313013,27.350006,18.499994,27.350006L29.5219
93,27.350006C29.707998,27.350006,29.865988,27.195007,29.865988,27.01001L29.8
65988,19.656006C29.865988,19.472,29.707998,19.317001,29.521993,19.317001z
M17.243006,17.443008L30.778003,17.443008C31.425007,17.445007,31.947986,17.96
2006,31.950001,18.602997L31.950001,28.542007C31.947986,29.182999,31.425007,2
9.702011,30.778003,29.703003L25.654012,29.703003C25.511007,29.703003
25.399008,29.824997 25.413992,29.964996 25.430013,30.13501
25.452993,30.360001 25.477011,30.559998 25.506002,30.809998
25.727987,30.980011
25.976003,31.033997L27.756002,31.419006C27.907003,31.452011 28.015005,31.584
28.015005,31.738007 28.015005,31.883011 27.895986,32
27.74999,32L27.571005,32 20.450004,32 20.318016,32C20.171013,32
20.053001,31.883011 20.053001,31.738007 20.053001,31.585007
20.161003,31.452011

```

```

20.312004,31.419998L22.115989,31.033005C22.35601,30.98201
22.572014,30.815002 22.596,30.574005 22.616997,30.363007 22.636009,30.130997
22.648002,29.960007 22.658012,29.819 22.542015,29.70401
22.399986,29.70401L17.243006,29.703003C16.596002,29.702011,16.072992,29.1829
99,16.071008,28.542007L16.071008,18.602997C16.072992,17.962006,16.596002,17.
445007,17.243006,17.443008z M7.9750035,16.133011C8.6020172,16.133011
9.1100128,16.641006 9.1100128,17.268005 9.1100128,17.893997
8.6020172,18.402008 7.9750035,18.402008 7.3489964,18.402008
6.8410013,17.893997 6.8410013,17.268005 6.8410013,16.641006
7.3489964,16.133011 7.9750035,16.133011z
M24.027,13.762009C24.654014,13.762009 25.16201,14.270004 25.16201,14.895996
25.16201,15.522003 24.654014,16.029999 24.027,16.029999 23.400993,16.029999
22.892998,15.522003 22.892998,14.895996 22.892998,14.270004
23.400993,13.762009 24.027,13.762009z M24.027,9.6110077C24.653007,9.6110077
25.161003,10.119003 25.161003,10.74501 25.161003,11.37001
24.653007,11.878006 24.027,11.878006 23.402,11.878006 22.894005,11.37001
22.894005,10.74501 22.894005,10.119003 23.402,9.6110077 24.027,9.6110077z
M24.027,5.6000061C24.654014,5.6000061 25.16201,6.1080017 25.16201,6.7350006
25.16201,7.3610077 24.654014,7.8690033 24.027,7.8690033 23.400993,7.8690033
22.892998,7.3610077 22.892998,6.7350006 22.892998,6.1080017
23.400993,5.6000061 24.027,5.6000061z
M19.876001,5.6000061C20.503013,5.6000061 21.011009,6.1080017
21.011009,6.7350006 21.011009,7.3610077 20.503013,7.8690033
19.876001,7.8690033 19.249994,7.8690033 18.743006,7.3610077
18.743006,6.7350006 18.743006,6.1080017 19.249994,5.6000061
19.876001,5.6000061z
M2.4290157,1.8740082C2.2420037,1.8740082,2.0850215,2.029007,2.0850215,2.2140
045L2.0850215,9.5680084C2.0850215,9.753006,2.2420037,9.9069977,2.4290157,9.9
069977L13.451014,9.9069977C13.637995,9.9069977,13.795008,9.753006,13.795008,
9.5680084L13.795008,2.2140045C13.795008,2.029007,13.637995,1.8740082,13.4510
14,1.8740082z
M1.1730042,0L14.706996,0C15.353999,0.0019989014,15.877009,0.51899719,15.8789
93,1.1600037L15.878993,11.100006C15.877009,11.740005,15.353999,12.26001,14.7
06996,12.26001L9.5830047,12.26001C9.4399994,12.26001 9.3290069,12.382004
9.3420074,12.52301 9.3600128,12.692001 9.3829925,12.917999
9.4060028,13.117004 9.4349945,13.367004 9.6570099,13.53801
9.9049957,13.591003L11.684994,13.975998C11.835994,14.009003
11.945003,14.141998 11.945003,14.294998 11.945003,14.440002
11.826015,14.557007 11.679012,14.557007L11.499996,14.557007
4.3789966,14.557007 4.2470081,14.557007C4.1000049,14.557007
3.9819935,14.440002 3.9819937,14.294998 3.9819935,14.141998
4.0899952,14.009003
4.2409961,13.977005L6.0450113,13.589996C6.2860086,13.539001
6.501005,13.373001 6.5249918,13.130997 6.5460184,12.921005
6.5650003,12.688004 6.5769937,12.516998 6.5870035,12.376999
6.4710062,12.262009
6.3290079,12.262009L1.1730042,12.26001C0.52499391,12.26001,0.0020143806,11.7
40005,0,11.100006L0,1.1600037C0.0020143806,0.51899719,0.52499391,0.001998901
4,1.1730042,0z"/>' +
    "</g>",
    },
  },
];
return basicShapes;
}
export function getHtmlShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [

```



```

    {
      id: "HTML",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content:
          '<div style="height:100%;width:100%;"><button type="button"
style="width:100%;overflow:hidden">HTML</button></div>',
      },
    },
    {
      id: "Checkbox",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content:
          '<div><div style="height:50%;width:100%;"><input type="checkbox"
value="Yes" style="width:25%">Yes</input></div>' +
          '<div style="height:50%;width:100%;"><input type="checkbox"
value="No" style="width:25%">No</input></div></div>',
      },
    },
    {
      id: "Dropdown",
      borderColor: "black",
      borderWidth: 1,
      shape: {
        type: "HTML",
        content:
          '<div style="height:100%;width:100%;"><select
style="width:100%"><option>SVG</option><option>Canvas</option></select></div>',
      },
    },
  ];
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
  let flowShapes: NodeModel[] = [
    {
      id: "process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ],
  {

```

```

        id: "predefinedprocess",
        shape: {
            type: "Flow",
            shape: "PreDefinedProcess",
        },
    },
];
return flowShapes;
}
//Initializes connector symbols for the symbol palette
export function getConnectors(): ConnectorModel[] {
    let connectorSymbols: ConnectorModel[] = [
        {
            id: "Link1",
            type: "Orthogonal",
            sourcePoint: {
                x: 0,
                y: 0,
            },
            targetPoint: {
                x: 40,
                y: 40,
            },
            targetDecorator: {
                shape: "Arrow",
            },
        },
        {
            id: "Link21",
            type: "Straight",
            sourcePoint: {
                x: 0,
                y: 0,
            },
            targetPoint: {
                x: 40,
                y: 40,
            },
            targetDecorator: {
                shape: "Arrow",
            },
        },
        {
            id: "link33",
            type: "Bezier",
            sourcePoint: {
                x: 0,
                y: 0,
            },
            targetPoint: {
                x: 40,
                y: 40,
            },
            style: {
                strokeWidth: 2,
            },
            targetDecorator: {

```

```

        shape: "None",
      },
    ],
  ];
  return connectorSymbols;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="palette1"
      //Defines how many palettes can be at expanded mode at a time
      expandMode={"Multiple"}
      //Defines the palette collection
      palettes=[
        {
          //Sets the id of the palette
          id: "flow",
          //Sets whether the palette expands/collapse its children
          expanded: true,
          //Adds the palette items to palette
          symbols: getFlowShapes(),
          //Sets the header text of the palette
          title: "Flow Shapes",
          iconCss: "e-ddb-icons e-flow",
        },
        {
          id: "basic",
          expanded: true,
          symbols: getBasicShapes(),
          title: "Basic Shapes",
          iconCss: "e-ddb-icons e-basic",
        },
        {
          id: "svg",
          expanded: true,
          symbols: getSvgShapes(),
          title: "SVG Shapes",
          iconCss: "e-ddb-icons e-basic",
        },
        {
          id: "html",
          expanded: true,
          symbols: getHtmlShapes(),
          title: "HTML Shapes",
          iconCss: "e-ddb-icons e-basic",
        },
        {
          id: "connectors",
          expanded: true,
          symbols: getConnectors(),
          title: "Connectors",
          iconCss: "e-ddb-icons e-connector",
        },
      ],
    )
  //Specifies the default size to render symbols
  symbolHeight={80}

```

```

        symbolWidth={80}
      />
    );
  }
  const root = ReactDOM.createRoot(document.getElementById("diagram"));
  root.render(<App />);

```

Customize the palette header

Palettes can be annotated with its header texts.

The `title` displayed as the header text of palette.

The `expanded` property of palette allows to expand/collapse its palette items.

The `height` property of palette sets the height of the symbol group.

The `iconCss` property sets the content of the symbol group.

The `description` defines the text to be displayed and how that is to be handled in `getSymbolInfo`.

Also, any HTML element into a palette header can be embedded by defining the `getSymbolInfo` property.

The following code example illustrates how to customize palette headers.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
  return basicShapes;
}

```

```

}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={"Multiple"}
  palettes=[
    {
      id: "basic",
      expanded: true,
      symbols: getBasicShapes(),
      title: "Basic Shapes",
      iconCss: "e-ddb-icons e-basic",
    },
  ] symbolHeight={80} symbolWidth={80}
  //Sets the size, appearance and description of a symbol
  getSymbolInfo=(symbol) => {
    if (symbol["text"] !== undefined) {
      return {
        width: 75,
        height: 40,
        //Add or Remove the Text for Symbol palette item.
        description: {
          //Defines the symbol description
          text: symbol["text"],
          //Defines how to handle the text when its size
          exceeds the given symbol size
          overflow: "Wrap",
        },
      };
    }
    return {
      width: 75,
      height: 40,
      description: {
        text: symbol.shape["shape"],
      },
    };
  }
  }>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo,
  Node,
  ConnectorModel,
  SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";

```

```

//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
  return basicShapes;
}

//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={"Multiple"}
      palettes=[
        {
          id: "basic",
          expanded: true,
          symbols: getBasicShapes(),
          title: "Basic Shapes",
          iconCss: "e-ddb-icons e-basic",
        },
      ]
    )
    symbolHeight={80}
    symbolWidth={80}
    //Sets the size, appearance and description of a symbol
    getSymbolInfo=(symbol) => {
      if (symbol["text"] !== undefined) {
        return {
          width: 75,
          height: 40,
          //Add or Remove the Text for Symbol palette item.
          description: {
            //Defines the symbol description
            text: symbol["text"],
            //Defines how to handle the text when its size exceeds the
            given symbol size
            overflow: "Wrap",

```

```

    },
  };
}
return {
  width: 75,
  height: 40,
  description: {
    text: symbol.shape["shape"],
  },
};
}}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

Restrict expansion of the palette panel

The symbol palette panel can be restricted from getting expanded. The `cancel` argument of the `paletteExpanding` property defines whether the palette's panel should be expanded or collapsed. By default, the panel is expanded. This restriction can be done for each of the palettes in the symbol palette as desired.

In the following code example, the basic shapes palette is restricted from getting collapsed whereas the swimlane shapes palette can be expanded or collapsed.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
}

```

```

    },
  ],
  ];
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
  let flowShapes = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ];
  return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={"Multiple"}
  palettes=[
    {
      id: "basic",
      expanded: true,
      symbols: getBasicShapes(),
      title: "Basic Shapes",
      iconCss: "e-ddb-icons e-basic",
    },
    {
      id: "flow",
      expanded: true,
      symbols: getFlowShapes(),
      title: "Flow Shapes",
      iconCss: "e-ddb-icons e-flow",
    },
  ]) symbolHeight={80} symbolWidth={80} paletteExpanding={function
  (args) {
    if (args.palette.id === "basic") {
      // Basic shapes panel does not collapse
      args.cancel = true;
    }
    else {
      // Swimlane shapes panel collapse and expand
      args.cancel = false;
    }
  }}
  //Sets the size, appearance and description of a symbol
  getSymbolInfo=(symbol) => {
    if (symbol["text"] !== undefined) {

```



```

        return {
            width: 75,
            height: 40,
            //Add or Remove the Text for Symbol palette item.
            description: {
                //Defines the symbol description
                text: symbol["text"],
                //Defines how to handle the text when its size
                //exceeds the given symbol size
                overflow: "Wrap",
            },
        };
    }
    return {
        width: 75,
        height: 40,
        description: {
            text: symbol.shape["shape"],
        },
    };
}
}}/>);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    SymbolPalette,
    SymbolInfo,
    Node,
    SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basic shapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
    let basicShapes: NodeModel[] = [
        {
            id: "Rectangle",
            shape: {
                type: "Basic",
                shape: "Rectangle",
            },
        },
        {
            id: "Ellipse",
            shape: {
                type: "Basic",
                shape: "Ellipse",
            },
        },
    ],
}

```

```

    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ],
];
return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
  let flowShapes: NodeModel[] = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ],
];
return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={"Multiple"}
      palettes=[
        {
          id: "basic",
          expanded: true,
          symbols: getBasicShapes(),
          title: "Basic Shapes",
          iconCss: "e-ddb-icons e-basic",
        },
        {
          id: "flow",
          expanded: true,
          symbols: getFlowShapes(),
          title: "Flow Shapes",
          iconCss: "e-ddb-icons e-flow",
        },
      ],
    >
    symbolHeight={80}
    symbolWidth={80}
    paletteExpanding={function (args) {
      if (args.palette.id === "basic") {
        // Basic shapes panel does not collapse

```

```

        args.cancel = true;
      } else {
        // Swimlane shapes panel collapse and expand
        args.cancel = false;
      }
    }
  }
  //Sets the size, appearance and description of a symbol
  getSymbolInfo=(symbol: Node | Connector): SymbolInfo => {
    if (symbol["text"] !== undefined) {
      return {
        width: 75,
        height: 40,
        //Add or Remove the Text for Symbol palette item.
        description: {
          //Defines the symbol description
          text: symbol["text"],
          //Defines how to handle the text when its size exceeds the
given symbol size
          overflow: "Wrap",
        },
      };
    }
    return {
      width: 75,
      height: 40,
      description: {
        text: symbol.shape["shape"],
      },
    };
  }
}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById("diagram"));
root.render(<App />);
{% enddraw %}

```

Stretch the symbols into the palette

The [fit](#) property defines whether the symbol has to be fit inside the size, that is defined by the symbol palette. For example, when you resize the rectangle in the symbol, ratio of the rectangle size has to be maintained rather changing into square shape. The following code example illustrates how to customize the symbol size.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",

```

```

        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ],
};
return basicShapes;
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
    palettes=[
      {
        id: 'basic',
        expanded: true,
        symbols: getBasicShapes(),
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic',
      },
    ]) symbolHeight={80} symbolWidth={80}
    //Sets the size, appearance and description of a symbol
    getSymbolInfo={function (symbol) {
      // Enables to fit the content into the specified palette item
      size
      return {
        fit: true,
      };
      // When it is set as false, the element is rendered with actual
      node size
    }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,

```

```

SymbolInfo,
SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
  return basicShapes;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={'Multiple'}
      palettes={[
        {
          id: 'basic',
          expanded: true,
          symbols: getBasicShapes(),
          title: 'Basic Shapes',
          iconCss: 'e-ddb-icons e-basic',
        },
      ]}
      symbolHeight={80}
      symbolWidth={80}
      //Sets the size, appearance and description of a symbol
      getSymbolInfo={function (symbol: NodeModel) {
        // Enables to fit the content into the specified palette item size
        return {
          fit: true,
        };
        // When it is set as false, the element is rendered with actual node
size
      }}
    />

```

```

    );
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% enddraw %}

```

Add/Remove symbols to palette at runtime

- Symbols can be added to palette at runtime by using public method, [addPalettItem](#).
- Symbols can be removed from palette at runtime by using public method, [removePalettItem](#).

Customize the size of symbols

The size of the individual symbol can be customized. The [symbolWidth](#) and [symbolHeight](#) properties of node enables you to define the size of the symbols. The following code example illustrates how to change the size of a symbol.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
  return basicShapes;
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
  palettes=[
    {
      id: 'basic',

```

```

        expanded: true,
        symbols: getBasicShapes(),
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic',
      },
    ]}
    //Specifies the size of the symbol
    symbolHeight={80} symbolWidth={80}
    //Sets the space to be left around a symbol
    symbolMargin={{
      left: 15,
      right: 15,
      top: 15,
      bottom: 15,
    }}/>;
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo,
  ConnectorModel,
  Connector,
  Node,
  SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",

```

```

        shape: "Hexagon",
      },
    ],
  ];
  return basicShapes;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={'Multiple'}
      palettes=[
        {
          id: 'basic',
          expanded: true,
          symbols: getBasicShapes(),
          title: 'Basic Shapes',
          iconCss: 'e-ddb-icons e-basic',
        },
      ]
    >
//Specifies the size of the symbol
    symbolHeight={80}
    symbolWidth={80}
//Sets the space to be left around a symbol
    symbolMargin={{
      left: 15,
      right: 15,
      top: 15,
      bottom: 15,
    }}
  </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

The [symbolMargin](#) property is used to create the space around elements, outside of any defined borders.

Symbol preview

The symbol preview size of the palette items can be customized using [symbolPreview](#).

The [width](#) and [height](#) properties of SymbolPalette enables you to define the preview size to all the symbol palette items.

The [offset](#) of the dragging helper relative to the mouse cursor.

The following code example illustrates how to change the preview size of a palette item.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";

```



```

import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
    let basicShapes = [
        {
            id: "Rectangle",
            shape: {
                type: "Basic",
                shape: "Rectangle",
            },
        },
        {
            id: "Ellipse",
            shape: {
                type: "Basic",
                shape: "Ellipse",
            },
        },
        {
            id: "Hexagon",
            shape: {
                type: "Basic",
                shape: "Hexagon",
            },
        },
    ];
    return basicShapes;
}
//Initializes the symbol palette
function App() {
    return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
palettes=[
        {
            id: 'basic',
            expanded: true,
            symbols: getBasicShapes(),
            title: 'Basic Shapes',
            iconCss: 'e-ddb-icons e-basic',
        },
    ]) symbolHeight={80} symbolWidth={80}
//Specifies the preview size and position to symbol palette items.
symbolPreview={
    height: 100,
    width: 100,
    offset: {
        x: 0.5,
        y: 0.5,
    },
}
//Sets the margin of the dragging helper relative to the mouse cursor
symbolMargin={
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
}
getSymbolInfo=(symbol) => {
    return {

```

```

        fit: true,
      };
    }>);
  }
  const root = ReactDOM.createRoot(document.getElementById('diagram'));
  root.render(<App />);
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo,
  SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
      },
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
      },
    },
  ];
  return basicShapes;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={'Multiple'}
      palettes=[
        {
          id: 'basic',

```

```

        expanded: true,
        symbols: getBasicShapes(),
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic',
    },
  ],
  symbolHeight={80}
  symbolWidth={80}
  //Specifies the preview size and position to symbol palette items.
  symbolPreview={{
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5,
    },
  }}
  //Sets the margin of the dragging helper relative to the mouse cursor
  symbolMargin={{
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
  }}
  getSymbolInfo=(symbol) => {
    return {
      fit: true,
    };
  }
}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Default settings

While adding more number of symbols such as nodes and connectors to the palette, define the default settings for those objects through the [getNodeDefaults](#) and the [getConnectorDefaults](#) properties of diagram allows to define the default settings for nodes and connectors.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
  let flowShapes = [
    {
      id: "process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
  ],

```

```

    },
    {
      id: "document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ],
  {
    id: "predefinedprocess",
    shape: {
      type: "Flow",
      shape: "PreDefinedProcess",
    },
  },
],
];
return flowShapes;
}
function setPaletteNodeDefaults(node) {
  node.width = 100;
  node.height = 100;
  node.style.strokeColor = "#3A3A3A";
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={"Multiple"}
  palettes=[
    {
      id: "flow",
      expanded: true,
      symbols: getFlowShapes(),
      title: "Flow Shapes",
    },
  ],
  symbolPreview={{
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5,
    },
  },
  symbolMargin={{
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
  }}
  //Returns the default properties of node
  getNodeDefaults={setPaletteNodeDefaults} getSymbolInfo={(symbol) => {
    return {
      fit: true,
    };
  }}/>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  NodeModel,
  SymbolPalette,
  SymbolInfo,
  SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
  let flowShapes: NodeModel[] = [
    {
      id: "process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
    {
      id: "predefinedprocess",
      shape: {
        type: "Flow",
        shape: "PreDefinedProcess",
      },
    },
  ];
  return flowShapes;
}
function setPaletteNodeDefaults(node: NodeModel): void {
  node.width = 100;
  node.height = 100;
  node.style.strokeColor = "#3A3A3A";
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={"Multiple"}
      palettes={[
        {
          id: "flow",
          expanded: true,
          symbols: getFlowShapes(),

```

```

        title: "Flow Shapes",
      },
    ],
  ]}
  symbolPreview={{
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5,
    },
  }}
  symbolMargin={{
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
  }}
  //Returns the default properties of node
  getNodeDefaults={setPaletteNodeDefaults}
  getSymbolInfo={(symbol: NodeModel): SymbolInfo => {
    return {
      fit: true,
    };
  }}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

Adding symbol description for symbols in the palette

The diagram provides support to add symbol description below each symbol of a palette. This descriptive representation of each symbol will enhance the details of the symbol visually. The height and width of the symbol description can also be set individually.

- The property `getSymbolInfo`, can be used to add the symbol description at runtime.

The following code is an example to set a symbol description for symbols in the palette.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
        symbolInfo: { description: { text: "Rectangle" } },
      },
    },
  ],

```

```

    },
  },
  {
    id: "Ellipse",
    shape: {
      type: "Basic",
      shape: "Ellipse",
      symbolInfo: { description: { text: "Ellipse" } } },
  },
},
{
  id: "Hexagon",
  shape: {
    type: "Basic",
    shape: "Hexagon",
    symbolInfo: { description: { text: "Hexagon" } } },
},
];
return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
  let flowShapes = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
        symbolInfo: { description: { text: "Process" } } },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
        symbolInfo: { description: { text: "Document" } } },
    },
  ];
  return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
palettes=[
  {
    id: 'basic',
    expanded: true,
    symbols: getBasicShapes(),
    title: 'Basic Shapes',
    iconCss: 'e-ddb-icons e-basic',
  },
  {
    id: 'flow',
    expanded: true,

```

```

        symbols: getFlowShapes(),
        title: 'Flow Shapes',
        iconCss: 'e-ddb-icons e-flow',
    },
    ]] symbolHeight={80} symbolWidth={80}
    //Specifies the preview size and position to symbol palette items.
    symbolPreview={{
        height: 100,
        width: 100,
        offset: {
            x: 0.5,
            y: 0.5,
        },
    }}
    //Sets the margin of the dragging helper relative to the mouse cursor
    symbolMargin={{
        left: 12,
        right: 12,
        top: 12,
        bottom: 12,
    }} getSymbolInfo=(symbol) => {
        //Defines the symbol description
        return { width: 75, height: 40, description: { text: symbol.id }
    };
    }}/>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    SymbolPalette,
    SymbolInfo,
    SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
    let basicShapes: NodeModel[] = [
        {
            id: "Rectangle",
            shape: {
                type: "Basic",
                shape: "Rectangle",
                symbolInfo: { description: { text: "Rectangle" } },
            },
        },
        {
            id: "Ellipse",
            shape: {

```



```

        type: "Basic",
        shape: "Ellipse",
        symbolInfo: { description: { text: "Ellipse" } } },
    },
},
{
    id: "Hexagon",
    shape: {
        type: "Basic",
        shape: "Hexagon",
        symbolInfo: { description: { text: "Hexagon" } } },
    },
},
];
return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
    let flowShapes: NodeModel[] = [
        {
            id: "Process",
            shape: {
                type: "Flow",
                shape: "Process",
                symbolInfo: { description: { text: "Process" } } },
        },
    ],
    {
        id: "Document",
        shape: {
            type: "Flow",
            shape: "Document",
            symbolInfo: { description: { text: "Document" } } },
        },
    ],
};
return flowShapes;
}
//Initializes the symbol palette
function App() {
    return (
        <SymbolPaletteComponent
            id="container"
            expandMode={'Multiple'}
            palettes=[
                {
                    id: 'basic',
                    expanded: true,
                    symbols: getBasicShapes(),
                    title: 'Basic Shapes',
                    iconCss: 'e-ddb-icons e-basic',
                },
                {
                    id: 'flow',
                    expanded: true,
                    symbols: getFlowShapes(),
                    title: 'Flow Shapes',

```

```

        iconCss: 'e-ddb-icons e-flow',
      },
    ]}
    symbolHeight={80}
    symbolWidth={80}
    //Specifies the preview size and position to symbol palette items.
    symbolPreview={{
      height: 100,
      width: 100,
      offset: {
        x: 0.5,
        y: 0.5,
      },
    },
  }}
  //Sets the margin of the dragging helper relative to the mouse cursor
  symbolMargin={{
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
  }}
  getSymbolInfo={(symbol: NodeModel): SymbolInfo => {
    //Defines the symbol description
    return { width: 75, height: 40, description: { text: symbol.id } };
  }}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

Appearance of symbol description

The appearance of a symbol description in the palette can be customized by changing its **color**, **fontSize**, **fontFamily**, **bold**, **italic**, **textDecoration**, and **margin**.

The following code is an example to change the color of a symbol description for symbols in the palette.

INDEX.JSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
        symbolInfo: { description: { text: "Rectangle" } },
      },
    },
  ],
  {

```

```

        id: "Ellipse",
        shape: {
            type: "Basic",
            shape: "Ellipse",
            symbolInfo: { description: { text: "Ellipse" } } },
    },
    tooltip:{
        content: 'Customized Tooltip',
    },
    //Enable customized tooltip content to display on the symbol
    constraints: NodeConstraints.Default | NodeConstraints.Tooltip
},
{
    id: "Hexagon",
    shape: {
        type: "Basic",
        shape: "Hexagon",
        symbolInfo: { description: { text: "Hexagon" } } },
    },
},
];
return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
    let flowShapes = [
        {
            id: "Process",
            shape: {
                type: "Flow",
                shape: "Process",
            },
        },
        {
            id: "Document",
            shape: {
                type: "Flow",
                shape: "Document",
            },
        },
    ];
    return flowShapes;
}
//Initializes the symbol palette
function App() {
    return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
    palettes=[
        {
            id: 'basic',
            expanded: true,
            symbols: getBasicShapes(),
            title: 'Basic Shapes',
            iconCss: 'e-ddb-icons e-basic',
        },
        {
            id: 'flow',
            expanded: true,

```

```

        symbols: getFlowShapes(),
        title: 'Flow Shapes',
        iconCss: 'e-ddb-icons e-flow',
    },
    ]] symbolHeight={80} symbolWidth={80}
    //Specifies the preview size and position to symbol palette items.
    symbolPreview={{
        height: 100,
        width: 100,
        offset: {
            x: 0.5,
            y: 0.5,
        },
    },
    ]}
    //Sets the margin of the dragging helper relative to the mouse cursor
    symbolMargin={{
        left: 12,
        right: 12,
        top: 12,
        bottom: 12,
    }} getSymbolInfo=(symbol) => {
        //Defines the symbol description
        return {
            width: 75,
            height: 40,
            description: {
                text: symbol.shape["shape"],
                color: "red",
                bold: true,
                fontSize: 15,
                fontFamily: "Arial",
                italic: true,
                textDecoration: "Underline",
                margin: { top: 30, left: 0, right: 0, bottom: 30 },
            },
        },
    };
    }>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
    Diagram,
    NodeModel,
    SymbolPalette,
    SymbolInfo,
    NodeConstraints,
    SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette

```

```

export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
        symbolInfo: { description: { text: "Rectangle" } } },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
        symbolInfo: { description: { text: "Ellipse" } } },
      tooltip: {
        content: 'Customized Tooltip',
      },
      //Enable customized tooltip content to display on the symbol
      constraints: NodeConstraints.Default | NodeConstraints.Tooltip
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
        symbolInfo: { description: { text: "Hexagon" } } },
    },
  ],
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
  let flowShapes: NodeModel[] = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ],
  return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (

```

```

<SymbolPaletteComponent
  id="container"
  expandMode={'Multiple'}
  palettes={[
    {
      id: 'basic',
      expanded: true,
      symbols: getBasicShapes(),
      title: 'Basic Shapes',
      iconCss: 'e-ddb-icons e-basic',
    },
    {
      id: 'flow',
      expanded: true,
      symbols: getFlowShapes(),
      title: 'Flow Shapes',
      iconCss: 'e-ddb-icons e-flow',
    },
  ]}
  symbolHeight={80}
  symbolWidth={80}
  //Specifies the preview size and position to symbol palette items.
  symbolPreview={{
    height: 100,
    width: 100,
    offset: {
      x: 0.5,
      y: 0.5,
    },
  }}
  //Sets the margin of the dragging helper relative to the mouse cursor
  symbolMargin={{
    left: 12,
    right: 12,
    top: 12,
    bottom: 12,
  }}
  getSymbolInfo={(symbol: NodeModel): SymbolInfo => {
    //Defines the symbol description
    return {
      width: 75,
      height: 40,
      description: {
        text: symbol.shape["shape"],
        color: "red",
        bold: true,
        fontSize: 15,
        fontFamily: "Arial",
        italic: true,
        textDecoration: "Underline",
        margin: { top: 30, left: 0, right: 0, bottom: 30 },
      },
    },
  }};
/>
);
}

```

```
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}
```

Tooltip for symbols in symbol palette

The Symbol palette supports displaying tooltips when mouse hovers over the symbols. You can customize the tooltip for each symbol in the symbol palette.

Default tooltip for symbols

When hovering over symbols in the symbol palette, the default tooltip displays the symbol's ID.

Refer to the image below for an illustration of the tooltip behavior in the symbol palette.



Custom tooltip for symbols

To customize the tooltips for symbols in the symbol palette, assign a custom tooltip to the 'Tooltip' content property of each symbol. Once you define the custom tooltip, enable the Tooltip constraints for each symbol, ensuring that the tooltips are displayed when users hover over them.

Here, the code provided below demonstrates how to define tooltip content to symbols within a symbol palette.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { SymbolPaletteComponent, } from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes() {
  let basicShapes = [
    {
      id: "Rectangle",
```

```

        shape: {
          type: "Basic",
          shape: "Rectangle",
          symbolInfo: { description: { text: "Rectangle" } } },
      },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
        symbolInfo: { description: { text: "Ellipse" } } },
      },
      tooltip: {
        content: 'Customized Tooltip',
      },
      //Enable customized tooltip content to display on the symbol
      constraints: NodeConstraints.Default | NodeConstraints.Tooltip
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
        symbolInfo: { description: { text: "Hexagon" } } },
      },
    },
  ];
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes() {
  let flowShapes = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ];
  return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (<SymbolPaletteComponent id="container" expandMode={'Multiple'}
  palettes=[
    {
      id: 'basic',
      expanded: true,

```



```

        symbols: getBasicShapes(),
        title: 'Basic Shapes',
        iconCss: 'e-ddb-icons e-basic',
    },
    {
        id: 'flow',
        expanded: true,
        symbols: getFlowShapes(),
        title: 'Flow Shapes',
        iconCss: 'e-ddb-icons e-flow',
    },
  ]} symbolHeight={80} symbolWidth={80}
  //Specifies the preview size and position to symbol palette items.
  symbolPreview={
    {
      height: 100,
      width: 100,
      offset: {
        x: 0.5,
        y: 0.5,
      },
    },
  }
  //Sets the margin of the dragging helper relative to the mouse cursor
  symbolMargin={
    {
      left: 12,
      right: 12,
      top: 12,
      bottom: 12,
    }
  }
  getSymbolInfo=(symbol) => {
    //Defines the symbol description
    return {
      width: 75,
      height: 40,
      description: {
        text: symbol.shape["shape"],
        color: "red",
        bold: true,
        fontSize: 15,
        fontFamily: "Arial",
        italic: true,
        textDecoration: "Underline",
        margin: { top: 30, left: 0, right: 0, bottom: 30 },
      },
    },
  };
  }>);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,

```

```

NodeModel,
SymbolPalette,
SymbolInfo,
NodeConstraints,
SymbolPaletteComponent,
} from "@syncfusion/ej2-react-diagrams";
//Initialize the basicshapes for the symbol palette
export function getBasicShapes(): NodeModel[] {
  let basicShapes: NodeModel[] = [
    {
      id: "Rectangle",
      shape: {
        type: "Basic",
        shape: "Rectangle",
        symbolInfo: { description: { text: "Rectangle" } } },
    },
    {
      id: "Ellipse",
      shape: {
        type: "Basic",
        shape: "Ellipse",
        symbolInfo: { description: { text: "Ellipse" } } },
      tooltip: {
        content: 'Customized Tooltip',
      },
      //Enable customized tooltip content to display on the symbol
      constraints: NodeConstraints.Default | NodeConstraints.Tooltip
    },
    {
      id: "Hexagon",
      shape: {
        type: "Basic",
        shape: "Hexagon",
        symbolInfo: { description: { text: "Hexagon" } } },
    },
  ];
  return basicShapes;
}
//Initialize the flowshapes for the symbol palette
export function getFlowShapes(): NodeModel[] {
  let flowShapes: NodeModel[] = [
    {
      id: "Process",
      shape: {
        type: "Flow",
        shape: "Process",
      },
    },
    {
      id: "Document",
      shape: {
        type: "Flow",
        shape: "Document",
      },
    },
  ];
}

```

```

    },
  ];
  return flowShapes;
}
//Initializes the symbol palette
function App() {
  return (
    <SymbolPaletteComponent
      id="container"
      expandMode={'Multiple'}
      palettes=[
        {
          id: 'basic',
          expanded: true,
          symbols: getBasicShapes(),
          title: 'Basic Shapes',
          iconCss: 'e-ddb-icons e-basic',
        },
        {
          id: 'flow',
          expanded: true,
          symbols: getFlowShapes(),
          title: 'Flow Shapes',
          iconCss: 'e-ddb-icons e-flow',
        },
      ],
    >
    symbolHeight={80}
    symbolWidth={80}
    //Specifies the preview size and position to symbol palette items.
    symbolPreview={{
      height: 100,
      width: 100,
      offset: {
        x: 0.5,
        y: 0.5,
      },
    }},
    //Sets the margin of the dragging helper relative to the mouse cursor
    symbolMargin={{
      left: 12,
      right: 12,
      top: 12,
      bottom: 12,
    }},
    getSymbolInfo={(symbol: NodeModel): SymbolInfo => {
      //Defines the symbol description
      return {
        width: 75,
        height: 40,
        description: {
          text: symbol.shape["shape"],
          color: "red",
          bold: true,
          fontSize: 15,
          fontFamily: "Arial",
          italic: true,
          textDecoration: "Underline",

```

```

        margin: { top: 30, left: 0, right: 0, bottom: 30 },
      },
    };
  }}
  />
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

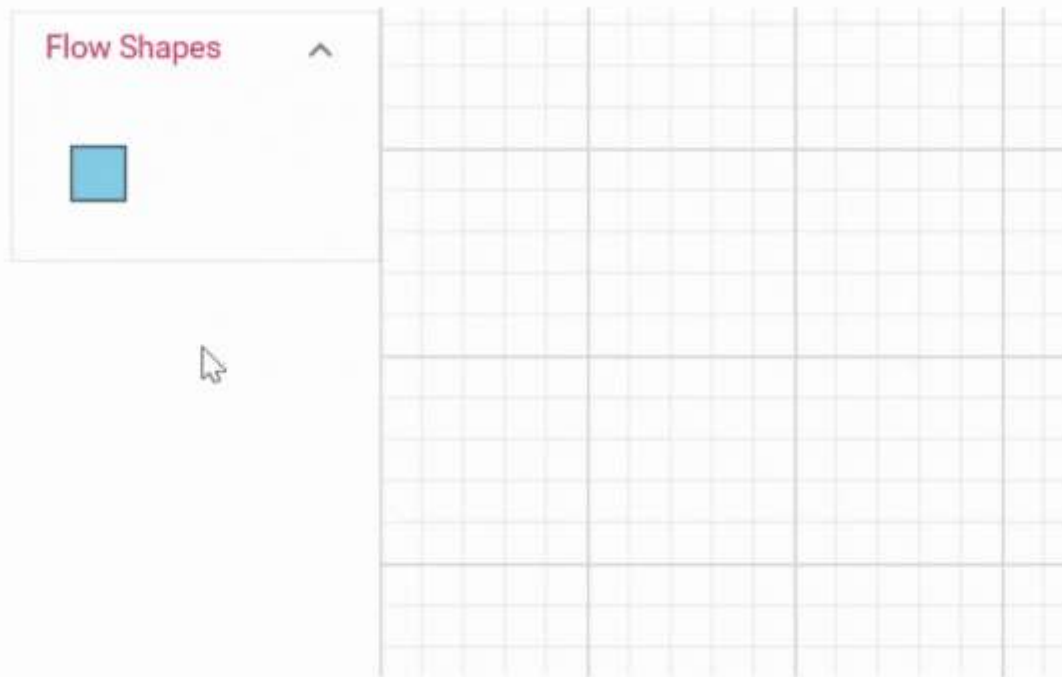
```

How to provide different tooltip for Symbol palette and diagram elements.

Differentiate the tooltips between symbols in the symbol palette and dropped nodes by utilizing the `dragEnter` event. When a custom tooltip is defined for a symbol, it will be displayed for both the symbol and the dropped node in the diagram canvas. However, to provide distinct tooltips for symbols in the palette and dropped nodes, capture the `dragEnter` event and assign specific tooltips dynamically.

When a symbol is dragged from the symbol palette and enters the diagram canvas, the `[DragEnter]` [IDragEnterEventArgs](#) event is triggered. Within this event, you can define a new tooltip for the dropped node. By assigning custom tooltip content to the `Tooltip` property of the node, you can provide a distinct tooltip that is specific to the dropped node.

The following image illustrates the differentiation of tooltips displayed in the Symbol Palette and the Diagram.



The following code snippet will demonstrate how to define two different tooltip for symbol in the symbol palette and dropped node in the diagram canvas.

```

`ts
//Initialize the Diagram

```

```

let diagram: Diagram = new Diagram({
width: '100%', height: '500px',
connectors: connectors, nodes: nodes,
//event to change tooltip content while dragging symbols into Diagram
dragEnter: dragEnter,
});
diagram.appendTo('#diagram');
function dragEnter(args:IDragEnterEventArgs)
{
//enable tooltip constraints for the dragged symbol
args.dragItem.constraints = NodeConstraints.Default | NodeConstraints.Tooltip;
//change the tooltip content of the dragged symbol
args.dragItem.tooltip.content='This is Diagram Tooltip';
}
`

```

Palette interaction

Palette interaction notifies the element enter, leave, and dragging of the symbols into the diagram.

DragEnter

[DragEnter] [IDragEnterEventArgs](#) notifies, when the element enter into the diagram from symbol palette.

DragLeave

[DragLeave] [IDragLeaveEventArgs](#) notifies, when the element leaves from the diagram.

DragOver

[DragOver] [IDragOverEventArgs](#) notifies, when an element drag over another diagram element.

Note: The diagram provides support to cancel the drag and drop operation from the symbol palette to the diagram when the ESC key is pressed.

See Also

- [How to add the symbol to the diagram](#)

Overview in React Diagram component

Overview control allows you to see a preview or an overall view of the entire content of a diagram. This helps you to look at the overall picture of a large diagram and also to navigate, pan, or zoom, on a particular position of the page.

When you work on a very large diagram, you may not know the part you are actually working on, or navigation from one part to another might be difficult. One solution for navigation is to zoom out the

entire diagram and find where you are. Then, you can zoom in a particular area you want to. This solution is not suitable when you need some frequent navigation.

Overview control solves these problems by showing a preview, that is, an overall view of the entire diagram. A rectangle indicates viewport of the diagram. Navigation becomes easy by dragging this rectangle.

Create overview

The `sourceID` property of overview should be set with the corresponding diagram ID for the overall view.

The `width` and `height` properties of the overview allow you to define the size of the overview.

The following code illustrates how to create overview.

Zoom and Pan

In overview, the view port of the diagram is highlighted with a red colored rectangle. Diagram can be zoomed/panned by interacting with that. You can interact with overview as follows:

- Resize the rectangle: Zooms in/out the diagram.
- Drag the rectangle: Pans the diagram.
- Click at a position: Navigates to the clicked region.
- Choose a particular region by clicking and dragging: Navigates to the specified region.

The following image shows how the diagram is zoomed/panned with overview.

INDEX.JSX

```
{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import { DiagramComponent, Inject } from "@syncfusion/ej2-react-diagrams";
import { DataManager, Query } from '@syncfusion/ej2-data';
import { StackPanel, TextElement, DataBinding, OverviewComponent,
HierarchicalTree } from "@syncfusion/ej2-react-diagrams";
/**
 * Overview
 */
let diagram;
let overview;
let data = [{
    'Id': 'parent',
    'Name': 'Maria Anders',
    'Designation': 'Managing Director',
    'IsExpand': 'true',
    'RatingColor': '#C34444'
},
{
    'Id': 1,
    'Name': 'Ana Trujillo',
    'Designation': 'Project Manager',
    'IsExpand': 'false',
    'RatingColor': '#68C2DE',
    'ReportingPerson': 'parent'
},
{

```

```

    'Id': 2,
    'Name': 'Anto Moreno',
    'Designation': 'Project Lead',
    'IsExpand': false,
    'RatingColor': '#93B85A',
    'ReportingPerson': 'parent'
  },
  {
    'Id': 3,
    'Name': 'Thomas Hardy',
    'Designation': 'Senior S/w Engg',
    'IsExpand': false,
    'RatingColor': '#68C2DE',
    'ReportingPerson': 1
  },
  {
    'Id': 4,
    'Name': 'Christina kaff',
    'Designation': 'S/w Engg',
    'IsExpand': false,
    'RatingColor': '#93B85A',
    'ReportingPerson': 2
  },
  {
    'Id': 5,
    'Name': 'Hanna Moos',
    'Designation': 'Project Trainee',
    'IsExpand': true,
    'RatingColor': '#D46E89',
    'ReportingPerson': 2
  }
];

let diagramInstance;
let items = new DataManager(data, new Query().take(7));
// Initializes the Diagram control
function App() {
  return (<
    <div style={{ width: '75%', float: 'left' }}>
      <DiagramComponent id="container" ref={(diagram) => (diagramInstance
= diagram)} height={'850px'} layout={{
        type: 'HierarchicalTree',
        margin: {
          top: 20,
        },
        getLayoutInfo: (node, tree) => {
          if (!tree.hasSubTree) {
            tree.orientation = 'Vertical';
            tree.type = 'Alternate';
          }
        },
      }} dataSourceSettings={{
        id: 'Id',
        parentId: 'ReportingPerson',
        dataManager: items,
      }} getNodeDefaults=(node) => {
        node.height = 50;
        node.style.fill = '#6BA5D7';

```

```
node.borderColor = 'white';
node.style.strokeColor = 'white';
return node;
}} getConnectorDefaults=(obj) => {
  obj.style.strokeColor = '#6BA5D7';
  obj.style.fill = '#6BA5D7';
  obj.style.strokeWidth = 2;
  obj.targetDecorator.style.fill = '#6BA5D7';
  obj.targetDecorator.style.strokeColor = '#6BA5D7';
  obj.targetDecorator.shape = 'None';
  obj.type = 'Orthogonal';
  return obj;
}} setNodeTemplate=(obj, diagram) => {
  let content = new StackPanel();
  content.id = obj.id + '_outerstack';
  content.style.strokeColor = 'darkgreen';
  content.style.fill = '#6BA5D7';
  content.orientation = 'Horizontal';
  content.padding = {
    left: 5,
    right: 10,
    top: 5,
    bottom: 5,
  };
  let innerStack = new StackPanel();
  innerStack.style.strokeColor = 'none';
  innerStack.style.fill = '#6BA5D7';
  innerStack.margin = {
    left: 5,
    right: 0,
    top: 0,
    bottom: 0,
  };
  innerStack.id = obj.id + '_innerstack';
  let text = new TextElement();
  text.content = obj.data['Name'];
  text.style.color = 'white';
  text.style.strokeColor = 'none';
  text.style.fill = 'none';
  text.id = obj.id + '_text1';
  let desigText = new TextElement();
  desigText.margin = {
    left: 0,
    right: 0,
    top: 5,
    bottom: 0,
  };
  desigText.content = obj.data['Designation'];
  desigText.style.color = 'white';
  desigText.style.strokeColor = 'none';
  desigText.style.fill = 'none';
  desigText.style.textWrapping = 'Wrap';
  desigText.id = obj.id + '_desig';
  innerStack.children = [text, desigText];
  content.children = [innerStack];
  return content;
}}>
```



```

        <Inject services={[DataBinding, HierarchicalTree]}/>
      </DiagramComponent>
    </div>
    <OverviewComponent id="overview" style={{
      float: 'Right',
    }} sourceID="container" width={'24%'} height={'130px'}/>
  </>;
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

INDEX.TSX

```

{% raw %}
import * as React from "react";
import * as ReactDOM from "react-dom";
import {
  Diagram,
  DiagramComponent,
  ConnectorModel,
  Overview,
  OverviewModel,
  Inject
} from "@syncfusion/ej2-react-diagrams";
import {
  DataManager,
  Query
} from '@syncfusion/ej2-data';
import {
  TreeInfo,
  Node,
  NodeModel,
  StackPanel,
  Container,
  TextElement,
  DataBinding,
  OverviewComponent,
  HierarchicalTree
} from "@syncfusion/ej2-react-diagrams";
/**
 * Overview
 */
let diagram: DiagramComponent;
let overview: OverviewComponent;
let data: object[] = [{
  'Id': 'parent',
  'Name': 'Maria Anders',
  'Designation': 'Managing Director',
  'IsExpand': 'true',
  'RatingColor': '#C34444'
},
{
  'Id': 1,
  'Name': 'Ana Trujillo',
  'Designation': 'Project Manager',

```

```

        'IsExpand': 'false',
        'RatingColor': '#68C2DE',
        'ReportingPerson': 'parent'
    },
    {
        'Id': 2,
        'Name': 'Anto Moreno',
        'Designation': 'Project Lead',
        'IsExpand': 'false',
        'RatingColor': '#93B85A',
        'ReportingPerson': 'parent'
    },
    {
        'Id': 3,
        'Name': 'Thomas Hardy',
        'Designation': 'Senior S/w Engg',
        'IsExpand': 'false',
        'RatingColor': '#68C2DE',
        'ReportingPerson': 1
    },
    {
        'Id': 4,
        'Name': 'Christina kaff',
        'Designation': 'S/w Engg',
        'IsExpand': 'false',
        'RatingColor': '#93B85A',
        'ReportingPerson': 2
    },
    {
        'Id': 5,
        'Name': 'Hanna Moos',
        'Designation': 'Project Trainee',
        'IsExpand': 'true',
        'RatingColor': '#D46E89',
        'ReportingPerson': 2
    }
];
let diagramInstance: DiagramComponent;
let items: DataManager = new DataManager(data as JSON[], new
Query().take(7));
// Initializes the Diagram control
function App() {
    return (
        <div>
            <div style={{ width: '75%', float: 'left' }}>
                <DiagramComponent
                    id="container"
                    ref={(diagram) => (diagramInstance = diagram)}
                    height={'850px'}
                    layout={{
                        type: 'HierarchicalTree',
                        margin: {
                            top: 20,
                        },
                    },
                    getLayoutInfo: (node, tree) => {
                        if (!tree.hasSubTree) {
                            tree.orientation = 'Vertical';

```

```
        tree.type = 'Alternate';
    }
},
}}
dataSourceSettings={{
    id: 'Id',
    parentId: 'ReportingPerson',
    dataManager: items,
}}
getNodeDefaults=(node) => {
    node.height = 50;
    node.style.fill = '#6BA5D7';
    node.borderColor = 'white';
    node.style.strokeColor = 'white';
    return node;
}
getConnectorDefaults=(obj) => {
    obj.style.strokeColor = '#6BA5D7';
    obj.style.fill = '#6BA5D7';
    obj.style.strokeWidth = 2;
    obj.targetDecorator.style.fill = '#6BA5D7';
    obj.targetDecorator.style.strokeColor = '#6BA5D7';
    obj.targetDecorator.shape = 'None';
    obj.type = 'Orthogonal';
    return obj;
}
setNodeTemplate=(obj, diagram) => {
    let content = new StackPanel();
    content.id = obj.id + '_outerstack';
    content.style.strokeColor = 'darkgreen';
    content.style.fill = '#6BA5D7';
    content.orientation = 'Horizontal';
    content.padding = {
        left: 5,
        right: 10,
        top: 5,
        bottom: 5,
    };
    let innerStack = new StackPanel();
    innerStack.style.strokeColor = 'none';
    innerStack.style.fill = '#6BA5D7';
    innerStack.margin = {
        left: 5,
        right: 0,
        top: 0,
        bottom: 0,
    };
    innerStack.id = obj.id + '_innerstack';
    let text = new TextElement();
    text.content = obj.data['Name'];
    text.style.color = 'white';
    text.style.strokeColor = 'none';
    text.style.fill = 'none';
    text.id = obj.id + '_text1';
    let designText = new TextElement();
    designText.margin = {
        left: 0,
```

```

        right: 0,
        top: 5,
        bottom: 0,
    };
    desigText.content = obj.data['Designation'];
    desigText.style.color = 'white';
    desigText.style.strokeColor = 'none';
    desigText.style.fill = 'none';
    desigText.style.textWrapping = 'Wrap';
    desigText.id = obj.id + '_desig';
    innerStack.children = [text, desigText];
    content.children = [innerStack];
    return content;
  })
  >
  <Inject services={[DataBinding, HierarchicalTree]} />
</DiagramComponent>
</div>
<OverviewComponent
  id="overview"
  style={{
    float: 'Right',
  }}
  sourceID="container"
  width={'24%'}
  height={'130px'}
/>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById('diagram'));
root.render(<App />);
{% enddraw %}

```

How to load EJ1 diagram in EJ2 diagram

To load EJ1 JSON data in an EJ2 diagram, follow these steps.

1. Import and inject the EJ1SerializationModule as shown in the following code example.

```

`ts
import {
  Diagram,
  DiagramComponent,
} from "@syncfusion/ej2-react-diagrams";
Diagram.Inject(EJ1SerializationModule);
//Initializes the symbol palette
function App() {
  return (
    <DiagramComponent

```

```
id="dialog"
width={'100%'}
height={'600px'}
/>
);
}
const root = ReactDOM.createRoot(document.getElementById("dialog"));
root.render(<App />);
`
```

2. Load the EJ1 JSON data using the diagram loadDiagram method and set the second parameter to true.

```
`ts
function diagramCreated(){
var diagram = document.getElementById("dialog").ej2_instances[0];
var ej1Data = {"JSONData"}; //Replace JSONData with your EJ1 JSON data
//Load the EJ1 JSON and pass a boolean value as true.
diagram.loadDiagram(ej1Data, true);
}
```

[Link to the Video`](#)

Summary of Dialog component

Dialog

Getting Started

The following section explains the required steps to build the Dialog component with its basic usage in step by step procedure.

To get start quickly with React Dialog component, you can check on this video:

Dependencies

The following list of dependencies are required to use the React Dialog component in your application.

```
`javascript
|-- @syncfusion/ej2-react-popups
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-react-buttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
```

,

Installation and configuration

You can use [Create-react-app](#) to setup the applications.

To install `create-react-app` run the following command.

```
`bash
```

```
npm install -g create-react-app
```

,

Start a new project using create-react-app command as follows

```
<div class='tsx'>
```

,

```
create-react-app quickstart --scripts-version=react-scripts-ts
```

```
cd quickstart
```

,

```
</div>
```

```
<div class='jsx'>
```

,

```
create-react-app quickstart
```

```
cd quickstart
```

,

```
</div>
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install. For this application, we are going to use Dialog component.

To install Dialog component, use the following command

```
`bash
```

```
npm install @syncfusion/ej2-react-popups --save
```

,

Adding Dialog to the application

Now, you can start adding React Dialog component to the application. We have added Dialog component in `src/App.tsx` file using following code.

```
[Class-component]
```

```
`ts
```

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
```

```
import * as React from "react";
```

```

class App extends React.Component<{}, {hideDialog: boolean}> {
  constructor(props: {}) {
    super(props);
    this.state = {
      hideDialog : false
    };
  }
  public handleClick() {
    this.setState({ hideDialog: true })
  }
  public dialogClose = () => {
    this.setState({ hideDialog: false })
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button' onClick={this.handleClick =
this.handleClick.bind(this)} >Open</button>
        <DialogComponent width='250px' content='This is a Dialog with content' target='.App' visible =
{this.state.hideDialog} close = {this.dialogClose}/>
      </div>);
    }
  }
  export default App;
}
`ts
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      hideDialog: false
    };

```

```

}
handleClick() {
this.setState({ hideDialog: true });
}
dialogClose = () => {
this.setState({ hideDialog: false });
};
render() {
return (<div className="App" id='dialog-target'>
<button className='e-control e-btn' id='targetButton1' role='button' onClick={this.handleClick =
this.handleClick.bind(this)}>Open</button>
<DialogComponent width='250px' content='This is a Dialog with content' target='.App'
visible={this.state.hideDialog} close={this.dialogClose}/>
</div>);
}
}
export default App;
`

```

[Functional-component]

```

`ts
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
const [status, setStatus] = React.useState({ hideDialog: false });
function handleClick() {
setStatus({ hideDialog: true })
}
function dialogClose() {
setStatus({ hideDialog: false })
}
return (
<div className="App" id='dialog-target'>
<button className='e-control e-btn' id='targetButton1' role='button' onClick={handleClick.bind(this)}
>Open</button>

```



```

<DialogComponent width='250px' content='This is a Dialog with content' target='.App' visible =
{status.hideDialog} close = {dialogClose}/>
</div>
);
}
export default App;
`ts
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  const [status, setStatus] = React.useState({ hideDialog: false });
  function handleClick() {
    setStatus({ hideDialog: true });
  }
  function dialogClose() {
    setStatus({ hideDialog: false });
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1' role='button'
    onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' content='This is a Dialog with content' target='.App'
    visible={status.hideDialog} close={dialogClose}/>
  </div>);
}
export default App;
`

```

Adding CSS reference

Import the Dialog component's required CSS references as follows in `src/App.css`.

```

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

Run the application

Now use the `npm run start` command to run the application in the browser.

,

`npm run start`

,

The below example shows the Dialog.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  constructor(props) {
    super(props);
    this.state = {
      hideDialog: false
    };
  }
  handleClick() {
    this.setState({ hideDialog: true });
  }
  dialogClose = () => {
    this.setState({ hideDialog: false });
  };
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' content='This is a Dialog with content'
        target='#dialog-target' visible={this.state.hideDialog}
        close={this.dialogClose}/>
    </div>);
  }
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean}> {
  public dialogInstance: DialogComponent;
  constructor(props: {}) {
    super(props);
    this.state = {
      hideDialog : false
    };
  }
}
```

```

    };
  }
  public handleClick() {
    this.setState({ hideDialog: true })
  }

  public dialogClose = () => {
    this.setState({ hideDialog: false })
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)} >Open</button>
        <DialogComponent width='250px' content='This is a Dialog with content'
target='#dialog-target' visible = {this.state.hideDialog} close =
{this.dialogClose}/>
      </div>);
    )
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  const [status, setStatus] = React.useState({ hideDialog: false });
  function handleClick() {
    setStatus({ hideDialog: true });
  }
  function dialogClose() {
    setStatus({ hideDialog: false });
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' content='This is a Dialog with
content' target='#dialog-target' visible={status.hideDialog}
close={dialogClose}/>
  </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  const [status, setStatus] = React.useState({ hideDialog: false });
  function handleClick() {
    setStatus({ hideDialog: true })
  }

```

```

    }
    function dialogClose () {
        setStatus({ hideDialog: false })
    }
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)} >Open</button>
            <DialogComponent width='250px' content='This is a Dialog with
content' target='#dialog-target' visible = {status.hideDialog} close =
{dialogClose}/>
        </div>
    );
}
export default App;

```

In the dialog control, max-height is calculated based on the dialog target element height. If the target property is not configured, the document.body is considered as a target. Therefore, to show a dialog in proper height, you need to add min-height to the target element.

If the dialog is rendered based on the body, then the dialog will get the height based on its body element height. If the height of the dialog is larger than the body height, then the dialog's height will not be set. For this scenario, we can set the CSS style for the html and body to get the dialog height.

```
`css
```

```
html, body {
height: 100%;
}
`
```

Modal Dialog

A [modal](#) shows an overlay behind the Dialog. So, the user should interact the Dialog compulsory before interacting with the remaining content in an application.

While the user clicks the overlay, the action can be handled through the [overlayClick](#) event. In the below sample, the Dialog close action is performed while clicking on the overlay.

When the modal dialog is opened, the Dialog's target scrolling will be disabled. The scrolling will be enabled again once close the Dialog.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    dialogInstance;
    constructor(props) {
        super(props);
        this.state = {
            hideDialog: true
        };
    }

```

```

    }
    onOverlayClick = () => {
        this.setState({ hideDialog: false });
    };
    dialogClose = () => {
        this.setState({ hideDialog: false });
    };
    handleClick() {
        this.setState({ hideDialog: true });
    }
    render() {
        return (<div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
                role='button' onClick={this.handleClick =
                this.handleClick.bind(this)}>Open</button>
            <DialogComponent width='250px' isModal={true} target='#dialog-
                target' visible={this.state.hideDialog} close={this.dialogClose}
                overlayClick={this.onOverlayClick}>
                This is a modal Dialog </DialogComponent>
            </div>);
    }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean}> {
    public dialogInstance: DialogComponent;
    constructor(props: {}) {
        super(props);
        this.state = {
            hideDialog : true
        };
    }
    public onOverlayClick = () => {
        this.setState({ hideDialog: false })
    }
    public dialogClose = () => {
        this.setState({ hideDialog: false })
    }
    public handleClick() {
        this.setState({ hideDialog: true })
    }
    public render() {
        return (
            <div className="App" id='dialog-target'>
                <button className='e-control e-btn' id='targetButton1'
                    role='button' onClick={this.handleClick = this.handleClick.bind(this)}
                >Open</button>
                <DialogComponent width='250px' isModal={true} target='#dialog-
                    target' visible = {this.state.hideDialog} close = {this.dialogClose}
                    overlayClick={ this.onOverlayClick } >
                    This is a modal Dialog </DialogComponent>
                </div>);
    }
}

```

```

    }
  }
  export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  const [status, setStatus] = React.useState({ hideDialog: true });
  function onOverlayClick() {
    setStatus({ hideDialog: false });
  }
  function dialogClose() {
    setStatus({ hideDialog: false });
  }
  function handleClick() {
    setStatus({ hideDialog: true });
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' isModal={true} target='#dialog-
        target' visible={status.hideDialog} close={dialogClose}
        overlayClick={onOverlayClick}>
        This is a modal Dialog </DialogComponent>
      </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  const [status, setStatus] = React.useState({ hideDialog: true });
  function onOverlayClick() {
    setStatus({ hideDialog: false })
  }
  function dialogClose() {
    setStatus({ hideDialog: false })
  }
  function handleClick() {
    setStatus({ hideDialog: true })
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={handleClick.bind(this)} >Open</button>
      <DialogComponent width='250px' isModal={true} target='#dialog-
        target' visible = {status.hideDialog} close = {dialogClose}
        overlayClick={ onOverlayClick } >

```

```

        This is a modal Dialog </DialogComponent>
      </div>
    );
  }
  export default App;

```

Enable header

The Dialog header can be enabled by adding the header content as text or HTML content through the [header](#) property.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  render() {
    return (<div className="App" id='dialog-target'>
      <DialogComponent width='250px' target='#dialog-target'
showCloseIcon={true} header='Dialog' closeOnEscape={false}>
        This is a dialog with header </DialogComponent>
      </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <DialogComponent width='250px' target='#dialog-target'
showCloseIcon={true} header='Dialog' closeOnEscape = {false}>
          This is a dialog with header </DialogComponent>
        </div>);
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  return (<div className="App" id='dialog-target'>
    <DialogComponent width='250px' target='#dialog-target'
showCloseIcon={true} header='Dialog' closeOnEscape={false}>
      This is a dialog with header </DialogComponent>
    </div>);
}

```

```
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  return (
    <div className="App" id='dialog-target'>
      <DialogComponent width='250px' target='#dialog-target'
showCloseIcon={true} header='Dialog' closeOnEscape = {false}>
        This is a dialog with header </DialogComponent>
      </div>
    );
}
export default App;
```

Enable footer

The React Dialog provides built-in support to render the **buttons** on the footer (for ex: 'OK' or 'Cancel' buttons). Each Dialog button allows the user to perform any action while clicking on it.

The primary button will be focused automatically on open the Dialog, and add the [click](#) event to handle the actions.

When the Dialog initialize with more than one primary buttons, the first primary button gets focus on open the Dialog.

The below sample render with button and its click event.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  }
  ]};
```



```

    constructor(props) {
      super(props);
      this.state = {
        hideDialog: true
      };
    }
    handleClick() {
      this.setState({ hideDialog: true });
    }
    dialogClose = () => {
      this.setState({ hideDialog: false });
    };
    render() {
      return (
        <div className="App" id='dialog-target'>
          <button className='e-control e-btn' id='targetButton1' role='button'
            onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
          <DialogComponent width='250px' target='#dialog-target'
            close={this.dialogClose} header='Dialog' visible={this.state.hideDialog}
            showCloseIcon={true} buttons={this.buttons}>
            This is a Dialog with button and primary button </DialogComponent>
          </div>);
    }
  }
  export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean}> {
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  }
  ]];
  constructor(props: {}) {
    super(props);
    this.state = {
      hideDialog : true
    };
  }
}

```

```

public handleClick() {
    this.setState({ hideDialog: true })
}

public dialogClose = () => {
    this.setState({ hideDialog: false })
}

public render() {
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
            <DialogComponent width='250px' target='#dialog-target' close =
{this.dialogClose} header='Dialog' visible =
{this.state.hideDialog} showCloseIcon={true} buttons={this.buttons}>
                This is a Dialog with button and primary button </DialogComponent>
            </div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    const [status, setStatus] = React.useState({ hideDialog: true });
    let buttons = [{
        buttonModel: {
            content: 'OK',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            setStatus({ hideDialog: false });
        }
    },
    {
        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            setStatus({ hideDialog: false });
        }
    }
    ]];
    function handleClick() {
        setStatus({ hideDialog: true });
    }
    function dialogClose() {
        setStatus({ hideDialog: false });
    }
    return (<div className="App" id='dialog-target'>

```

```

    <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' target='#dialog-target'
close={dialogClose} header='Dialog' visible={status.hideDialog}
showCloseIcon={true} buttons={buttons}>
    This is a Dialog with button and primary button </DialogComponent>
</div>;
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App(){
    const [status, setStatus] = React.useState({ hideDialog: true });
    let buttons: any = [{
        buttonModel: {
            content: 'OK',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            setStatus({ hideDialog: false })
        }
    },
    {
        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            setStatus({ hideDialog: false })
        }
    }
    ]];
    function handleClick() {
        setStatus({ hideDialog: true })
    }
    function dialogClose() {
        setStatus({ hideDialog: false })
    }
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
            <DialogComponent width='250px' target='#dialog-target' close =
{dialogClose} header='Dialog' visible = {status.hideDialog}
showCloseIcon={true} buttons={buttons}>
                This is a Dialog with button and primary button </DialogComponent>
            </div>
        );
    }
}
export default App;

```

Draggable

The Dialog supports to [drag](#) within its target container by grabbing the Dialog header, which allows the user to reposition the Dialog dynamically.

The Dialog can be draggable only when the Dialog header is enabled. From **16.2.x** version, enabled draggable support for modal dialog also.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  }
  ];
  constructor(props) {
    super(props);
    this.state = {
      hideDialog: true
    };
  }
  handleClick() {
    this.setState({ hideDialog: true });
  }
  dialogClose = () => {
    this.setState({ hideDialog: false });
  };
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' target='#dialog-target'
        visible={this.state.hideDialog} close={this.dialogClose} header='Dialog'
        allowDragging={true} showCloseIcon={true} buttons={this.buttons}>
        This is a Dialog with drag enabled </DialogComponent>
      </div>);
  }
}
export default App;
```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean}> {
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  }
  ]];
  constructor(props: {}) {
    super(props);
    this.state = {
      hideDialog : true
    };
  }
  public handleClick() {
    this.setState({ hideDialog: true })
  }
  public dialogClose = () => {
    this.setState({ hideDialog: false })
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent width='250px' target='#dialog-target' visible =
{this.state.hideDialog} close = {this.dialogClose} header='Dialog'
allowDragging={true} showCloseIcon={true} buttons={this.buttons}>
          This is a Dialog with drag enabled </DialogComponent>
        </div>);
  }
}
export default App;

```

[Functional-component]**APP.JSX**

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';

```

```

import * as React from "react";
function App() {
  const [status, setStatus] = React.useState({ hideDialog: true });
  let buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      setStatus({ hideDialog: false });
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      setStatus({ hideDialog: false });
    }
  }
  ];
  function handleClick() {
    setStatus({ hideDialog: true });
  }
  function dialogClose() {
    setStatus({ hideDialog: false });
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
    role='button' onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' target='#dialog-target'
    visible={status.hideDialog} close={dialogClose} header='Dialog'
    allowDragging={true} showCloseIcon={true} buttons={buttons}>
      This is a Dialog with drag enabled </DialogComponent>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  const [status, setStatus] = React.useState({ hideDialog: true });
  let buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      setStatus({ hideDialog: false });
    }
  },
  {
  }
  ];
}

```

```

    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      setStatus({ hideDialog: false })
    }
  ]];
  function handleClick() {
    setStatus({ hideDialog: true })
  }
  function dialogClose() {
    setStatus({ hideDialog: false })
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' target='#dialog-target' visible =
        {status.hideDialog} close = {dialogClose} header='Dialog'
        allowDragging={true} showCloseIcon={true} buttons={buttons}>
        This is a Dialog with drag enabled </DialogComponent>
    </div>
  );
}
export default App;

```

Positioning

The Dialog position can be set through the [position](#) property by providing X and Y coordinates. The Dialog can be positioned inside the target container based on the given X and Y values.

For example `<!-- markdownlint-disable MD033 --> <code>position:{ X:'center', Y:'center' }</code>` the possible values

- for X is: left, center, right (or) any offset value
- for Y is: top, center, bottom (or) any offset value

The below sample demonstrates the different Dialog positions.

[Class-component]

APP.JSX

```

import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  defaultDialogInstance;
  PositioningInstance;
  position;
  constructor(props) {
    super(props);
    this.state = {
      hideDialog: true
    };
  }
}

```

```

        this.position = { X: 'center', Y: 'center' };
    }
    changePosition = (event) => {
        this.defaultDialogInstance.position = { X:
event.currentTarget.value.split(" ")[0], Y:
event.currentTarget.value.split(" ")[1] };
        this.PositioningInstance.innerHTML = 'Position: {X: " ' +
event.currentTarget.value.split(" ")[0] + '" , Y: " ' +
event.currentTarget.value.split(" ")[1] + '"}';
        const txt = event.target.parentElement.querySelector('.e-
label').innerHTML.split(" ");
        this.PositioningInstance.innerHTML = 'Position: { X: " ' + txt[0] +
'" , Y: " ' + txt[1] + '" }';
    };
    dialogClose = () => {
        this.setState({ hideDialog: false });
    };
    dialogOpen = () => {
        this.setState({ hideDialog: true });
    };
    footerTemplate = () => {
        return (<span id="posvalue"/>);
    };
    render() {
        return (<div className="App" id='dialog-target'>
            <DialogComponent id='defaultDialog' header='Choose a Dialog Position'
visible={this.state.hideDialog} showCloseIcon={false}
position={this.position} footerTemplate={this.footerTemplate} width='452px'
ref={defaultDialog => this.defaultDialogInstance = defaultDialog}
target='#dialog-target' open={this.dialogOpen} close={this.dialogClose}
closeOnEscape={false}>
                <table id='poschange'>
                    <tbody>
                        <tr>
                            <td><RadioButtonComponent id='radio1' label='Left Top'
value='left top' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
                            <td><RadioButtonComponent id='radio2' label='Center Top'
value='center top' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
                            <td><RadioButtonComponent id='radio3' label='Right Top'
value='right top' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
                        </tr>
                        <tr>
                            <td><RadioButtonComponent id='radio4' label='Left Center'
value='left center' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
                            <td><RadioButtonComponent id='radio5' checked={true}
label='Center Center' value='center center' name='xy'
onClick={this.changePosition = this.changePosition.bind(this)} /></td>
                            <td><RadioButtonComponent id='radio6' label='Right Center'
value='right center' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
                        </tr>
                    </tbody>
                </table>
            </div>
        );
    }
}

```



```

        <td><RadioButtonComponent id='radio7' label='Left Bottom'
value='left bottom' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio8' label='Center Bottom'
value='center bottom' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio9' label='Right Bottom'
value='right bottom' name='xy' onClick={this.changePosition =
this.changePosition.bind(this)} /></td>
    </tr>
</tbody>
</table>
</DialogComponent>
</div>);
    }
}
export default App;

```

APP.TSX

```

import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean;}> {
    public defaultDialogInstance: DialogComponent;
    public PositioningInstance: HTMLElement;
    public position: any;
    constructor(props: {}) {
        super(props);
        this.state = {
            hideDialog : true
        };
        this.position = { X: 'center', Y: 'center' };
    }
    public changePosition = (event:any): void => {
        this.defaultDialogInstance.position = { X:
event.currentTarget.value.split(" ")[0], Y:
event.currentTarget.value.split(" ")[1] };
        this.PositioningInstance.innerHTML = 'Position: {X: " " +
event.currentTarget.value.split(" ")[0] + " ", Y: " " +
event.currentTarget.value.split(" ")[1] + " "}'
        const txt: string[] = event.target.parentElement.querySelector('.e-
label').innerText.split(" ");
        this.PositioningInstance.innerHTML = 'Position: { X: " " + txt[0] + " ",
Y: " " + txt[1] + " " }';
    }
    public dialogClose = () => {
        this.setState({ hideDialog: false });
    }
    public dialogOpen = () => {
        this.setState({ hideDialog: true });
    }
    public footerTemplate = (): JSX.Element => {
        return (
            <span id="posvalue" />
        )
    }
}

```

```

    }
    public render() {
        return (
            <div className="App" id='dialog-target'>
                <DialogComponent id='defaultDialog' header='Choose a Dialog Position'
                    visible={this.state.hideDialog} showCloseIcon={false}
                    position={this.position} footerTemplate={this.footerTemplate} width='452px'
                    ref={defaultDialog => this.defaultDialogInstance = defaultDialog!}
                    target='#dialog-target' open={this.dialogOpen}
                    close={this.dialogClose} closeOnEscape={false}>
                    <table id='poschange'>
                        <tbody>
                            <tr>
                                <td><RadioButtonComponent id='radio1' label='Left Top'
                                    value='left top' name='xy' onClick = {this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio2' label='Center Top'
                                    value='center top' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio3' label='Right Top'
                                    value='right top' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                            </tr>
                            <tr>
                                <td><RadioButtonComponent id='radio4' label='Left Center'
                                    value='left center' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio5' checked = {true}
                                    label='Center Center' value='center center' name='xy'
                                    onClick={this.changePosition = this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio6' label='Right Center'
                                    value='right center' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                            </tr>
                            <tr>
                                <td><RadioButtonComponent id='radio7' label='Left Bottom'
                                    value='left bottom' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio8' label='Center Bottom'
                                    value='center bottom' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                                <td><RadioButtonComponent id='radio9' label='Right Bottom'
                                    value='right bottom' name='xy' onClick={this.changePosition =
                                    this.changePosition.bind(this)} /></td>
                            </tr>
                        </tbody>
                    </table>
                </DialogComponent>
            </div>
        );
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let defaultDialogInstance;
    let PositioningInstance;
    const [status, setStatus] = React.useState({ hideDialog: true });
    let position;
    position = { X: 'center', Y: 'center' };
    function changePosition(event) {
        defaultDialogInstance.position = { X:
event.currentTarget.value.split(" ")[0], Y:
event.currentTarget.value.split(" ")[1] };
        PositioningInstance.innerHTML = 'Position: {X: " ' +
event.currentTarget.value.split(" ")[0] + '" , Y: " ' +
event.currentTarget.value.split(" ")[1] + '"}';
        const txt = event.target.parentElement.querySelector('.e-
label').innerText.split(" ");
        PositioningInstance.innerHTML = 'Position: { X: " ' + txt[0] + '" , Y:
" ' + txt[1] + '" }';
    }
    function dialogClose() {
        setStatus({ hideDialog: false });
    }
    function dialogOpen() {
        setStatus({ hideDialog: true });
    }
    function footerTemplate() {
        return (<span id="posvalue"/>);
    }
    return (<div className="App" id='dialog-target'>
        <DialogComponent id='defaultDialog' header='Choose a Dialog
Position' visible={status.hideDialog} showCloseIcon={false}
position={position} footerTemplate={footerTemplate} width='452px'
ref={defaultDialog => defaultDialogInstance = defaultDialog}
target='#dialog-target' open={dialogOpen} close={dialogClose}
closeOnEscape={false}>
            <table id='poschange'>
                <tbody>
                    <tr>
                        <td><RadioButtonComponent id='radio1' label='Left Top'
value='left top' name='xy' onClick={changePosition.bind(this)}></td>
                        <td><RadioButtonComponent id='radio2' label='Center Top'
value='center top' name='xy' onClick={changePosition.bind(this)}></td>
                        <td><RadioButtonComponent id='radio3' label='Right Top'
value='right top' name='xy' onClick={changePosition.bind(this)}></td>
                    </tr>
                    <tr>
                        <td><RadioButtonComponent id='radio4' label='Left Center'
value='left center' name='xy' onClick={changePosition.bind(this)}></td>
                        <td><RadioButtonComponent id='radio5' checked={true}
label='Center Center' value='center center' name='xy'
onClick={changePosition.bind(this)}></td>
                        <td><RadioButtonComponent id='radio6' label='Right Center'
value='right center' name='xy' onClick={changePosition.bind(this)}></td>

```

```

        </tr>
        <tr>
        <td><RadioButtonComponent id='radio7' label='Left Bottom'
value='left bottom' name='xy' onClick={changePosition.bind(this)}></td>
        <td><RadioButtonComponent id='radio8' label='Center Bottom'
value='center bottom' name='xy' onClick={changePosition.bind(this)}></td>
        <td><RadioButtonComponent id='radio9' label='Right Bottom'
value='right bottom' name='xy' onClick={changePosition.bind(this)}></td>
        </tr>
    </tbody>
</table>
</DialogComponent>
</div>);
}
export default App;

```

APP.TSX

```

import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent, PositionDataModel } from '@syncfusion/ej2-react-
popups';
import * as React from "react";
function App() {
    let defaultDialogInstance: DialogComponent;
    let PositioningInstance: HTMLElement;
    const [status, setStatus] = React.useState({ hideDialog: true });
    let position: PositionDataModel;
    position = { X: 'center', Y: 'center' };
    function changePosition(event: any): void {
        defaultDialogInstance.position = { X: event.currentTarget.value.split("
")[0], Y: event.currentTarget.value.split(" ")[1] };
        PositioningInstance.innerHTML = 'Position: {X: "' +
event.currentTarget.value.split(" ")[0] + '", Y: "' +
event.currentTarget.value.split(" ")[1] + '"}';
        const txt: string[] = event.target.parentElement.querySelector('.e-
label').innerText.split(" ");
        PositioningInstance.innerHTML = 'Position: { X: "' + txt[0] + '", Y: "'
+ txt[1] + '" }';
    }
    function dialogClose () {
        setStatus({ hideDialog: false });
    }
    function dialogOpen() {
        setStatus({ hideDialog: true });
    }
    function footerTemplate (): JSX.Element {
        return (
            <span id="posvalue" />
        )
    }
    return (
        <div className="App" id='dialog-target'>
            <DialogComponent id='defaultDialog' header='Choose a Dialog
Position' visible={status.hideDialog} showCloseIcon={false}
position={position} footerTemplate={footerTemplate} width='452px'
ref={defaultDialog => defaultDialogInstance = defaultDialog!}>

```

```

        target='#dialog-target' open={dialogOpen} close={dialogClose}
closeOnEscape={false}>
    <table id='poschange'>
    <tbody>
    <tr>
        <td><RadioButtonComponent id='radio1' label='Left Top'
value='left top' name='xy' onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio2' label='Center Top'
value='center top' name='xy' onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio3' label='Right Top'
value='right top' name='xy' onClick={changePosition.bind(this)} /></td>
    </tr>
    <tr>
        <td><RadioButtonComponent id='radio4' label='Left Center'
value='left center' name='xy' onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio5' checked={true}
label='Center Center' value='center center' name='xy'
onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio6' label='Right Center'
value='right center' name='xy' onClick={changePosition.bind(this)} /></td>
    </tr>
    <tr>
        <td><RadioButtonComponent id='radio7' label='Left Bottom'
value='left bottom' name='xy' onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio8' label='Center Bottom'
value='center bottom' name='xy' onClick={changePosition.bind(this)} /></td>
        <td><RadioButtonComponent id='radio9' label='Right Bottom'
value='right bottom' name='xy' onClick={changePosition.bind(this)} /></td>
    </tr>
    </tbody>
    </table>
    </DialogComponent>
</div>
);
}
export default App;

```

See Also

- [Load dialog content using AJAX](#)
- [How to position the dialog on center of the page on scrolling](#)
- [Prevent closing of modal dialog](#)
- [Close dialog while click on outside of dialog](#)
- [How to make a reusable alert and confirm dialog](#)

Getting Started

The following section explains the required steps to build the Dialog component with its basic usage in step by step procedure.

Dependencies

The following list of dependencies are required to use the Dialog component in your application.

`javascript`

```
|-- @syncfusion/ej2-react-popups
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-react-buttons
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-buttons
,`
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
,`
```

Start a new project using create-react-app command as follows

```
<div class='tsx'>
,`
```

```
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
,`
```

```
</div>
<div class='jsx'>
,`
```

```
create-react-app quickstart
cd quickstart
,`
```

```
</div>
```

Adding Syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install. For this application, we are going to use Dialog component.

To install Dialog component, use the following command

```
`bash
npm install @syncfusion/ej2-react-popups --save
,`
```

Adding Dialog to the application

Now, you can start adding Dialog component to the application. We have added Dialog component in `src/App.tsx` file using following code.

```
`ts
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  function dialogClose() {
    setDialogVisibility(false);
  }
  function handleClick() {
    setDialogVisibility(true);
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button' onClick={handleClick}>Open</button>
      <DialogComponent width='250px' content='This is a Dialog with content' target='.App' visible =
        {visibility} close = {dialogClose}/>
    </div>
  );
}
export default App;
`ts
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  function dialogClose() {
    setDialogVisibility(false);
```

```

}
function handleClick() {
  setDialogVisibility(true);
}
return (<div className="App" id='dialog-target'>
  <button className='e-control e-btn' id='targetButton1' role='button'
    onClick={handleClick}>Open</button>
  <DialogComponent width='250px' content='This is a Dialog with content' target='.App' visible={visibility}
    close={dialogClose}/>
</div>);
}
export default App;
`

```

Adding CSS reference

Import the Dialog component's required CSS references as follows in `src/App.css`.

```

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

Run the application

Now use the `npm run start` command to run the application in the browser.

```

npm run start
`

```

The below example shows the Dialog.

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  function dialogClose() {
    setDialogVisibility(false);
  }

```



```

    }
    function handleClick() {
        setDialogVisibility(true);
    }
    return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick}>Open</button>
        <DialogComponent width='250px' content='This is a Dialog with
content' target='#dialog-target' visible={visibility} close={dialogClose}/>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
    function dialogClose() {
        setDialogVisibility(false);
    }
    function handleClick() {
        setDialogVisibility(true);
    }
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick} >Open</button>
            <DialogComponent width='250px' content='This is a Dialog with
content' target='#dialog-target' visible = {visibility} close =
{dialogClose}/>
        </div>
    );
}
export default App;

```

In the dialog control, max-height is calculated based on the dialog target element height. If the target property is not configured, the document.body is considered as a target. Therefore, to show a dialog in proper height, you need to add min-height to the target element.

If the dialog is rendered based on the body, then the dialog will get the height based on its body element height. If the height of the dialog is larger than the body height, then the dialog's height will not be set. For this scenario, we can set the CSS style for the html and body to get the dialog height.

`css

```

html, body {
height: 100%;
}
`

```

Modal Dialog

A [modal](#) shows an overlay behind the Dialog. So, the user should interact the Dialog compulsory before interacting with the remaining content in an application.

While the user clicks the overlay, the action can be handled through the [overlayClick](#) event. In the below sample, the Dialog close action is performed while clicking on the overlay.

When the modal dialog is opened, the Dialog's target scrolling will be disabled. The scrolling will be enabled again once close the Dialog.

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
    function onOverlayClick() {
        setDialogVisibility(false);
    }
    function dialogClose() {
        setDialogVisibility(false);
    }
    function handleClick() {
        setDialogVisibility(true);
    }
    return (<div className="App" id="dialog-target">
        <button className="e-control e-btn" id="targetButton1" role="button"
onClick={handleClick}> Open</button>
        <DialogComponent width="250px" isModal={true} target="#dialog-target"
visible={visibility} close={dialogClose} overlayClick={onOverlayClick}>This
is a modal Dialog{' '}</DialogComponent>
        </div>);
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
    function onOverlayClick() {
        setDialogVisibility(false);
    }
    function dialogClose() {
        setDialogVisibility(false);
    }
    function handleClick() {
        setDialogVisibility(true);
    }
    return (
        <div className="App" id="dialog-target">
            <button
                className="e-control e-btn"
                id="targetButton1"
                role="button"
                onClick={handleClick}> Open</button>
            <DialogComponent width="250px" isModal={true} target="#dialog-target"
                visible={visibility} close={dialogClose} overlayClick={onOverlayClick}>This
                is a modal Dialog{' '}</DialogComponent>
        </div>
    );
}
```

```

        id="targetButton1"
        role="button"
        onClick={handleClick}> Open</button>
      <DialogComponent
        width="250px"
        isModal={true}
        target="#dialog-target"
        visible={visibility}
        close={dialogClose}
        overlayClick={onOverlayClick}>This is a modal Dialog{ '
      '}</DialogComponent>
    </div>
  );
}
export default App;

```

Enable header

The Dialog header can be enabled by adding the header content as text or HTML content through the [header](#) property.

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  return (
    <div className="App" id='dialog-target'>
      <DialogComponent width='250px' target='#dialog-target'
        showCloseIcon={true} header='Dialog' closeOnEscape={false}>
        This is a dialog with header </DialogComponent>
      </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  return (
    <div className="App" id='dialog-target'>
      <DialogComponent width='250px' target='#dialog-target'
        showCloseIcon={true} header='Dialog' closeOnEscape = {false}>
        This is a dialog with header </DialogComponent>
      </div>
    );
}
export default App;

```

Enable footer

The Dialog provides built-in support to render the **buttons** on the footer (for ex: 'OK' or 'Cancel' buttons). Each Dialog button allows the user to perform any action while clicking on it.

The primary button will be focused automatically on open the Dialog, and add the [click](#) event to handle the actions

When the Dialog initialize with more than one primary buttons, the first primary button gets focus on open the Dialog.

The below sample render with button and its click event.

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
    const buttons = [
        {
            buttonModel: {
                content: 'OK',
                cssClass: 'e-flat',
                isPrimary: true,
            },
            click: () => {
                setDialogVisibility(false);
            },
        },
        {
            buttonModel: {
                content: 'Cancel',
                cssClass: 'e-flat',
            },
            click: () => {
                setDialogVisibility(false);
            },
        },
    ];
    function handleClick() {
        setDialogVisibility(true);
    }
    function dialogClose() {
        setDialogVisibility(false);
    }
    return (<div className="App" id="dialog-target">
        <button className="e-control e-btn" id="targetButton1" role="button"
onClick={handleClick}>Open</button>
        <DialogComponent width="250px" target="#dialog-target"
close={dialogClose} header="Dialog" visible={visibility}
showCloseIcon={true} buttons={buttons}>This is a Dialog with button and
primary button{ ' '}</DialogComponent>
    </div>);
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
```

```

const buttons: object = [
  {
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    click: () => {
      setDialogVisibility(false);
    },
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat',
    },
    click: () => {
      setDialogVisibility(false);
    },
  },
];
function handleClick() {
  setDialogVisibility(true);
}
function dialogClose() {
  setDialogVisibility(false);
}
return (
  <div className="App" id="dialog-target">
    <button
      className="e-control e-btn"
      id="targetButton1"
      role="button"
      onClick={handleClick}>Open</button>
    <DialogComponent
      width="250px"
      target="#dialog-target"
      close={dialogClose}
      header="Dialog"
      visible={visibility}
      showCloseIcon={true}
      buttons={buttons}>This is a Dialog with button and primary button{ '
    }</DialogComponent>
  </div>
);
}
export default App;

```

Draggable

The Dialog supports to [drag](#) within its target container by grabbing the Dialog header, which allows the user to reposition the Dialog dynamically.

The Dialog can be draggable only when the Dialog header is enabled. From 16.2.x version, enabled draggable support for modal dialog also.

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  const buttons = [
    {
      buttonModel: {
        content: 'OK',
        cssClass: 'e-flat',
        isPrimary: true,
      },
      click: () => {
        setDialogVisibility(false);
      },
    },
    {
      buttonModel: {
        content: 'Cancel',
        cssClass: 'e-flat',
      },
      click: () => {
        setDialogVisibility(false);
      },
    },
  ];
  function handleClick() {
    setDialogVisibility(true);
  }
  function dialogClose() {
    setDialogVisibility(false);
  }
  return (<div className="App" id="dialog-target">
    <button className="e-control e-btn" id="targetButton1" role="button"
    onClick={handleClick}>Open</button>
    <DialogComponent width="250px" target="#dialog-target"
    visible={visibility} close={dialogClose} header="Dialog"
    allowDragging={true} showCloseIcon={true} buttons={buttons}>This is a Dialog
    with drag enabled{' '}</DialogComponent>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  const buttons: object = [
    {
      buttonModel: {
        content: 'OK',

```

```

        cssClass: 'e-flat',
        isPrimary: true,
    },
    click: () => {
        setDialogVisibility(false);
    },
},
{
    buttonModel: {
        content: 'Cancel',
        cssClass: 'e-flat',
    },
    click: () => {
        setDialogVisibility(false);
    },
},
];
function handleClick() {
    setDialogVisibility(true);
}
function dialogClose() {
    setDialogVisibility(false);
}
return (
    <div className="App" id="dialog-target">
        <button
            className="e-control e-btn"
            id="targetButton1"
            role="button"
            onClick={handleClick}>Open</button>
        <DialogComponent
            width="250px"
            target="#dialog-target"
            visible={visibility}
            close={dialogClose}
            header="Dialog"
            allowDragging={true}
            showCloseIcon={true}
            buttons={buttons}>This is a Dialog with drag enabled{ '
        }</DialogComponent>
    </div>
    );
}
export default App;

```

Positioning

The Dialog position can be set through the [position](#) property by providing X and Y coordinates. The Dialog can be positioned inside the target container based on the given X and Y values.

For example `<!-- markdownlint-disable MD033 --> <code>position:{ X:'center', Y:'center' }</code>` the possible values

- for X is: left, center, right (or) any offset value
- for Y is: top, center, bottom (or) any offset value

The below sample demonstrates the different Dialog positions.

APP.JSX

```
{% raw %}
import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState, useRef } from 'react';
function App() {
    const [visibility, setDialogVisibility] = useState(true);
    const defaultDialog = useRef(null);
    const positioningInstance = useRef(null);
    const position = { X: 'center', Y: 'center' };
    function changePosition(event) {
        defaultDialog.current.position = {
            X: event.currentTarget.value.split(' ')[0],
            Y: event.currentTarget.value.split(' ')[1],
        };
        positioningInstance.current.innerHTML =
            'Position: {X: "' +
                event.currentTarget.value.split(' ')[0] +
                '", Y: "' +
                event.currentTarget.value.split(' ')[1] +
                '"}';
        const txt = event.target.parentElement
            .querySelector('.e-label')
            .innerText.split(' ');
        positioningInstance.current.innerHTML =
            'Position: { X: "' + txt[0] + '", Y: "' + txt[1] + '" }';
    }
    function dialogClose() {
        setDialogVisibility(false);
    }
    function dialogOpen() {
        setDialogVisibility(true);
    }
    function footerTemplate() {
        return (<span ref={positioningInstance} style={{ float: 'left',
paddingLeft: '15px' }}/>);
    }
    return (<div className="App" id="dialog-target">
        <DialogComponent id="defaultDialog" header="Choose a Dialog Position"
visible={visibility} position={position} footerTemplate={footerTemplate}
width="402px" ref={defaultDialog} target="#dialog-target" open={dialogOpen}
close={dialogClose} closeOnEscape={false}>
            <table id="poschange">
                <tbody>
                    <tr>
                        <td>
                            <RadioButtonComponent id="radio1" label="Left Top"
value="left top" name="xy" onClick={changePosition}/>
                        </td>
                        <td>
                            <RadioButtonComponent id="radio2" label="Center Top"
value="center top" name="xy" onClick={changePosition}/>
                        </td>
                    </tr>
                </tbody>
            </table>
        </DialogComponent>
    </div>);
}
```



```

        <td>
          <RadioButtonComponent id="radio3" label="Right Top"
value="right top" name="xy" onClick={changePosition}/>
        </td>
      </tr>
      <tr>
        <td>
          <RadioButtonComponent id="radio4" label="Left Center"
value="left center" name="xy" onClick={changePosition}/>
        </td>
        <td>
          <RadioButtonComponent id="radio5" checked={true}
label="Center Center" value="center center" name="xy"
onClick={changePosition}/>
        </td>
        <td>
          <RadioButtonComponent id="radio6" label="Right Center"
value="right center" name="xy" onClick={changePosition}/>
        </td>
      </tr>
      <tr>
        <td>
          <RadioButtonComponent id="radio7" label="Left Bottom"
value="left bottom" name="xy" onClick={changePosition}/>
        </td>
        <td>
          <RadioButtonComponent id="radio8" label="Center Bottom"
value="center bottom" name="xy" onClick={changePosition}/>
        </td>
        <td>
          <RadioButtonComponent id="radio9" label="Right Bottom"
value="right bottom" name="xy" onClick={changePosition}/>
        </td>
      </tr>
    </tbody>
  </table>
</DialogComponent>
</div>);
}
export default App;
{% enddraw %}

```

APP.TSX

```

{% raw %}
import { RadioButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
import { useState, useRef } from 'react';
function App() {
  const [visibility, setDialogVisibility] = useState(true);
  const defaultDialog = useRef<DialogComponent>(null);
  const positioningInstance = useRef<HTMLElement>(null);
  const position: object = { X: 'center', Y: 'center' };
  function changePosition(event: any): void {
    defaultDialog.current.position = {

```

```

    X: event.currentTarget.value.split(' ')[0],
    Y: event.currentTarget.value.split(' ')[1],
  };
  positioningInstance.current.innerHTML =
    'Position: {X: "' +
    event.currentTarget.value.split(' ')[0] +
    '", Y: "' +
    event.currentTarget.value.split(' ')[1] +
    '"}';
  const txt: string[] = event.target.parentElement
    .querySelector('.e-label')
    .innerText.split(' ');
  positioningInstance.current.innerHTML =
    'Position: { X: "' + txt[0] + '", Y: "' + txt[1] + '" }';
}
function dialogClose() {
  setDialogVisibility(false);
}
function dialogOpen() {
  setDialogVisibility(true);
}
function footerTemplate(): JSX.Element {
  return (
    <span
      ref={positioningInstance}
      style={{ float: 'left', paddingLeft: '15px' }}
    />
  );
}
return (
  <div className="App" id="dialog-target">
    <DialogComponent
      id="defaultDialog"
      header="Choose a Dialog Position"
      visible={visibility}
      position={position}
      footerTemplate={footerTemplate}
      width="402px"
      ref={defaultDialog}
      target="#dialog-target"
      open={dialogOpen}
      close={dialogClose}
      closeOnEscape={false}
    >
      <table id="poschange">
        <tbody>
          <tr>
            <td>
              <RadioButtonComponent
                id="radio1"
                label="Left Top"
                value="left top"
                name="xy"
                onClick={changePosition}
              />
            </td>
          </tr>
        </tbody>
      </table>
    </DialogComponent>
  </div>
);

```

```

        <RadioButtonComponent
            id="radio2"
            label="Center Top"
            value="center top"
            name="xy"
            onClick={changePosition}
        />
    </td>
    <td>
        <RadioButtonComponent
            id="radio3"
            label="Right Top"
            value="right top"
            name="xy"
            onClick={changePosition}
        />
    </td>
</tr>
<tr>
    <td>
        <RadioButtonComponent
            id="radio4"
            label="Left Center"
            value="left center"
            name="xy"
            onClick={changePosition}
        />
    </td>
    <td>
        <RadioButtonComponent
            id="radio5"
            checked={true}
            label="Center Center"
            value="center center"
            name="xy"
            onClick={changePosition}
        />
    </td>
    <td>
        <RadioButtonComponent
            id="radio6"
            label="Right Center"
            value="right center"
            name="xy"
            onClick={changePosition}
        />
    </td>
</tr>
<tr>
    <td>
        <RadioButtonComponent
            id="radio7"
            label="Left Bottom"
            value="left bottom"
            name="xy"
            onClick={changePosition}
        />
    </td>

```

```

        </td>
        <td>
          <RadioButtonComponent
            id="radio8"
            label="Center Bottom"
            value="center bottom"
            name="xy"
            onClick={changePosition}
          />
        </td>
        <td>
          <RadioButtonComponent
            id="radio9"
            label="Right Bottom"
            value="right bottom"
            name="xy"
            onClick={changePosition}
          />
        </td>
      </tr>
    </tbody>
  </table>
</DialogComponent>
</div>
);
}
export default App;
{% enddraw %}

```

See Also

- [Load dialog content using AJAX](#)
- [How to position the dialog on center of the page on scrolling](#)
- [Prevent closing of modal dialog](#)
- [Close dialog while click on outside of dialog](#)
- [How to make a reusable alert and confirm dialog](#)

Template in React Dialog component

In Dialog the template support is provided to header, content and footer sections. So any text or HTML content can be appending in these sections.

Header

The Dialog header content can be provided through the [header](#) property, and it will allow both text and any HTML content as a string.

Also in header, close button is provided as built-in support, and this can be enabled through the [showCloseIcon](#) property.

Footer

The Dialog footer can be enabled by adding built-in [buttons](#) or providing any HTML string through the [footerTemplate](#).

The [buttons](#) and [footerTemplate](#) properties can't be used at the same time.

Content

The Dialog content can be provided through the [content](#) property, and it accepts both text and HTML string as content.

The below example demonstrates the usage of header, footer and content template in the Dialog.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  buttonInstance;
  dialogTextInstance;
  handleClick = () => {
    this.dialogInstance.show();
  };
  header() {
    return (<div>
      
      <div title="Nancy" className="e-icon-settings dlg-template">Nancy</div>
    </div>);
  }
  footerTemplate() {
    return (<div>
      <input id="inVal" className="e-input" type="text"
placeholder="Enter your message here!"/>
      <button id="sendButton" className="e-control e-btn e-primary"
data-ripple="true">Send</button>
    </div>);
  }
  dialogClose = () => {
    this.buttonInstance.style.display = 'inline-block';
  };
  dialogOpen = () => {
    this.buttonInstance.style.display = 'none';
  };
  keyDown = (e) => {
    if (e.keyCode === 13) {
      this.updateTextValue();
    }
  };
  updateTextValue = () => {
    const enteredVal = document.getElementById('inVal');
    const dialogTextElement =
document.getElementsByClassName('dialogText')[0];
    if (enteredVal.value !== '') {
      dialogTextElement.innerHTML = enteredVal.value;
    }
    enteredVal.value = '';
  };
  componentDidMount() {
```

```

        setTimeout(() => {
            document.getElementById('sendButton').onkeydown = (e) => {
                if (e.keyCode === 13) {
                    this.updateTextValue();
                }
            };
            document.getElementById('inVal').onkeydown = (e) => {
                if (e.keyCode === 13) {
                    this.updateTextValue();
                }
            };
            document.getElementById('sendButton').onclick = () => {
                this.updateTextValue();
            };
        });
    }
    render() {
        return (<div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
                ref={buttonElement => this.buttonInstance = buttonElement} role='button'
                onClick={this.handleClick = this.handleClick}>Open</button>
            <DialogComponent width='350px' target='#dialog-target'
                header={this.header} footerTemplate={this.footerTemplate}
                showCloseIcon={true} open={this.dialogOpen} close={this.dialogClose}
                ref={dialog => this.dialogInstance = dialog}>
                <div className="dialogContent">
                    <span className="dialogText">Greetings Nancy! When
will you share me the source files of the project?</span>
                </div>
            </DialogComponent>
        </div>);
    }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public dialogInstance: DialogComponent;
    public buttonInstance: HTMLElement;
    public dialogTextInstance: HTMLSpanElement;
    public handleClick = () => {
        this.dialogInstance.show();
    }
    public header(): JSX.Element {
        return (
            <div>
                
                <div title="Nancy" className="e-icon-settings dlg-template">Nancy</div>
            </div>
        )
    }
}

```

```

    }
    public footerTemplate(): JSX.Element {
        return (
            <div>
                <input id="inVal" className="e-input" type="text"
placeholder="Enter your message here!"/>
                <button id="sendButton" className="e-control e-btn e-primary"
data-ripple="true">Send</button>
            </div>
        )
    }
    public dialogClose = () => {
        this.buttonInstance.style.display='inline-block';
    }
    public dialogOpen = () => {
        this.buttonInstance.style.display='none';
    }
    public keyDown = (e: any) => {
        if (e.keyCode === 13) { this.updateTextValue(); }
    }
    public updateTextValue = () => {
        const enteredVal: HTMLInputElement = document.getElementById('inVal')
as HTMLInputElement;
        const dialogTextElement: HTMLElement =
document.getElementsByClassName('dialogText')[0] as HTMLElement;
        if (enteredVal.value !== '') {
            dialogTextElement.innerHTML = enteredVal.value;
        }
        enteredVal.value = '';
    }
    public componentDidMount() {
        setTimeout(() => {
            (document as any).getElementById('sendButton').onkeydown = (e:
any) => {
                if (e.keyCode === 13) { this.updateTextValue(); }
            };
            (document as any).getElementById('inVal').onkeydown = (e: any) =>
{
                if (e.keyCode === 13) { this.updateTextValue(); }
            };
            (document as any).getElementById('sendButton').onclick = (): void
=> {
                this.updateTextValue();
            };
        });
    }
    public render() {
        return (
            <div className="App" id='dialog-target'>
                <button className='e-control e-btn' id='targetButton1'
ref={buttonElement => this.buttonInstance = buttonElement!} role='button'
onClick={this.handleClick = this.handleClick}>Open</button>
                <DialogComponent width='350px' target='#dialog-target'
header={this.header} footerTemplate={this.footerTemplate}
showCloseIcon={true}
open= {this.dialogOpen} close= {this.dialogClose}
ref={dialog => this.dialogInstance = dialog!}>

```

```

        <div className="dialogContent">
            <span className="dialogText">Greetings Nancy! When
will you share me the source files of the project?</span>
        </div>
    </DialogComponent>
</div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    let buttonInstance;
    let dialogTextInstance;
    function handleClick() {
        dialogInstance.show();
    }
    function header() {
        return <div>
            
            <div title="Nancy" className="e-icon-settings dlg-template">Nancy</div>
        </div>;
    }
    function footerTemplate() {
        return <div>
            <input id="inVal" className="e-input" type="text"
placeholder="Enter your message here!"/>
            <button id="sendButton" className="e-control e-btn e-primary"
data-ripple="true">Send</button>
        </div>;
    }
    function dialogClose() {
        buttonInstance.style.display = 'inline-block';
    }
    function dialogOpen() {
        buttonInstance.style.display = 'none';
    }
    function keyDown(e) {
        if (e.keyCode === 13) {
            updateTextValue();
        }
    }
    function updateTextValue() {
        const enteredVal = document.getElementById('inVal');
        const dialogTextElement =
document.getElementsByClassName('dialogText')[0];
        if (enteredVal.value !== '') {

```



```

        dialogTextElement.innerHTML = enteredVal.value;
    }
    enteredVal.value = '';
}
function componentCreated() {
    setTimeout(() => {
        document.getElementById('sendButton').onkeydown = (e) => {
            if (e.keyCode === 13) {
                updateTextValue();
            }
        };
        document.getElementById('inVal').onkeydown = (e) => {
            if (e.keyCode === 13) {
                updateTextValue();
            }
        };
        document.getElementById('sendButton').onclick = () => {
            updateTextValue();
        };
    });
}
return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
    ref={buttonElement => buttonInstance = buttonElement} role='button'
    onClick={handleClick}>Open</button>
    <DialogComponent created = {componentCreated} width='350px'
    target='#dialog-target' header={header} footerTemplate={footerTemplate}
    showCloseIcon={true} open={dialogOpen} close={dialogClose} ref={dialog =>
    dialogInstance = dialog}>
        <div className="dialogContent">
            <span className="dialogText">Greetings Nancy! When
            will you share me the source files of the project?</span>
        </div>
    </DialogComponent>
</div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    let buttonInstance: HTMLElement;
    let dialogTextInstance: HTMLSpanElement;
    function handleClick () {
        dialogInstance.show();
    }
    function header(): JSX.Element {
        return (
            <div>
                
            </div>
        );
    }
}

```

```

        <div title="Nancy" className="e-icon-settings dlg-
template">Nancy</div>
    </div>
    )
}
function footerTemplate(): JSX.Element {
    return (
        <div>
            <input id="inVal" className="e-input" type="text"
placeholder="Enter your message here!"/>
            <button id="sendButton" className="e-control e-btn e-primary"
data-ripple="true">Send</button>
        </div>
    )
}
function dialogClose () {
    buttonInstance.style.display='inline-block';
}
function dialogOpen () {
    buttonInstance.style.display='none';
}
function keyDown (e: any) {
    if (e.keyCode === 13) { updateTextValue(); }
}
function updateTextValue() {
    const enteredVal: HTMLInputElement = document.getElementById('inVal')
as HTMLInputElement;
    const dialogTextElement: HTMLElement =
document.getElementsByClassName('dialogText')[0] as HTMLElement;
    if (enteredVal.value !== '') {
        dialogTextElement.innerHTML = enteredVal.value;
    }
    enteredVal.value = '';
}
function componentCreated() {
    setTimeout(() => {
        (document as any).getElementById('sendButton').onkeydown = (e:
any) => {
            if (e.keyCode === 13) {updateTextValue(); }
        };
        (document as any).getElementById('inVal').onkeydown = (e: any) =>
{
            if (e.keyCode === 13) {updateTextValue(); }
        };
        (document as any).getElementById('sendButton').onclick = (): void
=> {
            updateTextValue();
        };
    });
}
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
ref={buttonElement => buttonInstance = buttonElement!} role='button'
onClick={handleClick}>Open</button>

```

```

    <DialogComponent created = {componentCreated} width='350px'
    target='#dialog-target' header={header} footerTemplate={footerTemplate }
    showCloseIcon={true}
      open= {dialogOpen} close= {dialogClose}
      ref={dialog => dialogInstance = dialog!}>
        <div className="dialogContent">
          <span className="dialogText">Greetings Nancy! When
will you share me the source files of the project?</span>
        </div>
      </DialogComponent>
    </div>
  );
}
export default App;

```

See Also

- [How to add an icon to dialog buttons](#)
- [How to customize the dialog appearance](#)

Animation in React Dialog component

The Dialog can be animated during the open and close actions. Also, user can customize animation's [delay](#), [duration](#) and [effect](#) by using [animationSettings](#) property.

<!-- markdownlint-disable MD033 -->

delay	The Dialog animation will start with the mentioned delay
duration	Specifies the animation duration to complete with one animation cycle
effect	<p>Specifies the animation effects of Dialog open and close actions effect.</p> <p>List of supported animation effects: 'Fade' 'FadeZoom' 'FlipLeftDown' 'FlipLeftUp' 'FlipRightDown' 'FlipRightUp' 'FlipXDown' 'FlipXUp' 'FlipYLeft' 'FlipYRight' 'SlideBottom' 'SlideLeft' 'SlideRight' 'SlideTop' 'Zoom' 'None'</p> <p>If the user sets 'Fade' effect, then the Dialog will open with 'FadeIn' effect and close with 'FadeOut' effect</p>

In the below sample, **Zoom** effect is enabled. So, The Dialog will open with **ZoomIn** and close with **ZoomOut** effects.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  settings = { effect: 'Zoom', duration: 400, delay: 0 };

```

```

    buttons = [{
      buttonModel: {
        content: 'OK',
        cssClass: 'e-flat',
        isPrimary: true,
      },
      'click': () => {
        this.dialogInstance.hide();
      }
    },
    {
      buttonModel: {
        content: 'Cancel',
        cssClass: 'e-flat'
      },
      'click': () => {
        this.dialogInstance.hide();
      }
    }
  ]];
  handleClick() {
    this.dialogInstance.show();
  }
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' animationSettings={this.settings}
        target='#dialog-target' header='Dialog' showCloseIcon={true}
        buttons={this.buttons} ref={dialog => this.dialogInstance = dialog}>
        Dialog enabled with Zoom effect</DialogComponent>
      </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  public dialogInstance: DialogComponent;
  public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    }
  }
  ]
}

```

```

    },
    'click': () => {
        this.dialogInstance.hide();
    }
  ]];
  public handleClick() {
    this.dialogInstance.show();
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
          onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent width='250px' animationSettings={this.settings}
          target='#dialog-target' header='Dialog' showCloseIcon={true}
          buttons={this.buttons} ref={dialog => this.dialogInstance = dialog!}>
          Dialog enabled with Zoom effect</DialogComponent>
        </div>);
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  const settings = { effect: 'Zoom', duration: 400, delay: 0 };
  let buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      dialogInstance.hide();
    }
  }
  ]];
  function handleClick() {
    dialogInstance.show();
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
      role='button' onClick={handleClick.bind(this)}>Open</button>

```

```

        <DialogComponent width='250px' animationSettings={settings}
target='#dialog-target' header='Dialog' showCloseIcon={true}
buttons={buttons} ref={dialog => dialogInstance = dialog}>
        Dialog enabled with Zoom effect</DialogComponent>
    </div>;
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    const settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    let buttons: any = [{
        buttonModel: {
            content: 'OK',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            dialogInstance.hide();
        }
    }
    ];
    function handleClick() {
        dialogInstance.show();
    }
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
role='button' onClick={handleClick.bind(this)}>Open</button>
            <DialogComponent width='250px' animationSettings={settings}
target='#dialog-target' header='Dialog' showCloseIcon={true}
buttons={buttons} ref={dialog => dialogInstance = dialog!}>
                Dialog enabled with Zoom effect</DialogComponent>
            </div>
        );
}
export default App;

```

Resize in React Dialog component

The Dialog supports resizing feature. To resize the dialog, we have to select and resize it by using its handle (grip) or hovering on any of the edges or borders of the dialog within the sample container.

The resizable dialog can be created by setting the [enableResize](#) property to true, which is used to change the size of a dialog dynamically and view its content with expanded mode. The [resizeHandles](#) property can also be configured for all the which directions in which the dialog should be resized. When you configure the target property along with the [enableResize](#) property, the dialog can be resized within its specified target container.

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false });
    }
  }
  ]];
  constructor(props) {
    super(props);
    this.state = {
      hideDialog: true
    };
  }
  handleClick() {
    this.setState({ hideDialog: true });
  }
  dialogClose = () => {
    this.setState({ hideDialog: false });
  };
  render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
          onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent width='250px' target='#dialog-target'
          visible={this.state.hideDialog} close={this.dialogClose} header='Dialog'
          enableResize={true} resizeHandles={['All']} allowDragging={true}
          showCloseIcon={true} buttons={this.buttons}>
          This is a Dialog with drag enabled </DialogComponent>
        </div>
      );
  }
}
export default App;
```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {hideDialog: boolean;}> {
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.setState({ hideDialog: false })
    }
  }
  ]];
  constructor(props: {}) {
    super(props);
    this.state = {
      hideDialog : true
    };
  }
  public handleClick() {
    this.setState({ hideDialog: true })
  }
  public dialogClose = () => {
    this.setState({ hideDialog: false })
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent width='250px' target='#dialog-target' visible =
{this.state.hideDialog} close = {this.dialogClose} header='Dialog'
enableResize={true} resizeHandles={['All']} allowDragging={true}
showCloseIcon={true} buttons={this.buttons}>
          This is a Dialog with drag enabled </DialogComponent>
        </div>);
  }
}
export default App;

```

[Functional-component]**APP.JSX**

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";

```



```

function App() {
  const [status, setStatus] = React.useState({ hideDialog: true });
  let buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      setStatus({ hideDialog: false });
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      setStatus({ hideDialog: false });
    }
  }
  ];
  function handleClick() {
    setState({ hideDialog: true });
  }
  function dialogClose() {
    setStatus({ hideDialog: false });
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
    role='button' onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' target='#dialog-target'
    visible={status.hideDialog} close={dialogClose} header='Dialog'
    enableResize={true} resizeHandles={['All']} allowDragging={true}
    showCloseIcon={true} buttons={buttons}>
      This is a Dialog with drag enabled </DialogComponent>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  const [status, setStatus] = React.useState({ hideDialog: true });
  let buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      setStatus({ hideDialog: false })
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      setStatus({ hideDialog: false })
    }
  }
  ];
}

```

```

        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            setStatus({ hideDialog: false })
        }
    }];
    function handleClick() {
        setState({ hideDialog: true })
    }
    function dialogClose(){
        setStatus({ hideDialog: false })
    }
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
            role='button' onClick={handleClick.bind(this)}>Open</button>
            <DialogComponent width='250px' target='#dialog-target' visible =
            {status.hideDialog} close = {dialogClose} header='Dialog'
            enableResize={true} resizeHandles={['All']} allowDragging={true}
            showCloseIcon={true} buttons={buttons}>
                This is a Dialog with drag enabled </DialogComponent>
            </div>
        );
    }
}
export default App;

```

Localization in React Dialog component

[Localization](#) library allows to localize the default text content of Dialog. In Dialog, The close button's tooltip text alone will be localize based on culture.

| Locale key | en-US (default) |

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

| close | Close |

Loading translations

To load translation object in an application use `load` function of `L10n` class.

In the below sample, `French` culture is set to Dialog and change the close button's tooltip text.

[Class-component]

APP.JSX

```

import { L10n } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
// Load French culture for Dialog close button tooltip text
L10n.load({
    'fr-BE': {
        'dialog': {
            'close': "Fermer"
        }
    }
})

```

```

});
class App extends React.Component {
  content = () => {
    return (<div>
      Dialogue avec la culture française
    </div>);
  };
  render() {
    return (<div className="App" id='dialog-target'>
      <DialogComponent width='250px' locale='fr-BE' content={this.content}
        header='Dialog' closeOnEscape={false} showCloseIcon={true} target='#dialog-target' />
    </div>);
  }
}
export default App;

```

APP.TSX

```

import { L10n } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
// Load French culture for Dialog close button tooltip text
L10n.load({
  'fr-BE': {
    'dialog': {
      'close': "Fermer"
    }
  }
});
class App extends React.Component<{}, {}> {
  public content = (): JSX.Element => {
    return (
      <div>
        Dialogue avec la culture française
      </div>
    )
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <DialogComponent
          width='250px' locale= 'fr-BE' content={this.content} header='Dialog'
          closeOnEscape={false}
          showCloseIcon={true} target='#dialog-target' />
      </div>
    );
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { L10n } from '@syncfusion/ej2-base';

```

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
// Load French culture for Dialog close button tooltip text
L10n.load({
  'fr-BE': {
    'dialog': {
      'close': "Fermer"
    }
  }
});
function App() {
  function content() {
    return (<div>
      Dialogue avec la culture française
    </div>);
  }
  return (<div className="App" id='dialog-target'>
    <DialogComponent width='250px' locale='fr-BE' content={content}
    header='Dialog' closeOnEscape={false} showCloseIcon={true} target='#dialog-
    target' />
    </div>);
}
export default App;
```

APP.TSX

```
import { L10n } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
// Load French culture for Dialog close button tooltip text
L10n.load({
  'fr-BE': {
    'dialog': {
      'close': "Fermer"
    }
  }
});
function App() {
  function content(): JSX.Element {
    return (
      <div>
        Dialogue avec la culture française
      </div>
    )
  }
  return (
    <div className="App" id='dialog-target'>
      <DialogComponent width='250px' locale= 'fr-BE' content={content}
      header='Dialog' closeOnEscape={false} showCloseIcon={true} target='#dialog-
      target' />
      </div>
    );
}
export default App;
```

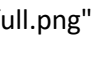
Accessibility in React Dialog component

The Dialog component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dialog component is outlined below.

| Accessibility Criteria | Compatibility |

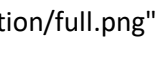
| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Dialog is characterized with complete ARIA Accessibility support which helps to be accessible by on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in [WAI ARIA Accessibility Practices](#).

The Dialog component uses the **Dialog** role and following ARIA properties to its element based on its state.

| Property | Functionalities |

| --- | --- |

| aria-describedby | It indicates the Dialog content description is notified to the user through assistive technologies. |

| aria-labelledby | The Dialog title is notified to the user through assistive technologies. |

| aria-modal | For modal dialog its value is true and non-modal dialog its value is false |

| aria-grabbed | Enable the draggable Dialog and starts dragging it its value is true and stopping the drag its value is false |

Keyboard interaction

Keyboard interaction of Dialog component has been designed based on [WAI-ARIA Practices](#) described for Dialog. User can use the following shortcut keys to interact with the Dialog.

<!-- markdownlint-disable MD033 -->

Keyboard shortcuts	Actions
Esc	Closes the Dialog. This functionality can be controlled by using closeOnEscape
Enter	When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will trigger. Enter key is not working when the Dialog content contains any text area with initial focus
Ctrl + Enter	When the Dialog content contains text area and it is in focus state, and press the Ctrl + Enter key to call the click event function associated with primary button
Tab	Focus will be changed within the Dialog elements
Shift + Tab	The Focus will be changed to previous focusable element within the Dialog elements. When focusing a first focusable element in the Dialog, then press the shift + tab key, it will change the focus to last focusable element

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from 'react';
class App extends React.Component {
  dialogInstance;
  buttons = [{
    buttonModel: {
```

```

        content: 'Submit',
        cssClass: 'e-flat',
        isPrimary: true,
      },
      'click': () => {
        this.dialogInstance.hide();
      }
    ]];
    handleClick() {
      this.dialogInstance.show();
    }
    render() {
      return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={this.handleClick =
        this.handleClick.bind(this)}>Open</button>

        <DialogComponent width='400px' target='#dialog-target'
        header='Feedback' showCloseIcon={true} buttons={this.buttons} ref={dialog =>
        this.dialogInstance = dialog}>
          <form>
            <div className='form-group'>
              <label htmlFor='email'> Email:</label>
              <input type='email' className='form-control' id='email' />
            </div>
            <div className='form-group'>
              <label htmlFor='comment'>Comments:</label>
              <textarea className='form-control' id='comment' />
            </div>
          </form>
        </DialogComponent>
      </div>);
    }
  }
  export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  public dialogInstance: DialogComponent;
  public buttons: any = [{
    buttonModel: {
      content: 'Submit',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];

  public handleClick() {
    this.dialogInstance.show();
  }
}

```

```

public render() {
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={this.handleClick =
        this.handleClick.bind(this)}>Open</button>

      <DialogComponent width='400px' target='#dialog-target'
        header='Feedback' showCloseIcon={true} buttons={this.buttons} ref={dialog
        => this.dialogInstance = dialog!}>
        <form>
          <div className='form-group'>
            <label htmlFor='email'> Email:</label>
            <input type='email' className='form-control' id='email' />
          </div>
          <div className='form-group'>
            <label htmlFor='comment'>Comments:</label>
            <textarea className='form-control' id='comment' />
          </div>
        </form>
      </DialogComponent>
    </div>);
}
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  let buttons = [{
    buttonModel: {
      content: 'Submit',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  }];
  function handleClick() {
    dialogInstance.show();
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
      role='button' onClick={handleClick.bind(this)}>Open</button>

    <DialogComponent width='400px' target='#dialog-target'
      header='Feedback' showCloseIcon={true} buttons={buttons} ref={dialog =>
      dialogInstance = dialog}>
      <form>

```



```

        <div className='form-group'>
            <label htmlFor='email'> Email:</label>
            <input type='email' className='form-control' id='email' />
        </div>
        <div className='form-group'>
            <label htmlFor='comment'>Comments:</label>
            <textarea className='form-control' id='comment' />
        </div>
    </form>
</DialogComponent>
</div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    let buttons: any = [{
        buttonModel: {
            content: 'Submit',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    }];

    function handleClick() {
        dialogInstance.show();
    }

    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
            role='button' onClick={ handleClick.bind(this) }>Open</button>

            <DialogComponent width='400px' target='#dialog-target'
            header='Feedback' showCloseIcon={true} buttons={buttons} ref={dialog =>
            dialogInstance = dialog!}>
                <form>
                    <div className='form-group'>
                        <label htmlFor='email'> Email:</label>
                        <input type='email' className='form-control' id='email' />
                    </div>
                    <div className='form-group'>
                        <label htmlFor='comment'>Comments:</label>
                        <textarea className='form-control' id='comment' />
                    </div>
                </form>
            </DialogComponent>
        </div>
    );
}

```

```
}  
export default App;
```

See Also

- [Show dialog with full-screen](#)

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Style in React Dialog 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 dialog header

Use the following CSS to customize the dialog header properties.

```
`css  
.e-dialog .e-dlg-header {  
  color: green;  
  font-size: 20px;  
  font-weight: normal;  
}
```

,

Customizing the dialog content

Use the following CSS to customize the dialog content properties.

```
`css  
.e-dialog .e-dlg-content {  
  color: red;  
  font-size: 10px;  
  font-weight: normal;  
  line-height: normal;  
}
```

,

Customizing modal dialog overlay

Use the following CSS to customize the modal dialog overlay.

```
`css
.e-dlg-overlay {
background-color: slategray;
opacity: 0.6;
}
`
```

Customizing the dialog resize icon

Use the following CSS to customize the dialog resize icon.

```
`css
/ To change the icon content /
.e-dialog .e-south-east::before, .e-dialog .e-south-west::before {
content: '\f047';
}
/ To set the icon pack /
.e-dialog .e-resize-handle {
font: normal normal normal 14px/1 FontAwesome;
}
`
```

The above CSS demonstration uses the font awesome icon.

Customizing the dialog close button

Use the following CSS to customize the dialog close button.

```
`css
/ To specify font size and color /
.e-dialog .e-btn .e-btn-icon.e-icon-dlg-close {
font-size: 12px;
color: red;
}
`
```

Customizing the dialog footer button

Use the following CSS to customize the dialog footer button.

```
`css
/ To specify font color, background color and border color /
```

```
.e-btn.e-flat.e-primary, .e-css.e-btn.e-flat.e-primary {
background-color: transparent;
border-color: transparent;
color: blue;
}
,
```

How To

Create nested dialog in React Dialog component

A Dialog can be nested within another Dialog. The below sample contains parent and child Dialog (inner Dialog).

Step 1:

Render two Dialog components in a page.

Step 2:

Set the inner Dialog target as `.outerDialog`.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  innerDialogInstance;
  handleClick() {
    this.dialogInstance.show();
  }
  nestedbuttonClick = () => {
    this.innerDialogInstance.show();
  };
  render() {
    const effect = { effect: 'None' };
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent header='Outer Dialog' cssClass="outerDialog"
showCloseIcon={true} width='400px' height='300px' ref={dialog =>
this.dialogInstance = dialog} target='#dialog-target' closeOnEscape={false}
animationSettings={effect}>
        <button className="e-control e-btn" id="innerButton"
onClick={this.nestedbuttonClick} role="button">Open InnerDialog</button>
      </DialogComponent>

      <DialogComponent id='innerDialog' header='Inner Dialog'
showCloseIcon={true} width='250px' height='150px' ref={dialog =>
this.innerDialogInstance = dialog} animationSettings={effect}
closeOnEscape={false} target='.outerDialog'> This is a Nested Dialog
    </DialogComponent>
    </div>);
```

```

    }
  }
  export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogInstance: DialogComponent;
  public innerDialogInstance: DialogComponent;
  public handleClick() {
    this.dialogInstance.show();
  }
  public nestedbuttonClick = () => {
    this.innerDialogInstance.show();
  }

  public render() {
    const effect: any = { effect: 'None' };
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
          onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent header='Outer Dialog' cssClass="outerDialog"
          showCloseIcon={true} width='400px' height='300px'
            ref={dialog => this.dialogInstance = dialog!} target='#dialog-
            target' closeOnEscape={false} animationSettings={effect}>
          <button className="e-control e-btn" id="innerButton"
            onClick={this.nestedbuttonClick} role="button" >Open InnerDialog</button>
        </DialogComponent>

        <DialogComponent id='innerDialog' header='Inner Dialog'
          showCloseIcon={true} width='250px' height='150px'
            ref={dialog => this.innerDialogInstance = dialog!}
            animationSettings={effect} closeOnEscape={false}
            target='.outerDialog'> This is a Nested Dialog </DialogComponent>
      </div>);
    }
  }
  export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  let innerDialogInstance;
  function handleClick() {
    dialogInstance.show();
  }
  function nestedbuttonClick() {
    innerDialogInstance.show();
  }

```

```

    }
    const effect = { effect: 'None' };
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent header='Outer Dialog' cssClass="outerDialog"
showCloseIcon={true} width='400px' height='300px' ref={dialog =>
dialogInstance = dialog} target='#dialog-target' closeOnEscape={false}
animationSettings={effect}>
        <button className="e-control e-btn" id="innerButton"
onClick={nestedbuttonClick} role="button">Open InnerDialog</button>
      </DialogComponent>
      <DialogComponent id='innerDialog' header='Inner Dialog'
showCloseIcon={true} width='250px' height='150px' ref={dialog =>
innerDialogInstance = dialog} animationSettings={effect}
closeOnEscape={false} target='.outerDialog'> This is a Nested Dialog
    </DialogComponent></div>);
  }
  export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  let innerDialogInstance: DialogComponent;
  function handleClick() {
    dialogInstance.show();
  }
  function nestedbuttonClick () {
    innerDialogInstance.show();
  }
  const effect: any = { effect: 'None' };
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent header='Outer Dialog' cssClass="outerDialog"
showCloseIcon={true} width='400px' height='300px'
        ref={dialog => dialogInstance = dialog!} target='#dialog-target'
closeOnEscape={false} animationSettings={effect}>
        <button className="e-control e-btn" id="innerButton"
onClick={nestedbuttonClick} role="button" >Open InnerDialog</button>
      </DialogComponent>
      <DialogComponent id='innerDialog' header='Inner Dialog'
showCloseIcon={true} width='250px' height='150px'
        ref={dialog => innerDialogInstance = dialog!}
animationSettings={effect} closeOnEscape={false}
target='.outerDialog'> This is a Nested Dialog </DialogComponent></div>
    );
  }
  export default App;

```

Position the dialog on center of the page on scrolling in React Dialog component

By default, when scroll the page/container Dialog also scrolled along with the page/container. When a user expects to display the Dialog in the same position without scrolling achieving this in sample level as like below. Here added 'e-fixed' class to Dialog element and prevent the scrolling.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  handleClick = () => {
    this.dialogInstance.cssClass = 'e-fixed';
  };
  render() {
    return (<div className="App" id='dialog-target'>
      <DialogComponent header='Dialog' width='250px' ref={dialog =>
        this.dialogInstance = dialog} target='#dialog-target' closeOnEscape={false}>
        <button className="e-control e-btn" id="targetButton" role="button"
          onClick={this.handleClick = this.handleClick}>Prevent Dialog Scroll</button>
      </DialogComponent>
    </div>);
  }
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogInstance: DialogComponent;
  public handleClick = () => {
    this.dialogInstance.cssClass = 'e-fixed';
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <DialogComponent header='Dialog' width='250px' ref={dialog =>
          this.dialogInstance = dialog!}
          target='#dialog-target' closeOnEscape={false}>
          <button className="e-control e-btn" id="targetButton" role="button"
            onClick={this.handleClick = this.handleClick}>Prevent Dialog Scroll</button>
        </DialogComponent>
      </div>
    );
  }
}
export default App;
```

[Functional-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  function handleClick() {
    dialogInstance.cssClass = 'e-fixed';
  }
  return (<div className="App" id='dialog-target'>
    <DialogComponent header='Dialog' width='250px' ref={dialog =>
dialogInstance = dialog} target='#dialog-target' closeOnEscape={false}>
    <button className="e-control e-btn" id="targetButton" role="button"
onClick={handleClick}>Prevent Dialog Scroll</button>
    </DialogComponent>
  </div>);
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  function handleClick () {
    dialogInstance.cssClass = 'e-fixed';
  }

  return (
    <div className="App" id='dialog-target'>
      <DialogComponent header='Dialog' width='250px' ref={dialog =>
dialogInstance = dialog!}
      target='#dialog-target' closeOnEscape={false}>
        <button className="e-control e-btn" id="targetButton" role="button"
onClick={handleClick}>Prevent Dialog Scroll</button>
      </DialogComponent>
    </div>
  );
}
export default App;
```

Load dialog content using ajax in React Dialog component

You can load dialog's content dynamically from external source like external file using AJAX library.

The AJAX library can make the request and load dialog's content using its `success` event.

Refer the following link to learn about how to load dialog content using AJAX.

[AJAX Content](#)

Render a dialog without header in React Dialog component

The dialog can be rendered without header by setting the header property value as empty string or null. By default, dialog is rendered without header.

[Class-component]

APP.JSX


```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }
  ];
  handleClick() {
    this.dialogInstance.show();
  }
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' target='#dialog-target'
        buttons={this.buttons} ref={dialog => this.dialogInstance = dialog}>
        This is a dialog without header </DialogComponent>
      </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogInstance: DialogComponent;
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {

```

```

        content: 'Cancel',
        cssClass: 'e-flat'
    },
    'click': () => {
        this.dialogInstance.hide();
    }
  ]];

  public handleClick() {
    this.dialogInstance.show();
  }

  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent width='250px' target='#dialog-target'
buttons={this.buttons} ref={dialog => this.dialogInstance = dialog!}>
          This is a dialog without header </DialogComponent>
        </div>);
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance;
  const buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      dialogInstance.hide();
    }
  }
  ]];
  function handleClick() {
    dialogInstance.show();
  }
  return (<div className="App" id='dialog-target'>

```

```

    <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent width='250px' target='#dialog-target'
buttons={buttons} ref={dialog => dialogInstance = dialog}>
    This is a dialog without header </DialogComponent>
  </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  const buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      dialogInstance.hide();
    }
  }
  ]];
  function handleClick() {
    dialogInstance.show();
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent width='250px' target='#dialog-target'
buttons={buttons} ref={dialog => dialogInstance = dialog!}>
        This is a dialog without header </DialogComponent>
      </div>
    );
}
export default App;

```

Show dialog with full screen in React Dialog component

You can show the dialog in fullscreen by passing **true** as argument to the dialog [show](#) method.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  buttons = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'Cancel',
      cssClass: 'e-flat'
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }
  ];
  handleClick() {
    this.dialogInstance.show(true);
  }
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick} = this.handleClick.bind(this)>Open</button>
      <DialogComponent header='Dialog' showCloseIcon={true} visible={false}
        width='250px' target='#dialog-target' buttons={this.buttons} ref={dialog =>
          this.dialogInstance = dialog}>
        This is a Dialog with fullscreen display </DialogComponent>
      </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogInstance: DialogComponent;
  public buttons: any = [{
    buttonModel: {
      content: 'OK',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
  }
  ];
}

```

```

        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    ]];
    public handleClick() {
        this.dialogInstance.show(true);
    }
    public render() {
        return (
            <div className="App" id='dialog-target'>
                <button className='e-control e-btn' id='targetButton1' role='button'
                    onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
                <DialogComponent header='Dialog' showCloseIcon={true} visible={false}
                    width='250px' target='#dialog-target' buttons={this.buttons} ref={dialog =>
                        this.dialogInstance = dialog!}>
                    This is a Dialog with fullscreen display </DialogComponent>
            </div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    const buttons = [{
        buttonModel: {
            content: 'OK',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            dialogInstance.hide();
        }
    }
    ];
    function handleClick() {
        dialogInstance.show(true);
    }
    return (<div className="App" id='dialog-target'>

```

```

        <button className='e-control e-btn' id='targetButton1'
        role='button' onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent header='Dialog' showCloseIcon={true}
        visible={false} width='250px' target='#dialog-target' buttons={buttons}
        ref={dialog => dialogInstance = dialog}>
            This is a Dialog with fullscreen display </DialogComponent>
        </div>;
    }
    export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    const buttons: any = [{
        buttonModel: {
            content: 'OK',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'Cancel',
            cssClass: 'e-flat'
        },
        'click': () => {
            dialogInstance.hide();
        }
    }
    ];

    function handleClick() {
        dialogInstance.show(true);
    }

    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1'
            role='button' onClick={handleClick.bind(this)}>Open</button>
            <DialogComponent header='Dialog' showCloseIcon={true}
            visible={false} width='250px' target='#dialog-target' buttons={buttons}
            ref={dialog => dialogInstance = dialog!}>
                This is a Dialog with fullscreen display </DialogComponent>
            </div>
        );
    }
    export default App;

```

Display a dialog with custom position in React Dialog component

By default, the dialog is displayed in the center of the target container. The dialog position can be set using the position property by providing custom X and Y coordinates.

The dialog can be positioned inside the target based on the given X and Y values.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  render() {
    const firstPosition = { X: 160, Y: 14 };
    const secondPosition = { X: 160, Y: 240 };
    const effect = { effect: 'None' };
    return (<div className="App" id='dialog-target'>
      <DialogComponent id='firstDialog' header='Position-01' visible={true}
width='360px' height='120px' target='#dialog-target' closeOnEscape={false}
animationSettings={effect} position={firstPosition}>
        The dialog is positioned at X: 160, Y: 14 coordinates
      </DialogComponent>
      <DialogComponent id='secondDialog' header='Position-02' visible={true}
width='360px' height='120px' target='#dialog-target' closeOnEscape={false}
animationSettings={effect} position={secondPosition}>
        The dialog is positioned at X: 160, Y: 240 coordinates
      </DialogComponent>
    </div>);
  }
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public render() {
    const firstPosition: any = { X: 160, Y: 14 };
    const secondPosition: any = { X: 160, Y: 240 };
    const effect: any = { effect: 'None' };
    return (
      <div className="App" id='dialog-target'>
        <DialogComponent id='firstDialog' header='Position-01' visible={true}
width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect}
position={firstPosition}
>
          The dialog is positioned at X: 160, Y: 14 coordinates
        </DialogComponent>
        <DialogComponent id='secondDialog' header='Position-02' visible={true}
width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect}
position={secondPosition}
>
          The dialog is positioned at X: 160, Y: 240 coordinates
        </DialogComponent>
      </div>);
  }
}
```

```
export default App;
```

[Functional-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    const firstPosition = { X: 160, Y: 14 };
    const secondPosition = { X: 160, Y: 240 };
    const effect = { effect: 'None' };
    return (<div className="App" id='dialog-target'>
        <DialogComponent id='firstDialog' header='Position-01'
visible={true} width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect} position={firstPosition}>
            The dialog is positioned at X: 160, Y: 14 coordinates
        </DialogComponent>

        <DialogComponent id='secondDialog' header='Position-02'
visible={true} width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect} position={secondPosition}>
            The dialog is positioned at X: 160, Y: 240 coordinates
        </DialogComponent>
    </div>);
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    const firstPosition: any = { X: 160, Y: 14 };
    const secondPosition: any = { X: 160, Y: 240 };
    const effect: any = { effect: 'None' };
    return (
        <div className="App" id='dialog-target'>
            <DialogComponent id='firstDialog' header='Position-01'
visible={true}
width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect}
position={firstPosition}
>
                The dialog is positioned at X: 160, Y: 14 coordinates
            </DialogComponent>

            <DialogComponent id='secondDialog' header='Position-02'
visible={true}
width='360px' height='120px' target='#dialog-target'
closeOnEscape={false} animationSettings={effect}
position={secondPosition}
>
                The dialog is positioned at X: 160, Y: 240 coordinates
            </DialogComponent>
        </div>);
}
```



```
}
export default App;
```

Prevent closing of modal dialog in React Dialog component

You can prevent closing of modal dialog by setting the [beforeClose](#) event argument cancel value to true. In the following sample, the dialog is closed when you enter the username value with minimum 4 characters. Otherwise, it will not be closed.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  dialogInstance;
  userName;
  password;
  buttons = [{
    buttonModel: {
      content: 'LOG IN',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];
  dialogContent() {
    return (<div className="login-form">
      <div className='wrap'>
        <div id="heading"/>
        <div className="e-float-input">
          <input id="textvalue" type="text" ref={user =>
this.userName = user} required={true}/>
          <span className="e-float-line"/>
          <label className="e-float-text">Username</label>
        </div>
        <div className="e-float-input">
          <input id="textvalue2" type="password" ref={pwd =>
this.password = pwd} required={true}/>
          <span className="e-float-line"/>
          <label className="e-float-text">Password</label>
        </div>
      </div>
    </div>);
  }
  validation = (args) => {
    if (this.userName.value === "" && this.password.value === "") {
      args.cancel = true;
      alert("Enter the username and password");
    }
    else if (this.userName.value === "") {
      args.cancel = true;
      alert("Enter the username");
    }
  }
}
```

```

        else if (this.userName.value === "") {
            args.cancel = true;
            alert("Enter the password");
        }
        else if (this.userName.value.length < 4) {
            args.cancel = true;
            alert("Username must be minimum 4 characters");
        }
        else {
            args.cancel = false;
            this.userName.value = "";
            this.password.value = "";
        }
    };
    render() {
        return (<div className="App" id='container'>

            <DialogComponent id="dlg-button" width='300px' isModal={true}
target='#container' header='Sign In' showCloseIcon={false}
closeOnEscape={false} beforeClose={this.validation} buttons={this.buttons}
ref={dialog => this.dialogInstance = dialog}>
                <div>{this.dialogContent()}</div>
            </DialogComponent>
        </div>);
    }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public dialogInstance: DialogComponent;
    public userName: HTMLInputElement;
    public password: HTMLInputElement;
    public buttons: any = [{
        buttonModel: {
            content: 'LOG IN',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }];
    public dialogContent(): JSX.Element {
        return (
            <div className="login-form">
                <div className='wrap'>
                    <div id="heading"/>
                    <div className="e-float-input">
                        <input id="textvalue" type="text" ref = {user =>
this.userName = user!} required = {true}/>
                        <span className="e-float-line"/>
                        <label className="e-float-text">Username</label>

```

```

        </div>
        <div className="e-float-input">
            <input id="textvalue2" type="password" ref = {pwd =>
this.password = pwd!} required = {true}/>
            <span className="e-float-line"/>
            <label className="e-float-text">Password</label>
        </div>
    </div>
</div>
)
}

public validation = (args: any): void =>{
    if (this.userName.value === "" && this.password.value === "") {
        args.cancel= true;
        alert("Enter the username and password")
    } else if (this.userName.value === "") {
        args.cancel= true;
        alert("Enter the username")
    } else if (this.userName.value === "") {
        args.cancel= true;
        alert("Enter the password")
    } else if (this.userName.value.length < 4) {
        args.cancel= true;
        alert("Username must be minimum 4 characters")
    } else {
        args.cancel= false;
        this.userName.value = "";
        this.password.value = "";
    }
}

public render() {
    return (
        <div className="App" id='container'>

            <DialogComponent id="dlg-button" width='300px' isModal={true}
target='#container' header='Sign In' showCloseIcon={false} closeOnEscape =
{false}
                beforeClose={this.validation} buttons={this.buttons} ref={dialog =>
this.dialogInstance = dialog!}>
                <div>{this.dialogContent()}</div>
            </DialogComponent>

        </div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;

```

```

let userName;
let password;
const buttons = [{
  buttonModel: {
    content: 'LOG IN',
    cssClass: 'e-flat',
    isPrimary: true,
  },
  'click': () => {
    dialogInstance.hide();
  }
}];
function dialogContent() {
  return (<div className="login-form">
    <div className='wrap'>
      <div id="heading"/>
      <div className="e-float-input">
        <input id="textvalue" type="text" ref={user => userName =
user} required={true}/>
        <span className="e-float-line"/>
        <label className="e-float-text">Username</label>
      </div>
      <div className="e-float-input">
        <input id="textvalue2" type="password" ref={pwd => password
= pwd} required={true}/>
        <span className="e-float-line"/>
        <label className="e-float-text">Password</label>
      </div>
    </div>
  </div>);
}
function validation(args) {
  if (userName.value === "" && password.value === "") {
    args.cancel = true;
    alert("Enter the username and password");
  }
  else if (userName.value === "") {
    args.cancel = true;
    alert("Enter the username");
  }
  else if (password.value === "") {
    args.cancel = true;
    alert("Enter the password");
  }
  else if (userName.value.length < 4) {
    args.cancel = true;
    alert("Username must be minimum 4 characters");
  }
  else {
    args.cancel = false;
    userName.value = "";
    password.value = "";
  }
}
return (<div className="App" id='container'>
  <DialogComponent id="dlg-button" width='300px' isModal={true}
target='#container' header='Sign In' showCloseIcon={false}

```

```

closeOnEscape={false} beforeClose={validation} buttons={buttons} ref={dialog
=> dialogInstance = dialog}>
  <div>{dialogContent()}</div>
</DialogComponent>
</div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  let userName: HTMLInputElement;
  let password: HTMLInputElement;
  const buttons: any = [{
    buttonModel: {
      content: 'LOG IN',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  }];
  function dialogContent(): JSX.Element {
    return (
      <div className="login-form">
        <div className='wrap'>
          <div id="heading"/>
          <div className="e-float-input">
            <input id="textvalue" type="text" ref = {user => userName =
user!} required = {true}/>
            <span className="e-float-line"/>
            <label className="e-float-text">Username</label>
          </div>
          <div className="e-float-input">
            <input id="textvalue2" type="password" ref = {pwd =>
password = pwd!} required = {true}/>
            <span className="e-float-line"/>
            <label className="e-float-text">Password</label>
          </div>
        </div>
      </div>
    )
  }
  function validation(args: any): void {
    if (userName.value === "" && password.value === "") {
      args.cancel= true;
      alert("Enter the username and password")
    } else if (userName.value === "") {
      args.cancel= true;
      alert("Enter the username")
    } else if (password.value === "") {
      args.cancel= true;
    }
  }
}

```

```

        alert("Enter the password")
    } else if (userName.value.length < 4) {
        args.cancel= true;
        alert("Username must be minimum 4 characters")
    } else {
        args.cancel= false;
        userName.value = "";
        password.value = "";
    }
}
return (
    <div className="App" id='container'>
        <DialogComponent id="dlg-button" width='300px' isModal={true}
target='#container' header='Sign In' showCloseIcon={false} closeOnEscape =
{false}
        beforeClose={validation} buttons={buttons} ref={dialog =>
dialogInstance = dialog!}>
            <div>{dialogContent()}</div>
        </DialogComponent>
    </div>
);
}
export default App;

```

Prevent the focus on the first element in React Dialog component

By default, the dialog focuses on the first elements of the content area which can be active and focusable. You can prevent this default focusing behavior using the [open](#) event and by enabling the `preventFocus` argument.

Bind the open event and enable the `preventFocus` argument within an event like the below sample.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    dialogInstance;
    buttons = [{
        buttonModel: {
            content: 'Ok',
            isPrimary: true,
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }, {
        buttonModel: {
            content: 'Cancel'
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }
    ]};
    dialogContent() {

```

```

    return (<div className='form-group'><label
htmlFor='email'>Email:</label>
      <input type='email' className='form-control' id='email' />
    </div>
    ,
      <div className='form-group'>
        <label htmlFor='comment'>Password:</label>
        <input type='password' className='form-control' id='password' />
      </div>);
  }
  onOpen = (args) => {
    args.preventDefault = true;
  };
  render() {
    return (<div className="App" id='container'>
      <DialogComponent id="dlg-focus" width='300px' isModal={true}
target='#container' header='Sign In' open={this.onOpen}
buttons={this.buttons} ref={dialog => this.dialogInstance = dialog}>
        <div>{this.dialogContent()}</div>
      </DialogComponent>
    </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogObj: DialogComponent;
  private buttonRef;
  private buttonElement: HTMLElement | null;
  constructor(props: {}) {
    super(props);
    this.buttonElement = null;
    this.buttonRef = element => {
      this.buttonElement = element;
    };
  }
  public buttons: any = [{
    buttonModel: {
      content: 'Ok',
      isPrimary: true,
    },
    'click': () => {
      this.dialogObj.hide();
    }
  }, {
    buttonModel: {
      content: 'Cancel'
    },
    'click': () => {
      this.dialogObj.hide();
    }
  }
]
}

```

```

    }];

    public onOpen = (args: any): void =>{
        args.preventDefault = true;
    }
    public onOpenDialog = function(): void {
        // Call the show method to open the Dialog
        this.dialogObj.show();
        this.buttonElement.style.display='none';
    };
    public onClose= function():void{
        this.buttonElement.style.display='block';
    }

    public render() {
        return (
            <div className="App" id='container'>
                <button className="e-control e-btn" style={{display: "none"}}
ref={this.buttonRef} onClick={this.onOpenDialog.bind(this)}>Open
dialog</button>
                <DialogComponent id="dialog" width='300px' isModal={true}
target='#container' header='Sign In' close={this.onClose} open={this.onOpen}
buttons={this.buttons} ref={dialog => this.dialogObj = dialog}>
                    <div className='form-group'><label htmlFor='email'>Email:</label>
                        <input type='email' className='form-control' id='email' />
                    </div>
                    <div className='form-group'>
                        <label htmlFor='comment'>Password:</label>
                        <input type='password' className='form-control'
id='password' />
                    </div>
                </DialogComponent>
            </div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    const buttons = [{
        buttonModel: {
            content: 'Ok',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    }, {
        buttonModel: {
            content: 'Cancel'

```



```

        },
        'click': () => {
            dialogInstance.hide();
        }
    }];
    function dialogContent() {
        return (<><div className='form-group'><label
htmlFor='email'>Email:</label>
        <input type='email' className='form-control' id='email' />
        </div><div className='form-group'>
            <label htmlFor='comment'>Password:</label>
            <input type='password' className='form-control'
id='password' />
        </div></>);
    }
    function onOpen(args) {
        args.preventDefault = true;
    }
    return (<div className="App" id='container'>
        <DialogComponent id="dlg-focus" width='300px' isModal={true}
target='#container' header='Sign In' open={onOpen} buttons={buttons}
ref={dialog => dialogInstance = dialog}>
            <div>{dialogContent()}</div>
        </DialogComponent>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogObj: DialogComponent;
    let buttonElement: HTMLElement;
    let buttonRef: React.Ref<HTMLButtonElement> = (element) => {
        buttonElement = element;
    };
    const buttons: any = [{
        buttonModel: {
            content: 'Ok',
            isPrimary: true,
        },
        'click': () => {
            dialogObj.hide();
        }
    }, {
        buttonModel: {
            content: 'Cancel'
        },
        'click': () => {
            dialogObj.hide();
        }
    }
    ];
    function onOpen (args: any): void {
        args.preventDefault = true;
    }
}

```

```

    }
    function onOpenDialog() {
        dialogObj.show();
        buttonElement.style.display = 'none';
    };
    function onClose() {
        buttonElement.style.display = 'block';
    }
    return (
        <div className="App" id='container'>
            <button className="e-control e-btn" style={{display: "none"}}
                ref={buttonRef} onClick={onOpenDialog.bind(this)}>Open dialog</button>
            <DialogComponent id="dialog" close={onClose} width='300px'
                isModal={true} target='#container' header='Sign In' open={onOpen}
                buttons={buttons} ref={dialog => dialogObj = dialog}>
                <div className='form-group'><label
                    htmlFor='email'>Email:</label>
                    <input type='email' className='form-control' id='email' />
                </div>
                <div className='form-group'>
                    <label htmlFor='comment'>Password:</label>
                    <input type='password' className='form-control' id='password'
                />
            </div>
        </DialogComponent>
    </div>
    );
}
export default App;

```

Prevent opening of the dialog in React Dialog component

You can prevent opening of the dialog by setting the [beforeOpen](#) event argument cancel value to true. In the following sample, the success dialog is opened when you enter the username value with minimum 4 characters. Otherwise, it will not be opened.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    userName;
    password;
    dialogInstance;
    buttons = [{
        buttonModel: {
            content: 'DISMISS',
            cssClass: 'e-primary',
            isPrimary: true,
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }];
    onSubmit() {

```

```

        this.dialogInstance.show();
    }
    validation = (args) => {
        if (this.userName.value === "" && this.password.value === "") {
            args.cancel = true;
            alert("Enter the username and password");
        }
        else if (this.userName.value === "") {
            args.cancel = true;
            alert("Enter the username");
        }
        else if (this.password.value === "") {
            args.cancel = true;
            alert("Enter the password");
        }
        else if (this.userName.value.length < 4) {
            args.cancel = true;
            alert("Username must be minimum 4 characters");
        }
        else {
            args.cancel = false;
            this.userName.value = "";
            this.password.value = "";
        }
    };
    onInputFocus = (args) => {
        if (!args.target.parentElement.classList.contains('e-input-in-
wrap')) {
            args.target.parentElement.classList.add('e-input-focus');
        }
        else {
            args.target.parentElement.parentElement.classList.add('e-input-
focus');
        }
    };
    onInputBlur = (args) => {
        if (!args.target.parentElement.classList.contains('e-input-in-
wrap')) {
            args.target.parentElement.classList.remove('e-input-focus');
        }
        else {
            args.target.parentElement.parentElement.classList.remove('e-
input-focus');
        }
    };
    render() {
        return (<div className='App' id='dialog-target'>
            <div className="login-form">
                <div className='wrap'>
                    <div id="heading">Sign in</div>
                    <div className="e-float-input">
                        <input id="textvalue" type="text" required={true}
ref={user => this.userName = user} onFocus={this.onInputFocus}
onBlur={this.onInputBlur}/>
                        <span className="e-float-line"/>
                        <label className="e-float-text">Username</label>
                    </div>

```

```

        <div className="e-float-input">
            <input id="textvalue2" type="password" required={true}
ref={pwd => this.password = pwd} onFocus={this.onInputFocus}
onBlur={this.onInputBlur}/>
            <span className="e-float-line"/>
            <label className="e-float-text">Password</label>
        </div>
        <div className="button-contain">
            <button className="e-control e-btn e-info"
id="targetButton" role="button" e-ripple="true" onClick={this.onSubmit =
this.onSubmit.bind(this)}>Log in</button>
        </div>
    </div>
    </div>
    <DialogComponent id='dialog' header='Success' buttons={this.buttons}
beforeOpen={this.validation} content='Congratulations! Login Success'
width='250px' isModal={true} ref={dialog => this.dialogInstance = dialog}
visible={false} target='#dialog-target' /></div>;
    }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public userName: HTMLInputElement;
    public password: HTMLInputElement;
    public dialogInstance: DialogComponent;
    private buttons: any = [{
        buttonModel: {
            content: 'DISMISS',
            cssClass: 'e-primary',
            isPrimary: true,
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }];
    public onSubmit(): void {
        this.dialogInstance.show();
    }
    public validation = (args: any): void => {
        if (this.userName.value === "" && this.password.value === "") {
            args.cancel= true;
            alert("Enter the username and password")
        } else if (this.userName.value === "") {
            args.cancel= true;
            alert("Enter the username")
        } else if (this.userName.value === "") {
            args.cancel= true;
            alert("Enter the password")
        } else if (this.userName.value.length < 4) {
            args.cancel= true;
            alert("Username must be minimum 4 characters")
        }
    }
}

```

```

    } else {
      args.cancel= false;
      this.userName.value = "";
      this.password.value = "";
    }
  }
  public onInputFocus = (args: any) => {
    if (!args.target.parentElement.classList.contains('e-input-in-wrap')) {
      args.target.parentElement.classList.add('e-input-focus');
    } else {
      args.target.parentElement.parentElement.classList.add('e-input-focus')
    }
  }
  public onInputBlur = (args: any) => {
    if (!args.target.parentElement.classList.contains('e-input-in-wrap')) {
      args.target.parentElement.classList.remove('e-input-focus');
    } else {
      args.target.parentElement.parentElement.classList.remove('e-input-focus');
    }
  }
  public render() {
    return (
      <div className='App' id='dialog-target'>
        <div className="login-form" >
          <div className='wrap'>
            <div id="heading">Sign in</div>
            <div className="e-float-input">
              <input id="textvalue" type="text" required = {true} ref
= {user => this.userName = user!} onFocus = {this.onInputFocus} onBlur =
{this.onInputBlur}/>
              <span className="e-float-line"/>
              <label className="e-float-text">Username</label>
            </div>
            <div className="e-float-input">
              <input id="textvalue2" type="password" required = {true}
ref = {pwd => this.password = pwd!} onFocus = {this.onInputFocus} onBlur =
{this.onInputBlur}/>
              <span className="e-float-line"/>
              <label className="e-float-text">Password</label>
            </div>
            <div className="button-contain">
              <button className="e-control e-btn e-info"
id="targetButton" role="button" e-ripple="true" onClick = {this.onSubmit =
this.onSubmit.bind(this)}>Log in</button>
            </div>
          </div>
          <DialogComponent id='dialog' header='Success' buttons={this.buttons}
beforeOpen = {this.validation} content='Congratulations! Login Success'
width='250px' isModal={true} ref={dialog => this.dialogInstance = dialog!}
visible={false}
target='#dialog-target' /></div>
        )
      )
    )
  }
}

```

```
export default App;
```

[Functional-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let userName;
    let password;
    let dialogInstance;
    const buttons = [{
        buttonModel: {
            content: 'DISMISS',
            cssClass: 'e-primary',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    }];
    function onSubmit() {
        dialogInstance.show();
    }
    function validation(args) {
        if (userName.value === "" && password.value === "") {
            args.cancel = true;
            alert("Enter the username and password");
        }
        else if (userName.value === "") {
            args.cancel = true;
            alert("Enter the username");
        }
        else if (password.value === "") {
            args.cancel = true;
            alert("Enter the password");
        }
        else if (userName.value.length < 4) {
            args.cancel = true;
            alert("Username must be minimum 4 characters");
        }
        else {
            args.cancel = false;
            userName.value = "";
            password.value = "";
        }
    }
    function onInputFocus(args) {
        if (!args.target.parentElement.classList.contains('e-input-in-
wrap')) {
            args.target.parentElement.classList.add('e-input-focus');
        }
        else {
            args.target.parentElement.parentElement.classList.add('e-input-
focus');
        }
    }
}
```

```

    }
  }
  function onInputBlur(args) {
    if (!args.target.parentElement.classList.contains('e-input-in-
wrap')) {
      args.target.parentElement.classList.remove('e-input-focus');
    }
    else {
      args.target.parentElement.parentElement.classList.remove('e-
input-focus');
    }
  }
  return (<div className='App' id='dialog-target'>
    <div className="login-form">
      <div className='wrap'>
        <div id="heading">Sign in</div>
        <div className="e-float-input">
          <input id="textvalue" type="text" required={true}
ref={user => userName = user} onFocus={onInputFocus} onBlur={onInputBlur}/>
          <span className="e-float-line"/>
          <label className="e-float-text">Username</label>
        </div>
        <div className="e-float-input">
          <input id="textvalue2" type="password" required={true}
ref={pwd => password = pwd} onFocus={onInputFocus} onBlur={onInputBlur}/>
          <span className="e-float-line"/>
          <label className="e-float-text">Password</label>
        </div>
        <div className="button-contain">
          <button className="e-control e-btn e-info"
id="targetButton" role="button" e-ripple="true"
onClick={onSubmit.bind(this)}>Log in</button>
        </div>
      </div>
    </div>
    <DialogComponent id='dialog' header='Success' buttons={buttons}
beforeOpen={validation} content='Congratulations! Login Success'
width='250px' isModal={true} ref={dialog => dialogInstance = dialog}
visible={false} target='#dialog-target' /></div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let userName: HTMLInputElement;
  let password: HTMLInputElement;
  let dialogInstance: DialogComponent;
  const buttons: any = [{
    buttonModel: {
      content: 'DISMISS',
      cssClass: 'e-primary',
      isPrimary: true,
    },
  ],

```

```

        'click': () => {
            dialogInstance.hide();
        }
    }];
function onSubmit(): void {
    dialogInstance.show();
}
function validation (args: any): void {
    if (userName.value === "" && password.value === "") {
        args.cancel= true;
        alert("Enter the username and password")
    } else if (userName.value === "") {
        args.cancel= true;
        alert("Enter the username")
    } else if (password.value === "") {
        args.cancel= true;
        alert("Enter the password")
    } else if (userName.value.length < 4) {
        args.cancel= true;
        alert("Username must be minimum 4 characters")
    } else {
        args.cancel= false;
        userName.value = "";
        password.value = "";
    }
}
function onInputFocus(args: any) {
    if (!args.target.parentElement.classList.contains('e-input-in-wrap')) {
        args.target.parentElement.classList.add('e-input-focus');
    } else {
        args.target.parentElement.parentElement.classList.add('e-input-
focus')
    }
}
function onInputBlur (args: any) {
    if (!args.target.parentElement.classList.contains('e-input-in-wrap')) {
        args.target.parentElement.classList.remove('e-input-focus');
    } else {
        args.target.parentElement.parentElement.classList.remove('e-input-
focus');
    }
}
return (
    <div className='App' id='dialog-target'>
        <div className="login-form" >
            <div className='wrap'>
                <div id="heading">Sign in</div>
                <div className="e-float-input">
                    <input id="textvalue" type="text" required = {true} ref
= {user => userName = user!} onFocus = {onInputFocus} onBlur =
{onInputBlur}/>
                    <span className="e-float-line"/>
                    <label className="e-float-text">Username</label>
                </div>
                <div className="e-float-input">

```



```

        <input id="textvalue2" type="password" required = {true}
    ref = {pwd => password = pwd!} onFocus = {onInputFocus} onBlur =
    {onInputBlur}/>
        <span className="e-float-line"/>
        <label className="e-float-text">Password</label>
    </div>
    <div className="button-contain">
        <button className="e-control e-btn e-info"
    id="targetButton" role="button" e-ripple="true" onClick =
    {onSubmit.bind(this)}>Log in</button>
    </div>
    </div>
    </div>
    <DialogComponent id='dialog' header='Success' buttons={buttons}
    beforeOpen = {validation} content='Congratulations! Login Success'
    width='250px' isModal={true} ref={dialog => dialogInstance = dialog!}
    visible={false}
    target='#dialog-target' /></div>
    );
    }
    export default App;

```

Read all the values from dialog on button click in React Dialog component

You can read the dialog element values by binding the action handler to the footer buttons. The buttons property provides the options to bind events to the action buttons. For detailed information about buttons, refer to the [footer](#) section.

In the below sample, value of input elements within the dialog has been checked in the footer button click event and send the values as the content of confirmation dialog.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    dialogInstance;
    innerDialogInstance;
    nameInput;
    emailInput;
    contactInput;
    addressInput;
    buttons = [{
        buttonModel: {
            content: 'Submit',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            this.dlgbuttonClick();
        }
    }];
    handleClick() {
        this.dialogInstance.show();
    }
}

```

```

    nestedbuttonClick() {
        this.innerDialogInstance.hide();
        this.dialogInstance.show();
    }
    footerbuttonclick() {
        this.innerDialogInstance.hide();
    }
    dialogContent() {
        return (<form>
            <div className='form-group'>
                <label>Name:</label>
                <input type='name' className='form-control' id='name' ref={n =>
this.nameInput = n}/>
            </div>
            <div className='form-group'>
                <label>Email Id:</label>
                <input type='email' placeholder='user@syncfusion.com'
className='form-control' id='email' ref={e => this.emailInput = e}/>
            </div>
            <div className='form-group'>
                <label>Contact Number:</label>
                <input type='contact' className='form-control' id='contact'
ref={c => this.contactInput = c}/>
            </div>
            <div className='form-group'><label>Address:</label>
                <textarea className='form-control' id='address' ref={a =>
this.addressInput = a}/>
            </div>
        </form>);
    }
    dlgbuttonClick() {
        this.dialogInstance.hide();
        this.innerDialogInstance.content = this.getDynamicContent();
        this.innerDialogInstance.buttons = [{ click:
this.footerbuttonclick.bind(this), buttonModel: { content: 'Yes', isPrimary:
true } },
            { click: this.nestedbuttonClick.bind(this), buttonModel: {
content: 'No', isPrimary: true } }];
        this.innerDialogInstance.show();
    }
    getDynamicContent() {
        const template = "<div class='row'><div class='col-xs-6 col-sm-6
col-lg-6 col-md-6'><b>Confirm your details</b></div>" +
            "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'><span id='name'> Name: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='nameValue'>" + this.nameInput.value + "</span> </div></div>" +
            "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-
6'><span id='email'> Email: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='emailValue'>" + this.emailInput.value +
            "</span></div></div><div class='row'><div class='col-xs-6 col-
sm-6 col-lg-6 col-md-6'>" +
            "<span id='Contact'> Contact number: </span></div><div
class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>" +

```

```

        "<span id='contactValue'>" + this.contactInput.value + "
    </span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
    col-md-6'>" +
        "<span id='Address'> Address: </span> </div><div class='col-xs-6
    col-sm-6 col-lg-6 col-md-6'><span id='AddressValue'>" +
    this.addressInput.value + "</span></div></div>";
    return template;
}
render() {
    return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
    onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' width='400px' target='#dialog-target'
    header='Dialog' visible={false} closeOnEscape={false} showCloseIcon={true}
    buttons={this.buttons} ref={dialog => this.dialogInstance = dialog}>
        <div>{this.dialogContent()}</div>
    </DialogComponent>
        <DialogComponent id='innerDialog' header='User details' isModal={true}
    showCloseIcon={true} width='400px' visible={false} ref={dialog =>
    this.innerDialogInstance = dialog} closeOnEscape={false} target='#dialog-
    target'>/>
    </div>);
}
}
export default App;

```

APP.TSX

```

{% raw %}
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public dialogInstance: DialogComponent;
    public innerDialogInstance: DialogComponent;
    public nameInput: HTMLInputElement;
    public emailInput: HTMLInputElement;
    public contactInput: HTMLInputElement;
    public addressInput: HTMLTextAreaElement;
    public buttons: any = [{
        buttonModel: {
            content: 'Submit',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            this.dlgbuttonClick();
        }
    }];
    public handleClick() {
        this.dialogInstance.show();
    }
    public nestedbuttonClick() {
        this.innerDialogInstance.hide();
        this.dialogInstance.show();
    }
    public footerbuttonclick() {

```

```

        this.innerDialogInstance.hide();
    }
    public dialogContent(): JSX.Element {
        return (
            <form>
                <div className='form-group'>
                    <label>Name:</label>
                    <input type='name' className='form-control' id='name' ref={n =>
this.nameInput = n!}/>
                </div>
                <div className='form-group'>
                    <label>Email Id:</label>
                    <input type='email' placeholder='user@syncfusion.com'
className='form-control' id='email' ref={e => this.emailInput = e!}/>
                </div>
                <div className='form-group'>
                    <label>Contact Number:</label>
                    <input type='contact' className='form-control' id='contact'
ref={c => this.contactInput = c!} />
                </div>
                <div className='form-group'><label>Address:</label>
                    <textarea className='form-control' id='address' ref={a =>
this.addressInput = a!}/>
                </div>
            </form>
        )
    }
    public dlgbuttonClick() {
        this.dialogInstance.hide();
        this.innerDialogInstance.content = this.getDynamicContent();
        this.innerDialogInstance.buttons = [{click:
this.footerbuttonclick.bind(this), buttonModel: { content: 'Yes', isPrimary:
true }},
        {click: this.nestedbuttonClick.bind(this), buttonModel: { content: 'No',
isPrimary: true } }];
        this.innerDialogInstance.show();
    }
    public getDynamicContent() {
        const template: string = "<div class='row'><div class='col-xs-6 col-sm-6
col-lg-6 col-md-6'><b>Confirm your details</b></div>" +
            "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-
6'><span id='name'> Name: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='nameValue'>"+ this.nameInput.value + "</span> </div></div>" +
            "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='email'> Email: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='emailValue'>"+ this.emailInput.value +
            "</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-
lg-6 col-md-6'>"+
            "<span id='Contact'> Contact number: </span></div><div class='col-xs-6
col-sm-6 col-lg-6 col-md-6'>"+
            "<span id='contactValue'>"+ this.contactInput.value +
            "</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'>"+

```

```

    "<span id='Address'> Address: </span> </div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span id='AddressValue'>" + this.addressInput.value
    + "</span></div></div>"
    return template;
}
public render() {
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
            onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
            <DialogComponent id='dialog' width='400px' target='#dialog-target'
            header='Dialog' visible={false} closeOnEscape={false}
            showCloseIcon={true} buttons={this.buttons} ref={dialog =>
            this.dialogInstance = dialog!}>
                <div>{this.dialogContent()}</div>
            </DialogComponent>
            <DialogComponent id='innerDialog'
                header='User details' isModal={true} showCloseIcon={true}
                width='400px' visible={false}
                ref={dialog => this.innerDialogInstance = dialog!}
                closeOnEscape={false} target='#dialog-target' />
        </div>);
    }
}
export default App;
{% enddraw %}

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    let innerDialogInstance;
    let nameInput;
    let emailInput;
    let contactInput;
    let addressInput;
    const buttons = [{
        buttonModel: {
            content: 'Submit',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dlgButtonClick();
        }
    }];
    function handleClick() {
        dialogInstance.show();
    }
    function nestedButtonClick() {
        innerDialogInstance.hide();
        dialogInstance.show();
    }
}

```

```

    }
    function footerbuttonclick() {
        innerDialogInstance.hide();
    }
    function dialogContent() {
        return (<form>
            <div className='form-group'>
                <label>Name:</label>
                <input type='name' className='form-control' id='name' ref={n =>
nameInput = n}/>
            </div>
            <div className='form-group'>
                <label>Email Id:</label>
                <input type='email' placeholder='user@syncfusion.com'
className='form-control' id='email' ref={e => emailInput = e}/>
            </div>
            <div className='form-group'>
                <label>Contact Number:</label>
                <input type='contact' className='form-control' id='contact'
ref={c => contactInput = c}/>
            </div>
            <div className='form-group'><label>Address:</label>
                <textarea className='form-control' id='address' ref={a =>
addressInput = a}/>
            </div>
        </form>);
    }
    function dlgbuttonClick() {
        dialogInstance.hide();
        innerDialogInstance.content = getDynamicContent();
        innerDialogInstance.buttons = [{ click:
footerbuttonclick.bind(this), buttonModel: { content: 'Yes', isPrimary: true
} },
            { click: nestedbuttonClick.bind(this), buttonModel: { content:
'No', isPrimary: true } }];
        innerDialogInstance.show();
    }
    function getDynamicContent() {
        const template = "<div class='row'><div class='col-xs-6 col-sm-6
col-lg-6 col-md-6'><b>Confirm your details</b></div>" +
            "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'><span id='name'> Name: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='nameValue'>" + nameInput.value + "</span> </div></div>" +
            "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-
6'><span id='email'> Email: </span>" +
            "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='emailValue'>" + emailInput.value +
            "</span></div></div><div class='row'><div class='col-xs-6 col-
sm-6 col-lg-6 col-md-6'>" +
            "<span id='Contact'> Contact number: </span></div><div
class='col-xs-6 col-sm-6 col-lg-6 col-md-6'>" +
            "<span id='contactValue'>" + contactInput.value + "
</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'>" +

```

```

        "<span id='Address'> Address: </span> </div><div class='col-xs-6
col-sm-6 col-lg-6 col-md-6'><span id='AddressValue'>" + addressInput.value +
"</span></div></div>";
        return template;
    }
    return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' width='400px' target='#dialog-target'
header='Dialog' visible={false} closeOnEscape={false} showCloseIcon={true}
buttons={buttons} ref={dialog => dialogInstance = dialog}>
        <div>{dialogContent()}</div>
        </DialogComponent>
        <DialogComponent id='innerDialog' header='User details'
isModal={true} showCloseIcon={true} width='400px' visible={false}
ref={dialog => innerDialogInstance = dialog} closeOnEscape={false}
target='#dialog-target' />
    </div>);
}
export default App;

```

APP.TSX

```

{% raw %}
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    let innerDialogInstance: DialogComponent;
    let nameInput: HTMLInputElement;
    let emailInput: HTMLInputElement;
    let contactInput: HTMLInputElement;
    let addressInput: HTMLTextAreaElement;
    const buttons: any = [{
        buttonModel: {
            content: 'Submit',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dlgbuttonClick();
        }
    }];
    function handleClick() {
        dialogInstance.show();
    }
    function nestedbuttonClick() {
        innerDialogInstance.hide();
        dialogInstance.show();
    }
    function footerbuttonclick() {
        innerDialogInstance.hide();
    }
    function dialogContent(): JSX.Element {
        return (
            <form>

```

```

        <div className='form-group'>
            <label>Name:</label>
            <input type='name' className='form-control' id='name' ref={n =>
nameInput = n!}/>
        </div>
        <div className='form-group'>
            <label>Email Id:</label>
            <input type='email' placeholder='user@syncfusion.com'
className='form-control' id='email' ref={e => emailInput = e!}/>
        </div>
        <div className='form-group'>
            <label>Contact Number:</label>
            <input type='contact' className='form-control' id='contact'
ref={c => contactInput = c!} />
        </div>
        <div className='form-group'><label>Address:</label>
            <textarea className='form-control' id='address' ref={a =>
addressInput = a!}/>
        </div>
    </form>
)
}
function dlgbuttonClick() {
    dialogInstance.hide();
    innerDialogInstance.content = getDynamicContent();
    innerDialogInstance.buttons = [{click: footerbuttonclick.bind(this),
buttonModel: { content: 'Yes', isPrimary: true }},
    {click: nestedbuttonClick.bind(this), buttonModel: { content: 'No',
isPrimary: true }}];
    innerDialogInstance.show();
}
function getDynamicContent() {
    const template: string = "<div class='row'><div class='col-xs-6 col-sm-6
col-lg-6 col-md-6'><b>Confirm your details</b></div>" +
        "</div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-
6'><span id='name'> Name: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='nameValue'>"+ nameInput.value + "</span> </div></div>" +
        "<div class='row'><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='email'> Email: </span>" +
        "</div><div class='col-xs-6 col-sm-6 col-lg-6 col-md-6'><span
id='emailValue'>"+ emailInput.value +
        "</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-
lg-6 col-md-6'>"+
        "<span id='Contact'> Contact number: </span></div><div class='col-xs-6
col-sm-6 col-lg-6 col-md-6'>"+
        "<span id='contactValue'>"+ contactInput.value + "
</span></div></div><div class='row'><div class='col-xs-6 col-sm-6 col-lg-6
col-md-6'>"+
        "<span id='Address'> Address: </span> </div><div class='col-xs-6 col-sm-
6 col-lg-6 col-md-6'><span id='AddressValue'>"+ addressInput.value
+ "</span></div></div>"
    return template;
}
return (
    <div className="App" id='dialog-target'>

```



```

    <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent id='dialog' width='400px' target='#dialog-target'
header='Dialog' visible={false} closeOnEscape={false}
    showCloseIcon={true} buttons={buttons} ref={dialog =>
dialogInstance = dialog!}>
    <div>{dialogContent()}</div>
</DialogComponent>
    <DialogComponent id='innerDialog'
    header='User details' isModal={true} showCloseIcon={true}
width='400px' visible={false}
    ref={dialog => innerDialogInstance = dialog!}
closeOnEscape={false} target='#dialog-target' />
</div>
);
}
export default App;
{% endraw %}

```

Customize the dialog appearance in React Dialog component

You can customize the dialog appearance by providing dialog template as string or HTML element to the [content](#) property. In the following sample, dialog is customized as error window appearance.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  settings = { effect: 'Zoom', duration: 400, delay: 0 };
  dialogInstance;
  buttons = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];
  handleClick() {
    this.dialogInstance.show();
  }
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent id="dlg-button" header='File and Folder Rename'
showCloseIcon={true} width='400px' buttons={this.buttons}
animationSettings={this.settings} closeOnEscape={true} ref={dialog =>
this.dialogInstance = dialog} target='#dialog-target'>
        <div>
          <div className='dialog-content'>
            <div className='msg-wrapper col-lg-12'>

```

```

        <span className='e-icons close-icon col-lg-2' />
        <span className='error-msg col-lg-10'>Can not rename
'pictures' because a file or folder with that name already exists </span>
      </div>
      <div className='error-detail col-lg-8'>
        <span>Specify a different name</span>
      </div>
    </div>
  </div>
</DialogComponent>
</div>);
}
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
  public dialogInstance: DialogComponent;
  public buttons: any = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];
  public handleClick() {
    this.dialogInstance.show();
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent id="dlg-button"
          header='File and Folder Rename' showCloseIcon={true}
          width='400px' buttons={this.buttons}
animationSettings={this.settings}
          closeOnEscape={true} ref={dialog => this.dialogInstance = dialog!}
          target='#dialog-target'>
          <div>
            <div className = 'dialog-content'>
              <div className='msg-wrapper col-lg-12'>
                <span className='e-icons close-icon col-lg-2' />
                <span className='error-msg col-lg-10'>Can not rename
'pictures' because a file or folder with that name already exists </span>
              </div>
              <div className='error-detail col-lg-8'>
                <span>Specify a different name</span>
              </div>
            </div>
          </div>
        </DialogComponent>
      </div>
    );
  }
}

```

```

        </div>
      </div>
    </DialogComponent>
  </div>);
}
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let settings = { effect: 'Zoom', duration: 400, delay: 0 };
  let dialogInstance;
  const buttons = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  }];
  function handleClick() {
    dialogInstance.show();
  }
  return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1' role='button'
    onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent id="dlg-button" header='File and Folder Rename'
    showCloseIcon={true} width='400px' buttons={buttons}
    animationSettings={settings} closeOnEscape={true} ref={dialog =>
    dialogInstance = dialog} target='#dialog-target'>
      <div>
        <div className='dialog-content'>
          <div className='msg-wrapper col-lg-12'>
            <span className='e-icons close-icon col-lg-2' />
            <span className='error-msg col-lg-10'>Can not rename
            'pictures' because a file or folder with that name already exists </span>
          </div>
          <div className='error-detail col-lg-8'>
            <span>Specify a different name</span>
          </div>
        </div>
      </div>
    </DialogComponent>
  </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
  let dialogInstance: DialogComponent;
  const buttons: any = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  }];
  function handleClick() {
    dialogInstance.show();
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent id="dlg-button"
        header='File and Folder Rename' showCloseIcon={true}
        width='400px' buttons={buttons} animationSettings={settings}
        closeOnEscape={true} ref={dialog => dialogInstance = dialog!}
        target='#dialog-target'>
        <div>
          <div className = 'dialog-content'>
            <div className='msg-wrapper col-lg-12'>
              <span className='e-icons close-icon col-lg-2' />
              <span className='error-msg col-lg-10'>Can not rename
'pictures' because a file or folder with that name already exists </span>
            </div>
            <div className='error-detail col-lg-8'>
              <span>Specify a different name</span>
            </div>
          </div>
        </div>
      </DialogComponent>
    </div>);
}
export default App;

```

Close dialog while click on outside of dialog in React Dialog component

By default, dialog can be closed by pressing Esc key and clicking the close icon on the right of dialog header. It can also be closed by clicking outside of the dialog using hide method.

Set the [CloseOnEscape](#) property value to false to prevent closing of the dialog when pressing Esc key.

In the following sample, dialog is closed when clicking outside the dialog area using [hide](#) method.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';

```

```

import * as React from "react";
class App extends React.Component {
  settings = { effect: 'Zoom', duration: 400, delay: 0 };
  dialogInstance;
  buttons = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];
  handleClick() {
    this.dialogInstance.show();
  }
  componentDidMount() {
    document.onclick = (args) => {
      if (args.target.id === 'dialog-target') {
        this.dialogInstance.hide();
      }
    };
  }
  render() {
    return (
      <div className="App" id="dialog-target">
        <button className="e-control e-btn" id="targetButton1" role="button"
          onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent header="Delete Multiple Items" showCloseIcon={true}
          visible={true} animationSettings={this.settings} width="300px"
          buttons={this.buttons} closeOnEscape={true} content="Are you sure you want
          to permanently delete all of these items?" ref={dialog =>
            this.dialogInstance = dialog} target="#dialog-target"/>
      </div>);
  }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
  public dialogInstance: DialogComponent;
  public buttons: any = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }];
}

```

```

public handleClick() {
    this.dialogInstance.show();
}

public componentDidMount() {
    document.onclick = (args: any) : void => {
        if(args.target.id === 'dialog-target') {
            this.dialogInstance.hide();
        }
    }
}

public render() {
    return (
        <div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
            <DialogComponent header='Delete Multiple Items' showCloseIcon={true}
visible={true} animationSettings={this.settings}
width='300px' buttons={this.buttons} closeOnEscape={true}
content='Are you sure you want to permanently delete all of these
items?' ref={dialog => this.dialogInstance = dialog!}
target='#dialog-target' />
        </div>);
    }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    const settings = { effect: 'Zoom', duration: 400, delay: 0 };
    let dialogInstance;
    const buttons = [{
        buttonModel: {
            content: 'Close',
            cssClass: 'e-flat',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    }];
    function handleClick() {
        dialogInstance.show();
    }
    function componentDidMount() {
        document.onclick = (args) => {
            if (args.target.id === 'dialog-target') {
                dialogInstance.hide();
            }
        }
    };
};

```

```

    }
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent header='Delete Multiple Items' showCloseIcon={true}
visible={true} animationSettings={settings} width='300px' buttons={buttons}
closeOnEscape={true} content='Are you sure you want to permanently delete
all of these items?' ref={dialog => dialogInstance = dialog}
target='#dialog-target' />
    </div>);
  }
  export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  const settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
  let dialogInstance: DialogComponent;
  const buttons: any = [{
    buttonModel: {
      content: 'Close',
      cssClass: 'e-flat',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  }];
  function handleClick() {
    dialogInstance.show();
  }
  function componentDidMount() {
    document.onclick = (args: any) : void => {
      if(args.target.id === 'dialog-target') {
        dialogInstance.hide();
      }
    }
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent header='Delete Multiple Items' showCloseIcon={true}
visible={true} animationSettings={settings}
width='300px' buttons={buttons} closeOnEscape={true}
content='Are you sure you want to permanently delete all of these
items?' ref={dialog => dialogInstance = dialog!}
target='#dialog-target' />
    </div>);
  }
  export default App;

```

Add an icons to dialog buttons in React Dialog component

You can add icons to the dialog buttons using the [buttons](#) property or [footerTemplate](#) property . For detailed information about dialog buttons, refer to the [documentation](#) section.

In the following sample, dialog footer buttons are customized with icons using [buttons](#) property.

[Class-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  settings = { effect: 'Zoom', duration: 400, delay: 0 };
  dialogInstance;
  buttons = [{
    buttonModel: {
      content: 'Ok',
      iconCss: 'e-icons e-ok-icon',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'No',
      iconCss: 'e-icons e-close-icon',
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  }
  ];
  handleClick() {
    this.dialogInstance.show();
  }
  render() {
    return (<div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
        onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent id='dialog' header='Delete Multiple Items'
        animationSettings={this.settings} showCloseIcon={true} closeOnEscape={true}
        width='300px' buttons={this.buttons} content='Are you sure you want to
        permanently delete all of these items?' ref={dialog => this.dialogInstance =
        dialog} target='#dialog-target' />
    </div>);
  }
}
export default App;
```

APP.TSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
```



```

    public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    public dialogInstance: DialogComponent;
    public buttons: any = [{
      buttonModel: {
        content: 'Ok',
        iconCss: 'e-icons e-ok-icon',
        isPrimary: true,
      },
      'click': () => {
        this.dialogInstance.hide();
      }
    },
    {
      buttonModel: {
        content: 'No',
        iconCss: 'e-icons e-close-icon',
      },
      'click': () => {
        this.dialogInstance.hide();
      }
    }
  ];
  public handleClick() {
    this.dialogInstance.show();
  }
  public render() {
    return (
      <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
          onClick={this.handleClick} = this.handleClick.bind(this)>Open</button>
        <DialogComponent id='dialog' header='Delete Multiple Items'
          animationSettings={this.settings} showCloseIcon={true} closeOnEscape={true}
          width='300px' buttons={this.buttons} content='Are you sure you want to
          permanently delete all of these items?' ref={dialog => this.dialogInstance =
          dialog!}
          target='#dialog-target' />
      </div>);
  }
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let settings = { effect: 'Zoom', duration: 400, delay: 0 };
  let dialogInstance;
  const buttons = [{
    buttonModel: {
      content: 'Ok',
      iconCss: 'e-icons e-ok-icon',
      isPrimary: true,
    },
  },

```

```

        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'No',
            iconCss: 'e-icons e-close-icon',
        },
        'click': () => {
            dialogInstance.hide();
        }
    }
];
function handleClick() {
    dialogInstance.show();
}
return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent id='dialog' header='Delete Multiple Items'
animationSettings={settings} showCloseIcon={true} closeOnEscape={true}
width='300px' buttons={buttons} content='Are you sure you want to
permanently delete all of these items?' ref={dialog => dialogInstance =
dialog} target='#dialog-target' />
</div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    let dialogInstance: DialogComponent;
    const buttons: any = [{
        buttonModel: {
            content: 'Ok',
            iconCss: 'e-icons e-ok-icon',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'No',
            iconCss: 'e-icons e-close-icon',
        },
        'click': () => {
            dialogInstance.hide();
        }
    }
    ];
}

```

```

function handleClick() {
    dialogInstance.show();
}
return (
    <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' header='Delete Multiple Items'
animationSettings={settings} showCloseIcon={true} closeOnEscape={true}
width='300px' buttons={buttons} content='Are you sure you want to
permanently delete all of these items?' ref={dialog => dialogInstance =
dialog!}>
            target='#dialog-target' />
        </div>);
}
export default App;

```

In the following sample, dialog footer buttons are customized with icons using `footerTemplate` property.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    dialogInstance;
    settings = { effect: 'Zoom', duration: 400, delay: 0 };
    footer() {
        return (<div>
            <button id='Button1' className="e-control e-btn e-primary e-
flat" data-ripple="true">
                <span className="e-btn-icon e-icons e-ok-icon e-icon-
left"/>Yes</button>
            <button id="Button2" className="e-control e-btn e-flat" data-
ripple="true">
                <span className="e-btn-icon e-icons e-close-icon e-icon-
left"/>No</button>
            </div>);
    }
    componentDidMount() {
        setTimeout(() => {
            document.getElementById('Button1').onclick = () => {
                this.dialogInstance.hide();
            };
            document.getElementById('Button2').onclick = () => {
                this.dialogInstance.hide();
            };
        });
    }
    handleClick() {
        this.dialogInstance.show();
    }
    render() {

```

```

        return (<div className="App" id='container'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
            <DialogComponent id='dialog' header='Delete Multiple Items'
animationSettings={this.settings} showCloseIcon={true} closeOnEscape={true}
width='300px' footerTemplate={this.footer} content='Are you sure you want to
permanently delete all of these items?' ref={dialog => this.dialogInstance =
dialog} target='#container'>/>
        </div>);
    }
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public dialogInstance: DialogComponent;
    public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    public footer(): JSX.Element {
        return (
            <div>
                <button id='Button1' className="e-control e-btn e-primary e-
flat" data-ripple="true">
                    <span className="e-btn-icon e-icons e-ok-icon e-icon-
left"/>Yes</button>
                <button id="Button2" className="e-control e-btn e-flat" data-
ripple="true">
                    <span className="e-btn-icon e-icons e-close-icon e-icon-
left"/>No</button>
            </div>
        )
    }
    public componentDidMount() {
        setTimeout(() => {
            (document.getElementById('Button1') as any).onclick = () => {
                this.dialogInstance.hide();
            };
            (document.getElementById('Button2') as any).onclick = () => {
                this.dialogInstance.hide();
            };
        });
    }
    public handleClick() {
        this.dialogInstance.show();
    }
    public render() {
        return (
            <div className="App" id='container'>
                <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
                <DialogComponent id='dialog'
                    header='Delete Multiple Items' animationSettings={this.settings}
showCloseIcon={true} closeOnEscape={true}
width='300px' footerTemplate={this.footer}

```

```

        content='Are you sure you want to permanently delete all of
these items?' ref={dialog => this.dialogInstance = dialog!}
        target='#container'/'>
    </div>);
}
}
export default App;

```

[Functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    let settings = { effect: 'Zoom', duration: 400, delay: 0 };
    function footer() {
        return (<div>
            <button id='Button1' className="e-control e-btn e-primary e-
flat" data-ripple="true">
                <span className="e-btn-icon e-icons e-ok-icon e-icon-
left"/>Yes</button>
                <button id="Button2" className="e-control e-btn e-flat"
data-ripple="true">
                <span className="e-btn-icon e-icons e-close-icon e-icon-
left"/>No</button>
            </div>);
    }
    function componentDidMount() {
        setTimeout(() => {
            document.getElementById('Button1').onclick = () => {
                dialogInstance.hide();
            };
            document.getElementById('Button2').onclick = () => {
                dialogInstance.hide();
            };
        });
    }
    function handleClick() {
        dialogInstance.show();
    }
    return (<div className="App" id='container'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' header='Delete Multiple Items'
animationSettings={settings} showCloseIcon={true} closeOnEscape={true}
width='300px' footerTemplate={footer} content='Are you sure you want to
permanently delete all of these items?' ref={dialog => dialogInstance =
dialog} target='#container'/'>
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance: DialogComponent;
    let settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    function footer(): JSX.Element {
        return (
            <div>
                <button id='Button1' className="e-control e-btn e-primary e-flat" data-ripple="true">
                    <span className="e-btn-icon e-icons e-ok-icon e-icon-left"/>Yes</button>
                <button id="Button2" className="e-control e-btn e-flat" data-ripple="true">
                    <span className="e-btn-icon e-icons e-close-icon e-icon-left"/>No</button>
            </div>
        )
    }
    function componentDidMount() {
        setTimeout(() => {
            (document.getElementById('Button1') as any).onclick = () => {
                dialogInstance.hide();
            };
            (document.getElementById('Button2') as any).onclick = () => {
                dialogInstance.hide();
            };
        });
    }
    function handleClick() {
        dialogInstance.show();
    }
    return (
        <div className="App" id='container'>
            <button className='e-control e-btn' id='targetButton1' role='button'
                onClick={handleClick.bind(this)}>Open</button>
            <DialogComponent id='dialog'
                header='Delete Multiple Items' animationSettings={settings}
                showCloseIcon={true} closeOnEscape={true}
                width='300px' footerTemplate={footer}
                content='Are you sure you want to permanently delete all of
                these items?' ref={dialog => dialogInstance = dialog!}
                target='#container' />
        </div>);
    }
    export default App;

```

Add a minimize maximize buttons in React Dialog component

React Dialog allows end users to either minimize or maximize the Dialog component. You can add minimize and maximize custom buttons near the close icon in the Dialog header using the [headerTemplate](#) property and handle the actions in the button click events, as shown in the following sample.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
  settings = { effect: 'Zoom', duration: 400, delay: 0 };
  dialogInstance;
  isFullScreen;
  dialogOldPositions;
  maxBtn;
  minBtn;
  header() {
    return (<div>
      <span className='title'>Dialog</span><span className='e-icons sf-
icon-Maximize' id='max-btn' title='Maximize'></span><span className='e-icons
sf-icon-Minimize' id='min-btn' title='Minimize'></span>
    </div>);
  }
  buttons = [{
    buttonModel: {
      content: 'Ok',
      iconCss: 'e-icons e-ok-icon',
      isPrimary: true,
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'No',
      iconCss: 'e-icons e-close-icon',
    },
    'click': () => {
      this.dialogInstance.hide();
    }
  },
  ];
  handleClick() {
    this.dialogInstance.show();
  }
  onCreate() {
    this.maxBtn = document.getElementById('max-btn');
    this.maxBtn.onclick = (e) => {
      var maximizeIcon;
      if (this.dialogInstance.element.classList.contains('dialog-
minimized')) {
        this.dialogInstance.element.querySelector('#min-
btn').classList.add('sf-icon-Minimize');
        this.dialogInstance.element.querySelector('#min-
btn').classList.remove('sf-icon-Restore');
        this.dialogInstance.element.querySelector('#min-
btn').setAttribute('title', 'Minimize');
      }
      if (!this.dialogInstance.element.classList.contains('dialog-
maximized') && !this.isFullScreen) {

```

```

        maximizeIcon =
this.dialogInstance.element.querySelector(".e-dlg-header-content .sf-icon-
Maximize");
    }
    else {
        maximizeIcon =
this.dialogInstance.element.querySelector(".e-dlg-header-content .sf-icon-
Restore");
    }
    if (!this.dialogInstance.element.classList.contains('dialog-
maximized')) {
        this.dialogInstance.element.classList.add('dialog-
maximized');
        this.dialogInstance.show(true);
        maximizeIcon.classList.add('sf-icon-Restore');
        maximizeIcon.setAttribute('title', 'Restore');
        maximizeIcon.classList.remove('sf-icon-Maximize');
        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        this.isFullScreen = true;
    }
    else {
        this.dialogInstance.element.classList.remove('dialog-
maximized');
        this.dialogInstance.show(false);
        maximizeIcon.classList.remove('sf-icon-Restore');
        maximizeIcon.classList.add('sf-icon-Maximize');
        maximizeIcon.setAttribute('title', 'Maximize');
        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        this.dialogInstance.position = this.dialogOldPositions;
        this.dialogInstance.dataBind();
        this.isFullScreen = false;
    }
};
this.minBtn = document.getElementById('min-btn');
this.minBtn.onclick = (e) => {
    var minimizeIcon =
this.dialogInstance.element.querySelector(".e-dlg-header-content .sf-icon-
Minimize");
    if (!this.dialogInstance.element.classList.contains('e-dlg-
fullscreen')) {
        if (!this.dialogInstance.element.classList.contains('dialog-
minimized')) {
            this.dialogOldPositions = { X:
this.dialogInstance.position.X, Y: this.dialogInstance.position.Y };
            this.dialogInstance.element.classList.add('dialog-
minimized');
            this.dialogInstance.element.classList.remove('dialog-
maximized');
            this.dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            this.dialogInstance.position = { X: 'center', Y:
'bottom' };
            this.dialogInstance.dataBind();
            minimizeIcon.classList.add('sf-icon-Restore');
            minimizeIcon.setAttribute('title', 'Restore');

```



```

        }
        else {
            this.dialogInstance.element.classList.remove('dialog-
minimized');
            this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
            minimizeIcon.classList.add('sf-icon-Minimize');
            minimizeIcon.setAttribute('title', 'Minimize');
            minimizeIcon.classList.remove('sf-icon-Restore');
            this.dialogInstance.position = this.dialogOldPositions;
            this.dialogInstance.dataBind();
        }
    }
    else {
        this.dialogInstance.show(false);
        this.dialogInstance.element.classList.remove('dialog-
maximized');
        this.dialogInstance.element.classList.add('dialog-
minimized');
        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
        minimizeIcon.classList.remove('sf-icon-Minimize');
        minimizeIcon.removeAttribute('title');
        this.dialogInstance.position = { X: 'center', Y: 'bottom' };
        this.dialogInstance.dataBind();
        this.isFullScreen = true;
    }
    };
}
render() {
    return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' created={this.onCreate.bind(this)}
header={this.header} animationSettings={this.settings} showCloseIcon={true}
closeOnEscape={true} width='300px' buttons={this.buttons} content='This is a
dialog with minimize and maximize buttons' ref={dialog =>
this.dialogInstance = dialog} target='#dialog-target' />
    </div>);
}
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
    public settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    public dialogInstance: DialogComponent;
    public isFullScreen: Boolean;
    public dialogOldPositions: any;
    public maxBtn: HTMLElement;
    public minBtn: HTMLElement;
    header() {
        return (<div>

```

```

        <span className='title'>Dialog</span><span className='e-icons sf-
icon-Maximize' id='max-btn' title='Maximize'></span><span className='e-icons
sf-icon-Minimize' id='min-btn' title='Minimize'></span>
    </div>);
    }
    public buttons: any = [{
        buttonModel: {
            content: 'Ok',
            iconCss: 'e-icons e-ok-icon',
            isPrimary: true,
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'No',
            iconCss: 'e-icons e-close-icon',
        },
        'click': () => {
            this.dialogInstance.hide();
        }
    }
    ];
    public handleClick() {
        this.dialogInstance.show();
    }
    public onCreate() {
        this.maxBtn = document.getElementById('max-btn') as HTMLElement;
        this.maxBtn.onclick = (e: Event) => {
            var maximizeIcon;
            if (this.dialogInstance.element.classList.contains('dialog-minimized'))
            {
                this.dialogInstance.element.querySelector('#min-
btn').classList.add('sf-icon-Minimize');
                this.dialogInstance.element.querySelector('#min-
btn').classList.remove('sf-icon-Restore');
                this.dialogInstance.element.querySelector('#min-
btn').setAttribute('title', 'Minimize');
            }
            if (!this.dialogInstance.element.classList.contains('dialog-maximized')
&& !this.isFullScreen) {
                maximizeIcon = this.dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Maximize");
            } else {
                maximizeIcon = this.dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Restore");
            }
            if (!this.dialogInstance.element.classList.contains('dialog-maximized'))
            {
                this.dialogInstance.element.classList.add('dialog-maximized');
                this.dialogInstance.show(true);
                maximizeIcon.classList.add('sf-icon-Restore');
                maximizeIcon.setAttribute('title', 'Restore');
                maximizeIcon.classList.remove('sf-icon-Maximize');
            }
        }
    }

```

```

        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        this.isFullScreen = true;
    } else {
        this.dialogInstance.element.classList.remove('dialog-maximized');
        this.dialogInstance.show(false);
        maximizeIcon.classList.remove('sf-icon-Restore');
        maximizeIcon.classList.add('sf-icon-Maximize');
        maximizeIcon.setAttribute('title', 'Maximize');
        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        this.dialogInstance.position = this.dialogOldPositions;
        this.dialogInstance.dataBind();
        this.isFullScreen = false;
    }
}

this.minBtn = document.getElementById('min-btn') as HTMLElement;
this.minBtn.onclick = (e: Event) => {
    var minimizeIcon = this.dialogInstance.element.querySelector(".e-
dlg-header-content .sf-icon-Minimize");
    if (!this.dialogInstance.element.classList.contains('e-dlg-
fullscreen')) {
        if (!this.dialogInstance.element.classList.contains('dialog-
minimized')) {
            this.dialogOldPositions = { X: this.dialogInstance.position.X,
Y: this.dialogInstance.position.Y }
            this.dialogInstance.element.classList.add('dialog-minimized');
            this.dialogInstance.element.classList.remove('dialog-
maximized');
            this.dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            this.dialogInstance.position = { X: 'center', Y: 'bottom' };
            this.dialogInstance.dataBind();
            minimizeIcon.classList.add('sf-icon-Restore');
            minimizeIcon.setAttribute('title', 'Restore');
        } else {
            this.dialogInstance.element.classList.remove('dialog-
minimized');
            this.dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
            minimizeIcon.classList.add('sf-icon-Minimize');
            minimizeIcon.setAttribute('title', 'Minimize');
            minimizeIcon.classList.remove('sf-icon-Restore');
            this.dialogInstance.position = this.dialogOldPositions;
            this.dialogInstance.dataBind();
        }
    } else {
        this.dialogInstance.show(false);
        this.dialogInstance.element.classList.remove('dialog-maximized');
        this.dialogInstance.element.classList.add('dialog-minimized');
        this.dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
        minimizeIcon.classList.remove('sf-icon-Minimize');
        minimizeIcon.removeAttribute('title');
        this.dialogInstance.position = { X: 'center', Y: 'bottom' };
        this.dialogInstance.dataBind();
        this.isFullScreen = true;
    }
}

```

```

    }
  }
}

public render() {
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
      <DialogComponent id='dialog' created={this.onCreate.bind(this)}
header={this.header} animationSettings={this.settings} showCloseIcon={true}
closeOnEscape={true}
        width='300px' buttons={this.buttons} content='This is a dialog with
minimize and maximize buttons' ref={dialog => this.dialogInstance = dialog!}
        target='#dialog-target' />
    </div>);
}
}
export default App;

```

[functional-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let settings = { effect: 'Zoom', duration: 400, delay: 0 };
  let dialogInstance;
  let isFullScreen;
  let dialogOldPositions;
  let maxBtn;
  let minBtn;
  function header() {
    return (<div>
      <span className='title'>Dialog</span><span className='e-icons sf-
icon-Maximize' id='max-btn' title='Maximize'></span><span class='e-icons sf-
icon-Minimize' id='min-btn' title='Minimize'></span>
    </div>);
  }
  let buttons = [{
    buttonModel: {
      content: 'Ok',
      iconCss: 'e-icons e-ok-icon',
      isPrimary: true,
    },
    'click': () => {
      dialogInstance.hide();
    }
  },
  {
    buttonModel: {
      content: 'No',
      iconCss: 'e-icons e-close-icon',
    },
    'click': () => {

```

```

        dialogInstance.hide();
    },
    ],
    function handleClick() {
        dialogInstance.show();
    }
    function onCreate() {
        maxBtn = document.getElementById('max-btn');
        maxBtn.onclick = (e) => {
            var maximizeIcon;
            if (dialogInstance.element.classList.contains('dialog-
minimized')) {
                dialogInstance.element.querySelector('#min-
btn').classList.add('sf-icon-Minimize');
                dialogInstance.element.querySelector('#min-
btn').classList.remove('sf-icon-Restore');
                dialogInstance.element.querySelector('#min-
btn').setAttribute('title', 'Minimize');
            }
            if (!dialogInstance.element.classList.contains('dialog-
maximized') && !isFullScreen) {
                maximizeIcon = dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Maximize");
            }
            else {
                maximizeIcon = dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Restore");
            }
            if (!dialogInstance.element.classList.contains('dialog-
maximized')) {
                dialogInstance.element.classList.add('dialog-maximized');
                dialogInstance.show(true);
                maximizeIcon.classList.add('sf-icon-Restore');
                maximizeIcon.setAttribute('title', 'Restore');
                maximizeIcon.classList.remove('sf-icon-Maximize');
                dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
                isFullScreen = true;
            }
            else {
                dialogInstance.element.classList.remove('dialog-maximized');
                dialogInstance.show(false);
                maximizeIcon.classList.remove('sf-icon-Restore');
                maximizeIcon.classList.add('sf-icon-Maximize');
                maximizeIcon.setAttribute('title', 'Maximize');
                dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
                dialogInstance.position = dialogOldPositions;
                dialogInstance.dataBind();
                isFullScreen = false;
            }
        };
        minBtn = document.getElementById('min-btn');
        minBtn.onclick = (e) => {
            var minimizeIcon = dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Minimize");

```

```

        if (!dialogInstance.element.classList.contains('e-dlg-
fullscreen')) {
            if (!dialogInstance.element.classList.contains('dialog-
minimized')) {
                dialogOldPositions = { X: dialogInstance.position.X, Y:
dialogInstance.position.Y };
                dialogInstance.element.classList.add('dialog-
minimized');
                dialogInstance.element.classList.remove('dialog-
maximized');
                dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
                dialogInstance.position = { X: 'center', Y: 'bottom' };
                dialogInstance.dataBind();
                minimizeIcon.classList.add('sf-icon-Restore');
                minimizeIcon.setAttribute('title', 'Restore');
            }
            else {
                dialogInstance.element.classList.remove('dialog-
minimized');
                dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
                minimizeIcon.classList.add('sf-icon-Minimize');
                minimizeIcon.setAttribute('title', 'Minimize');
                minimizeIcon.classList.remove('sf-icon-Restore');
                dialogInstance.position = dialogOldPositions;
                dialogInstance.dataBind();
            }
        }
        else {
            dialogInstance.show(false);
            dialogInstance.element.classList.remove('dialog-maximized');
            dialogInstance.element.classList.add('dialog-minimized');
            dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            minimizeIcon.classList.remove('sf-icon-Minimize');
            minimizeIcon.removeAttribute('title');
            dialogInstance.position = { X: 'center', Y: 'bottom' };
            dialogInstance.dataBind();
            isFullScreen = true;
        }
    };
}
return (<div className="App" id='dialog-target'>
    <button className='e-control e-btn' id='targetButton1'
role='button' onClick={handleClick.bind(this)}>Open</button>
    <DialogComponent id='dialog' created={onCreate.bind(this)}
header={header} animationSettings={settings} showCloseIcon={true}
closeOnEscape={true} width='300px' buttons={buttons} content='This is a
dialog with minimize and maximize buttons' ref={dialog => dialogInstance =
dialog} target='#dialog-target' />
    </div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let settings: any = { effect: 'Zoom', duration: 400, delay: 0 };
    let dialogInstance: DialogComponent;
    let isFullScreen: Boolean;
    let dialogOldPositions: any;
    let maxBtn: HTMLElement;
    let minBtn: HTMLElement;
    function header() {
        return (<div>
            <span className='title'>Dialog</span><span className='e-icons sf-
            icon-Maximize' id='max-btn' title='Maximize'></span><span class='e-icons sf-
            icon-Minimize' id='min-btn' title='Minimize'></span>
            </div>);
    }
    let buttons: any = [{
        buttonModel: {
            content: 'Ok',
            iconCss: 'e-icons e-ok-icon',
            isPrimary: true,
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    {
        buttonModel: {
            content: 'No',
            iconCss: 'e-icons e-close-icon',
        },
        'click': () => {
            dialogInstance.hide();
        }
    },
    ];
    function handleClick() {
        dialogInstance.show();
    }
    function onCreate() {
        maxBtn = document.getElementById('max-btn') as HTMLElement;
        maxBtn.onclick = (e: Event) => {
            var maximizeIcon;
            if (dialogInstance.element.classList.contains('dialog-minimized')) {
                dialogInstance.element.querySelector('#min-btn').classList.add('sf-
                icon-Minimize');
                dialogInstance.element.querySelector('#min-
                btn').classList.remove('sf-icon-Restore');
                dialogInstance.element.querySelector('#min-
                btn').setAttribute('title', 'Minimize');
            }
            if (!dialogInstance.element.classList.contains('dialog-maximized') &&
            !isFullScreen) {
                maximizeIcon = dialogInstance.element.querySelector(".e-dlg-header-
                content .sf-icon-Maximize");
            } else {

```

```

        maximizeIcon = dialogInstance.element.querySelector(".e-dlg-header-
content .sf-icon-Restore");
    }
    if (!dialogInstance.element.classList.contains('dialog-maximized')) {
        dialogInstance.element.classList.add('dialog-maximized');
        dialogInstance.show(true);
        maximizeIcon.classList.add('sf-icon-Restore');
        maximizeIcon.setAttribute('title', 'Restore');
        maximizeIcon.classList.remove('sf-icon-Maximize');
        dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        isFullScreen = true;
    } else {
        dialogInstance.element.classList.remove('dialog-maximized');
        dialogInstance.show(false);
        maximizeIcon.classList.remove('sf-icon-Restore');
        maximizeIcon.classList.add('sf-icon-Maximize');
        maximizeIcon.setAttribute('title', 'Maximize');
        dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
        dialogInstance.position = dialogOldPositions;
        dialogInstance.dataBind();
        isFullScreen = false;
    }
}
minBtn = document.getElementById('min-btn') as HTMLElement;
minBtn.onclick = (e: Event) => {
    var minimizeIcon = dialogInstance.element.querySelector(".e-dlg-
header-content .sf-icon-Minimize");
    if (!dialogInstance.element.classList.contains('e-dlg-fullscreen'))
    {
        if (!dialogInstance.element.classList.contains('dialog-minimized'))
        {
            dialogOldPositions = { X: dialogInstance.position.X, Y:
dialogInstance.position.Y }
            dialogInstance.element.classList.add('dialog-minimized');
            dialogInstance.element.classList.remove('dialog-maximized');
            dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
            dialogInstance.position = { X: 'center', Y: 'bottom' };
            dialogInstance.dataBind();
            minimizeIcon.classList.add('sf-icon-Restore');
            minimizeIcon.setAttribute('title', 'Restore');
        } else {
            dialogInstance.element.classList.remove('dialog-minimized');
            dialogInstance.element.querySelector('.e-dlg-
content').classList.remove('hide-content');
            minimizeIcon.classList.add('sf-icon-Minimize');
            minimizeIcon.setAttribute('title', 'Minimize');
            minimizeIcon.classList.remove('sf-icon-Restore');
            dialogInstance.position = dialogOldPositions;
            dialogInstance.dataBind();
        }
    } else {
        dialogInstance.show(false);
        dialogInstance.element.classList.remove('dialog-maximized');
        dialogInstance.element.classList.add('dialog-minimized');
    }
}

```



```

        dialogInstance.element.querySelector('.e-dlg-
content').classList.add('hide-content');
        minimizeIcon.classList.remove('sf-icon-Minimize');
        minimizeIcon.removeAttribute('title');
        dialogInstance.position = { X: 'center', Y: 'bottom' };
        dialogInstance.dataBind();
        isFullScreen = true;
    }
}
}
return (
    <div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1'
role='button' onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent id='dialog' created={onCreate.bind(this)}
header={header} animationSettings={settings} showCloseIcon={true}
closeOnEscape={true}
        width='300px' buttons={buttons} content='This is a dialog with
minimize and maximize buttons' ref={dialog => dialogInstance = dialog!}
        target='#dialog-target' />
    </div>
);
}
export default App;

```

Setting max height to the dialog in React Dialog component

By default, the maxHeight for the Dialog is calculated based on the target. If the target is not specified externally, the Dialog consider the body as target and will calculate the maxHeight based on it. We have an option to set the maxHeight of the Dialog in the [beforeOpen](#) event.

[Class-component]

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component {
    dialogInstance;
    handleClick() {
        this.dialogInstance.show(true);
    }
    onCreated = () => {
        // document.getElementById('dlgContent').style.display = 'block';
        this.dialogInstance.refreshPosition();
    };
    onBeforeOpen = (args) => {
        args.maxHeight = '300px';
    };
    render() {
        return (<div className="App" id='dialog-target'>
            <button className='e-control e-btn' id='targetButton1' role='button'
onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
            <DialogComponent header='Dialog' showCloseIcon={true} visible={false}
width='800px' showCloseIcon="true" target='#dialog-target' ref={dialog =>
this.dialogInstance = dialog} created={this.onCreated}
beforeOpen={this.onBeforeOpen}>

```

```

    <div className="dialogContent">
      <span>
        <b>JavaScript:</b><br />
        JavaScript is a high-level, dynamic, untyped, and
        interpreted programming language. It has been standardized in the ECMAScript
        language specification. Alongside HTML and CSS, it is
        one of the three essential technologies of World Wide Web
        content production; the majority of websites employ it
        and it is supported by all modern Web browsers without
        plug-ins. JavaScript is a prototype-based programming
        language with first-class functions, making it a multi-paradigm language,
        supporting object-oriented , imperative, and functional
        programming styles.
        <br /><br /><br />
        <b>MVC:</b><br />
        Model-view-controller (MVC) is a software architecture
        pattern which separates the representation of information from the user's
        interaction with it. The model consists of application data, business rules,
        logic, and functions. A view can be any output representation of data, such
        as a chart or a diagram. Multiple views of the same data are possible, such
        as a bar chart for management and a tabular view for accountants. The
        controller mediates input, converting it to commands for the model or
        view. The central ideas behind MVC are code reusability and in addition to
        dividing the application into three kinds of components, the MVC design
        defines the interactions between them.
        A controller can send commands to its associated view to
        change the view's presentation of the model (e.g., by scrolling through a
        document). It can also send commands to the model to update the model's
        state (e.g., editing a document).
        A model notifies its associated views and controllers
        when there is a change in its state. This notification allows the views to
        produce updated output, and the controllers to change the available set of
        commands. A passive implementation of MVC omits these notifications because
        the application does not require them or the software platform does not
        support them.
        A view requests from the model the information that it
        needs to generate an output representation to the user.
      </span>
    </div>
  </DialogComponent>
</div>);
}
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
class App extends React.Component<{}, {}> {
  public dialogInstance: DialogComponent;
  public handleClick() {
    this.dialogInstance.show(true);
  }
  public onCreated = () => {
    // document.getElementById('dlgContent').style.display = 'block';
  }
}

```

```

        this.dialogInstance.refreshPosition();
    }
    public onBeforeOpen = (args: beforeOpenEventArgs) => {
        args.maxHeight = '300px';
    }
    public render() {
        return (
            <div className="App" id='dialog-target'>
                <button className='e-control e-btn' id='targetButton1' role='button'
                    onClick={this.handleClick = this.handleClick.bind(this)}>Open</button>
                <DialogComponent header='Dialog' showCloseIcon={true} visible={false}
                    width='800px' showCloseIcon = "true" target='#dialog-target' ref={dialog =>
                        this.dialogInstance = dialog!} created = {this.onCreated} beforeOpen =
                        {this.onBeforeOpen}>
                    <div className="dialogContent">
                        <span>
                            <b>JavaScript:</b><br />
                            JavaScript is a high-level, dynamic, untyped, and
                            interpreted programming language. It has been standardized in the ECMAScript
                            language specification. Alongside HTML and CSS, it is
                            one of the three essential technologies of World Wide Web
                            content production; the majority of websites employ it
                            and it is supported by all modern Web browsers without
                            plug-ins. JavaScript is a prototype-based programming
                            language with first-class functions, making it a multi-paradigm language,
                            supporting object-oriented , imperative, and functional
                            programming styles.
                            <br /><br /><br />
                            <b>MVC:</b><br />
                            Model-view-controller (MVC) is a software architecture
                            pattern which separates the representation of information from the user's
                            interaction with it. The model consists of application data, business rules,
                            logic, and functions. A view can be any output representation of data, such
                            as a chart or a diagram. Multiple views of the same data are possible, such
                            as a bar chart for management and a tabular view for accountants. The
                            controller mediates input, converting it to commands for the model or
                            view. The central ideas behind MVC are code reusability and in addition to
                            dividing the application into three kinds of components, the MVC design
                            defines the interactions between them.
                            A controller can send commands to its associated view to
                            change the view's presentation of the model (e.g., by scrolling through a
                            document). It can also send commands to the model to update the model's
                            state (e.g., editing a document).
                            A model notifies its associated views and controllers
                            when there is a change in its state. This notification allows the views to
                            produce updated output, and the controllers to change the available set of
                            commands. A passive implementation of MVC omits these notifications because
                            the application does not require them or the software platform does not
                            support them.
                            A view requests from the model the information that it
                            needs to generate an output representation to the user.
                            </span>
                        </div>
                    </DialogComponent>
                </div>);
    }
}

```

```
export default App;
```

[Functional-component]

APP.JSX

```
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
    let dialogInstance;
    function handleClick() {
        dialogInstance.show(true);
    }
    function onCreate() {
        // document.getElementById('dlgContent').style.display = 'block';
        dialogInstance.refreshPosition();
    }
    function onBeforeOpen(args) {
        args.maxHeight = '300px';
    }
    return (<div className="App" id='dialog-target'>
        <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
        <DialogComponent header='Dialog' showCloseIcon={true}
visible={false} width='800px' showCloseIcon={true} target='#dialog-target'
ref={dialog => dialogInstance = dialog} created={onCreate}
beforeOpen={onBeforeOpen}>
            <div className="dialogContent">
                <span>
                    <b>JavaScript:</b><br />
                    JavaScript is a high-level, dynamic, untyped, and
interpreted programming language. It has been standardized in the ECMAScript
                    language specification. Alongside HTML and CSS, it is
one of the three essential technologies of World Wide Web
                    content production; the majority of websites employ it
and it is supported by all modern Web browsers without
                    plug-ins. JavaScript is a prototype-based programming
language with first-class functions, making it a multi-paradigm language,
                    supporting object-oriented , imperative, and
functional programming styles.
                    <br /><br /><br />
                    <b>MVC:</b><br />
                    Model-view-controller (MVC) is a software architecture
pattern which separates the representation of information from the user's
interaction with it. The model consists of application data, business rules,
logic, and functions. A view can be any output representation of data, such
as a chart or a diagram. Multiple views of the same data are possible, such
as a bar chart for management and a tabular view for accountants. The
controller mediates input, converting it to commands for the model or
view. The central ideas behind MVC are code reusability and in addition to
dividing the application into three kinds of components, the MVC design
defines the interactions between them.
                    A controller can send commands to its associated view
to change the view's presentation of the model (e.g., by scrolling through a
document). It can also send commands to the model to update the model's
state (e.g., editing a document).
```

A model notifies its associated views and controllers when there **is** a change **in** its state. This notification allows the views to produce updated output, and the controllers to change the available **set** of commands. A passive implementation of MVC omits these notifications because the application does not require them or the software platform does not support them.

A view requests **from** the model the information that it needs to generate an output representation to the user.

```

    </span>
  </div>
</DialogComponent>
</div>;
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import * as React from "react";
function App() {
  let dialogInstance: DialogComponent;
  function handleClick() {
    dialogInstance.show(true);
  }
  function onCreate () {
    // document.getElementById('dlgContent').style.display = 'block';
    dialogInstance.refreshPosition();
  }
  function onBeforeOpen(args: beforeOpenEventArgs) {
    args.maxHeight = '300px';
  }
  return (
    <div className="App" id='dialog-target'>
      <button className='e-control e-btn' id='targetButton1' role='button'
onClick={handleClick.bind(this)}>Open</button>
      <DialogComponent header='Dialog' showCloseIcon={true}
visible={false} width='800px' showCloseIcon = {true} target='#dialog-target'
ref={dialog => dialogInstance = dialog!} created = {onCreate} beforeOpen =
{onBeforeOpen}>
        <div className="dialogContent">
          <span>
            <b>JavaScript:</b><br />
            JavaScript is a high-level, dynamic, untyped, and
interpreted programming language. It has been standardized in the ECMAScript
language specification. Alongside HTML and CSS, it is
one of the three essential technologies of World Wide Web
content production; the majority of websites employ it
and it is supported by all modern Web browsers without
plug-ins. JavaScript is a prototype-based programming
language with first-class functions, making it a multi-paradigm language,
supporting object-oriented , imperative, and
functional programming styles.
            <br /><br /><br />
            <b>MVC:</b><br />
            Model-view-controller (MVC) is a software architecture
pattern which separates the representation of information from the user's

```

interaction with it. The model consists of application data, business rules, logic, and functions. A view can be any output representation of data, such as a chart or a diagram. Multiple views of the same data are possible, such as a bar chart for management and a tabular view for accountants. The controller mediates input, converting it to commands for the model or view. The central ideas behind MVC are code reusability and in addition to dividing the application into three kinds of components, the MVC design defines the interactions between them.

A controller can send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document). It can also send commands to the model to update the model's state (e.g., editing a document).

A model notifies its associated views and controllers when there is a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. A passive implementation of MVC omits these notifications because the application does not require them or the software platform does not support them.

A view requests from the model the information that it needs to generate an output representation to the user.

```

        </span>
      </div>
    </DialogComponent>
  </div>);
}
export default App;

```

React hooks dialog in React Dialog component

You can show the Dialog by using the React state change hook with Visible property. In the below example, we have rendered the React JSX component as content of Dialog component.

APP.JSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { TextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { useState } from 'react';
import * as React from 'react';
function App() {
  const [name, setName] = useState('');
  const [visibility, setDialogVisibility] = useState(true);
  function handleClick() {
    setDialogVisibility(true);
  }
  function dialogClose() {
    setDialogVisibility(false);
  }
  return (
    <div>
      <div className="control_wrapper" id="control_wrapper">
        <button className="e-control e-btn dlgbtn" onClick={handleClick}
id="dialogBtn">
          { ' ' }
          Open
        </button>
        <DialogComponent id="defaultdialog" showCloseIcon={true}
visible={visibility} close={dialogClose} header="Textbox">
          <div>

```

```

        <div>
            <div> Enter your content </div>
        </div>
        <div>
            <div>
                <TextBoxComponent placeholder="Enter Name" value={name}
input={ (e) => setName(e.value) }></TextBoxComponent>
                <div>Entered content: {name}</div>
            </div>
        </div>
    </DialogComponent>
</div>
</div>);
}
export default App;

```

APP.TSX

```

import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { TextBoxComponent } from '@syncfusion/ej2-react-inputs';
import { useState } from 'react';
import * as React from 'react';
function App() {
    const [name, setName] = useState('');
    const [visibility, setDialogVisibility] = useState(true);
    function handleClick() {
        setDialogVisibility(true);
    }
    function dialogClose() {
        setDialogVisibility(false);
    }
    return (
        <div>
            <div className="control_wrapper" id="control_wrapper">
                <button
                    className="e-control e-btn dlgbtn"
                    onClick={handleClick}
                    id="dialogBtn"
                >
                    { ' ' }
                    Open
                </button>
                <DialogComponent
                    id="defaultdialog"
                    width={"25%"}
                    showCloseIcon={true}
                    visible={visibility}
                    close={dialogClose}
                    header="Textbox"
                >
                    <div>
                        <div>
                            <div> Enter your content </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
}

```

```

        <div>
            <TextBoxComponent
                placeholder="Enter Name"
                value={name}
                input={(e) => setName(e.value)}
            ></TextBoxComponent>
            <div>Entered content: {name}</div>
        </div>
    </div>
</DialogComponent>
</div>
</div>
);
}
export default App;

```

Ej1 api migration in React Dialog component

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

Accessibility and Localization

{% raw %}

<!-- markdownlint-disable MD033 -->

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

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

| Keyboard Navigation | **Property** : allowKeyboardNavigation

<EJ.Dialog allowKeyboardNavigation={true}></EJ.Dialog> | No separate Property for enable/disable keyboard navigation. Its enabled by default. |

| Localization | **Property** : locale

<EJ.Dialog locale="es-ES"></EJ.Dialog> | **Property** : locale

<DialogComponent locale="es-ES" /> |

| Right to left | **Property**: enableRTL

<EJ.Dialog enableRTL={true}></EJ.Dialog> | **Property**: enableRTL

<DialogComponent enableRtl={true} /> |

Header

<!-- markdownlint-disable MD033 -->

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

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

| Header Content | **Property** : title

<EJ.Dialog title="EJ1 Dialog header"></EJ.Dialog>
 Method : setTitle
 \$(' #dialog').ejDialog('setTitle', 'EJ1 Dialog Header'); | **Property** : header

<DialogComponent header='EJ2 Dialog' /> |

| close button | **Property** : actionButtons

<EJ.Dialog actionButtons={this.actionbuttons}></EJ.Dialog>

actionbuttons: any; constructor() { this.actionbuttons = ['close'];}
 | **Property** : showCloseIcon

<DialogComponent showCloseIcon={true} /> |

| Event triggers when click on action buttons | **Event:** actionButtonClick

<EJ.Dialog
actionButtons={this.actionButtons}>actionButtonClick={this.buttonAction}</EJ.Dialog>

>actionButtons: any; constructor() { this.actionButtons = ['close'];, buttonAction(event) {} }

| Not Applicable |

| Minimize | **Property :** actionButtons

<EJ.Dialog
actionButtons={this.actionButtons}></EJ.Dialog>

actionButtons: any; constructor() {
this.actionButtons = ['minimize'];}
 | Not Applicable |

| Maximize | **Property :** actionButtons

<EJ.Dialog
actionButtons={this.actionButtons}></EJ.Dialog>

actionButtons: any; constructor() {
this.actionButtons = ['maximize'];}
 | Not Applicable |

| Collapse /Expand | **Property :** actionButtons
 Method : collapse(), expand ()

<EJ.Dialog actionButtons={this.actionButtons}></EJ.Dialog>

actionButtons: any;
constructor() { this.actionButtons = ['collapsible'];}

\$('#dialog').ejDialog('collapse');

\$('#dialog').ejDialog('expand') | Not Applicable |

| Event triggers when expanding the collapsed dialog | **Event:** expand

<EJ.Dialog
id="dialog" expand={this.expandAction}></EJ.Dialog>

expandAction(event) {}
|
Not Applicable |

| Event triggers when collapsing the expanded dialog | **Event:** collapse

<EJ.Dialog
id="dialog" collapse={this.collapseAction}></EJ.Dialog>

collapseAction(event) {}

| Not Applicable |

| Pin | **Property :** actionButtons

<EJ.Dialog
actionButtons={this.actionButtons}></EJ.Dialog>

actionButtons: any; constructor() {
this.actionButtons = ['pin'];}
 | Not Applicable |

| Header visibility | **Property:** showHeader

<EJ.Dialog showHeader={true}></EJ.Dialog> |
Not Applicable |

| Close on escape key press | **Property :** closeOnEscape

<EJ.Dialog
closeOnEscape={true}></EJ.Dialog>| **Property :** closeOnEscape

<DialogComponent
closeOnEscape={true} /> |

Footer

<!-- markdownlint-disable MD033 -->

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

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

| Footer Content | **Property :** footerTemplateId

<EJ.Dialog
footerTemplateId='sample'></EJ.Dialog>| **Property:** footerTemplate

<DialogComponent footerTemplate= {this.footerTemplate} />

footerTemplate {
<button>Submit</button> } |

| Footer action buttons | Not applicable | **Property :** buttons

<DialogComponent
buttons={this.dialogButtons} />

constructor(props: {}) { this.dialogButtons = [{ click:
this.dlgButtonClick, buttonModel: {content: 'OK', isPrimary: true} }]
 |

| Footer visibility | **Property** : showFooter

<EJ.Dialog showFooter={true}></EJ.Dialog> | Not Applicable |

Content

<!-- markdownlint-disable MD033 -->

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

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

| Dialog content | **Method** : setContent

<EJ.Dialog id="basicDialog"></EJ.Dialog>
 \$('#basicDialog').ejDialog('setContent', 'Dialog Content') | **Property** : content

<DialogComponent content= 'Dialog content' /> |

| Loading content using AJAX request | **Property** : contentType, contentUrl

<EJ.Dialog contentType="ajax" contentType=""></EJ.Dialog> | Not Applicable |

| Event triggers after the dialog content loaded in DOM | **Event**: contentLoad

<EJ.Dialog contentLoad={this.onLoad}></EJ.Dialog>

onLoad(event) {}
 | Not Applicable |

| Event trigger when fails to load ajax content | **Event**: ajaxError

<EJ.Dialog ajaxError={this.onAjaxError}></EJ.Dialog>

onAjaxError(event) {}
 | Not Applicable |

| Event trigger when load ajax content successfully | **Event**: ajaxSuccess

<EJ.Dialog ajaxSuccess={this.onAjaxSuccess}></EJ.Dialog>

onAjaxSuccess(event) {}
 | Not Applicable |

Animation

<!-- markdownlint-disable MD033 -->

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

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

| Enabling Animation | **Property** : enableAnimation

<EJ.Dialog enableAnimation={true}></EJ.Dialog> | Not Applicable |

| Animation effects | **Property** : animation.show.effect

<EJ.Dialog animation={this.animation}></EJ.Dialog>

animation: object = { show: { effect: 'slide' } };
 | **Property** : animationSettings.effect

 <DialogComponent animation={this.animationSettings} />

private animationSettings: Object; constructor(props: {}) { this.animationSettings = { effect: 'Zoom' } }
 |

| Animation duration | **Property**: animation.show.duration

<EJ.Dialog animation={this.animation}></EJ.Dialog>

animation: object = { show: { effect: 'slide', duration: 500 } };
 | **Property** : animationSettings.duration

 <DialogComponent animation={this.animationSettings} />

private animationSettings: Object; constructor(props: {}) { this.animationSettings = { effect: 'Zoom', duration: 500 } }
 |

| Animation delay | Not applicable | **Property**: animationSettings.delay

 <DialogComponent animation={this.animationSettings} />

private animationSettings: Object; constructor(props: {}) { this.animationSettings = { effect: 'Zoom', duration: 500, delay: 500 } }
 |

Draggable and resizing

<!-- markdownlint-disable MD033 -->

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

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

| Draggable dialog | **Property** : allowDraggable

<EJ.Dialog
allowDraggable={true}></EJ.Dialog> | **Property** : allowDragging

<DialogComponent
allowDragging= {true} /> |

| Event triggers when the user drags the dialog | **Event**: drag

<EJ.Dialog drag={this.onDrag
></EJ.Dialog>

onDrag(event) {}
 | **Event**: drag

<DialogComponent drag=
{this.onDrag} />

private onDrag() {}
 |

| Event triggers when the start to drag the dialog | **Event**: dragStart

<EJ.Dialog
dragStart={this.onDragStart }></EJ.Dialog>

onDragStart(event) {}
 | **Event**:
dragStart

<DialogComponent dragStart= {this.onDragStart} />

private
onDragStart() {}
 |

| Event triggers when the stops to drag the dialog | **Event**: dragStop

<EJ.Dialog
dragStop={this.onDragStop }></EJ.Dialog>

onDragStop(event) {}
 | **Event**:
dragStop

<DialogComponent dragStop= {this.onDragStop} />

private
onDragStop() {}
 |

| Resizing dialog | **Property** : enableResize

<EJ.Dialog enableResize={true}></EJ.Dialog> |
Not applicable |

| Event triggers when resizing the dialog | **Event**: resize

<EJ.Dialog resize={this.onReSize
></EJ.Dialog>

onReSize(event) {}
 | Not Applicable |

| Event triggers when starts to resizing the dialog | **Event**: resizeStart

<EJ.Dialog
resizeStart={this.onResizeStart }></EJ.Dialog>

onResizeStart(event) {}
 | Not
Applicable |

| Event triggers when the stops to resizing the dialog | **Event**: resizeStop

<EJ.Dialog
resizeStop={this.onResizeStop }></EJ.Dialog>

onResizeStop(event) {}
 | Not
Applicable |

Target

<!-- markdownlint-disable MD033 -->

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

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

| Target element to append dialog in document | **Property** : target

<EJ.Dialog
target='#dialogTarget'></EJ.Dialog> | **Property**: target

<DialogComponent
target='#dialogTarget' /> |

| Element for draggable area | **Property** : containment

<EJ.Dialog
containment='#dragArea'></EJ.Dialog> | Not applicable |

Position

<!-- markdownlint-disable MD033 -->

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

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

| Customizing dialog position using X, Y coordinate values | **Property** : position

<EJ.Dialog position={this.dialogPosition}></EJ.Dialog>

dialogPosition: any = { position: { X: 300, Y: 100 } }
| **Property** : position

<DialogComponent position={this.dialogPosition} />

private dialogPosition: object { position: {X:300, Y:100} }
 |

| positioning dialog using position values | Not Applicable | **Property**: position

<DialogComponent position={this.dialogPosition} />

private dialogPosition: object { position: {X: 'center', Y: 'center'} }
 |

Visibility

<!-- markdownlint-disable MD033 -->

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

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

| Render dialog in visible/hidden state | **Property**: showOnInit

<EJ.Dialog showOnInit={true}></EJ.Dialog> | **Property**: visible

<DialogComponent visible={this.state.hideDialog} />

constructor(props: {}) { this.setState({ hideDialog: false }) }; |

Dialog Mode

<!-- markdownlint-disable MD033 -->

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

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

| Render modal dialog | **Property** : enableModal

<EJ.Dialog enableModal={true}></EJ.Dialog> | **Property** : isModal

<DialogComponent isModal={true} /> |

Tooltip

<!-- markdownlint-disable MD033 -->

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

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

| Sets the tooltip for dialog buttons | **Property** : tooltip

<EJ.Dialog tooltip={this.tooltip}></EJ.Dialog>

tooltip: object { close: 'Exit' }
 | No Separate Property for tooltip. It renders based on locale text. |

Control State

<!-- markdownlint-disable MD033 -->

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

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

| Enable/Disable the control | **Property** : enabled

 <EJ.Dialog
enabled={false}></EJ.Dialog> | Not Applicable |

| Enable/ Disable page scrolling | **Property**: backgroundScroll

<EJ.Dialog
backgroundScroll={false}></EJ.Dialog> | No separate Property for disabling page scroll. By default,
scrolling prevented for modal dialog |

State Maintenance

<!-- markdownlint-disable MD033 -->

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

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

| Save the model values in local storage or cookies | **Property** : enablePersistence

<EJ.Dialog
enablePersistence={true}></EJ.Dialog> | **Property** : enablePersistence

<DialogComponent enablePersistence= {true} /> |

Common

<!-- markdownlint-disable MD033 -->

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

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

| Adjusting Height | **Property** : height

<EJ.Dialog height={400}></EJ.Dialog> | **Property** :
height

 <DialogComponent height= '50%' /> |

| Adjusting width | **Property**: width

<EJ.Dialog width={400}></EJ.Dialog> | **Property** :
width

 <DialogComponent width= '50%' /> |

| Adding custom class | **Property**: cssClass

<EJ.Dialog cssClass='custom-
class'></EJ.Dialog> | **Property**: cssClass

 <DialogComponent cssClass='custom-class' />
|

| Adding zIndex | **Property**: zIndex

 <EJ.Dialog zIndex={2000}></EJ.Dialog> | **Property**:
zIndex

 <DialogComponent zIndex='2000' /> |

| Maximum height | **Property**: maxHeight

 <EJ.Dialog maxHeight={600}></EJ.Dialog> |
Not Applicable |

| Maximum width | **Property**: maxWidth

 <EJ.Dialog maxWidth={600}></EJ.Dialog> | Not
Applicable |

| Minimum height | **Property**: minHeight

 <EJ.Dialog minHeight={300}></EJ.Dialog> | Not
Applicable |

| Minimum width | **Property**: minWidth

 <EJ.Dialog minWidth={300}></EJ.Dialog> | Not
Applicable |

| Adding html attributes | **Property**: htmlAttributes

 <EJ.Dialog
htmlAttributes={this.htmlAttributes}></EJ.Dialog>

htmlAttributes: object { class: 'my-
class' }
 | Not Applicable |

| Custom icon in the header | **Property:** faviconCSS

 <EJ.Dialog faviconCSS='custom-icon'></EJ.Dialog> | Not Applicable |

| Rounded corner appearance | **Property:** showRoundedCorner

 <EJ.Dialog showRoundedCorner={true}></EJ.Dialog> | Not Applicable |

| Make control flexible for mobile view | **Property:** isResponsive

 <EJ.Dialog isResponsive={true}></EJ.Dialog> | Not Applicable |

| Close the Dialog | **Method:** close()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('close') | **Method :** hide()

<DialogComponent id='dialog' ref={dialog => this.dialogInstance = dialog />

constructor(props: {})) { this.dialogInstance.hide(); }
 |

| Event triggers before the dialog closes | **Event:** beforeClose

<EJ.Dialog beforeClose={this.onBeforeClose}></EJ.Dialog>

onBeforeClose (event) {}
 | **Event:** beforeClose

<DialogComponent beforeClose= {this.onBeforeClose.bind(this)} />

private onBeforeClose() {}
 |

| Event triggers when the dialog closes | **Event:** close

<EJ.Dialog close={this.onClose}></EJ.Dialog>

onClose (event) {}
 | **Event:** close

<DialogComponent close= {this.onClose.bind(this)} />

private onClose() {}
 |

| Destroy the control | **Method:** destroy()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('destroy') | **Method:** destroy()

<DialogComponent id='dialog' ref={dialog => this.dialogInstance = dialog />

constructor(props: {})) { this.dialogInstance.destroy(); }
 |

| Focus the dialog element | **Method:** focus()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('focus') | Not Applicable |

| Check whether the dialog is open | **Method:** isOpen()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('isOpen') | Not Applicable |

| Maximize the dialog | **Method:** maximize()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('maximize') | Not Applicable |

| Minimize the dialog | **Method:** minimize()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('minimize') | Not Applicable |

| Open the dialog | **Method:** open()

 <EJ.Dialog id ='dialog'></EJ.Dialog>
\$('#dialog').ejDialog('open') | **Method :** show()

<DialogComponent id='dialog' ref={dialog => this.dialogInstance = dialog />

constructor(props: {})) { this.dialogInstance.show(); }
 |

| Event trigger before the dialog opens | **Event:** beforeOpen

 <EJ.Dialog beforeOpen={this.onBeforeOpen}></EJ.Dialog>

onBeforeOpen (event) {}
 | **Event:** beforeOpen

<DialogComponent beforeOpen= {this.onBeforeOpen.bind(this)} />

private onBeforeOpen() {}
 |

```

| Event triggers when the opens the dialog | Event: open <br/><br/><EJ.Dialog
open={this.onOpen}></EJ.Dialog><br/><br/>onOpen (event) {} <br/> | Event: open
<br/><br/><DialogComponent open= {this.onOpen.bind(this)} /><br/><br/>private onOpen() {}
<br/> |

| Refresh the dialog | Method: refresh() <br/><br/><EJ.Dialog id ='dialog'></EJ.Dialog>
<br/>$('#dialog').ejDialog('refresh') | Method : refreshPosition() <br/><br/><DialogComponent
id='dialog' ref={dialog => this.dialogInstance = dialog /><br/><br/>constructor(props: {}) {
this.dialogInstance.refreshPosition(); } <br/> |

| Pin/ unpin the dialog | Method: pin <br/><br/><EJ.Dialog id ='dialog'></EJ.Dialog>
<br/>$('#dialog').ejDialog('pin'); <br/>$('#dialog').ejDialog('unpin'); | Not Applicable |

| Event triggers after the dialog created successfully | Event: create <br/><br/><EJ.Dialog
create={this.onCreate}></EJ.Dialog><br/><br/>onCreate (event) {} <br/> | Event : created <br/>
<br/><DialogComponent created= {this.onCreated.bind(this)} /><br/><br/>private onCreated() {}
<br/> |

| Event triggers when the control destroyed successfully | Event: destroy <br/><br/><EJ.Dialog
destroy={this.onDestroy}></EJ.Dialog><br/><br/>onDestroy (event) {} <br/> | Not Applicable |

| Event triggers on clicking on modal dialog overlay | Not Applicable | Event : overlayClick
<br/><br/><DialogComponent overlayClick= {this.onOverlayClick.bind(this)} /><br/><br/>private
onOverlayClick() {} <br/> |

{% endraw %}

```

Document Editor

Overview

The Document Editor component is used to create, edit, view, and print Word documents in web applications. All the user interactions and editing operations that run purely in the client-side provides much faster editing experience to the users.

Key Features

- [Opens](#) the native Syncfusion Document Text (*.sfdt) format documents in the client-side.
- [Saves the documents](#) in the client-side as Syncfusion Document Text (.sfdt) and Word document (.docx).
- Supports document elements like text, [image](#), [table](#), fields, [bookmark](#), [shapes](#), [section](#), [header and footer](#).
- Supports the commonly used fields like [hyperlink](#), page number, page count, and table of contents.
- Supports formats like [text](#), [paragraph](#), [bullets and numbering](#), [table](#), [page settings](#), etc.
- Provides support to create, edit, and apply [paragraph and character styles](#).
- Provides support to [find and replace](#) text within the document.
- Supports all the common editing and formatting operations along with [undo and redo](#).
- Provides support to [cut](#), [copy](#), and [paste](#) rich text contents within the component. Also allows pasting simple text to and from other applications.
- Provides support to insert, and edit [form fields](#).

- Provides support to insert, and edit [comments](#).
- Provides support to track the [inserted and deleted content](#).
- Provides support to perform [spell checking](#) for any input text
- Allows user interactions like [zoom](#), [scroll](#), select contents through touch, mouse, and keyboard.
- Provides intuitive UI options like context menu, [dialogs](#), and [navigation pane](#).
- [Localizes](#) all the static text to any desired language.
- Allows to create a lightweight Word viewer using module injection to view and [prints](#) Word documents.
- Provides a [server-side helper library](#) to open the Word documents like DOCX, DOC, WordML, RTF, and Text, by converting it to SFDT file format.

Supported Web platforms

- [Javascript\(ES5\)](#)
- [Javascript](#)
- [Angular](#)
- [React](#)
- [Vue](#)
- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Blazor](#)

Supported platforms for server-side dependencies

You can deploy web APIs for server-side dependencies of Document Editor component in the following platforms.

- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Java](#)

To know more about server-side dependencies, refer this [pageLink to the Video](#).

Which operations require server-side interaction

- Open file formats other than SFDT
- Paste with formatting
- Restrict editing
- Spellcheck
- Save as file formats other than SFDT and DOCX

Note: If you don't require the above functionalities then you can deploy as pure client-side component without any server-side interactions.

Getting Started

This section explains the steps to create a Word document editor within your application and demonstrates the basic usage of the DocumentEditor component.

To get started quickly with DocumentEditor component, you can check the video below.

Dependencies

Following is the list of minimum dependencies required to use the documenteditor.

```
`javascript
|-- @syncfusion/ej2-react-documenteditor
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-build
|-- @syncfusion/ej2-buttons
|-- @syncfusion/ej2-compression
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-file-utils
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-splitbuttons
|-- @syncfusion/ej2-documenteditor
|-- @syncfusion/ej2-charts
`,`
```

Server side dependencies

The Document Editor component requires server-side interactions for the following operations:

- [Open file formats other than SFDT](#)
- [Paste with formatting](#)
- [Restrict editing](#)
- [Spellcheck](#)
- [Save as file formats other than SFDT and DOCX](#)

Note: If you don't require the above functionalities then you can deploy as pure client-side component without any server-side interactions.

To know about server-side dependencies, please refer this [page](#).

Setup for Local Development

You can use [create-react-app](#) to setup the applications.

To install `create-react-app` run the following command.

```
`npm install -g create-react-app`
```

,

- To setup basic **React** sample use following commands.

```
<div class='tsx'>
```

,

```
create-react-app quickstart --scripts-version=react-scripts-ts
```

```
cd quickstart
```

```
npm install
```

,

```
</div>
```

```
<div class='jsx'>
```

,

```
create-react-app quickstart
```

```
cd quickstart
```

```
npm install
```

,

```
</div>
```

Note: Creating react app without mentioning typescript, will create **src/app.js** instead of **src/app.tsx** file. You can also add the below code snippet in **src/app.js** file.
To create react app using typescript, you can use this **create-react-app quickstart --template typescript** command. This will create **app.tsx** file under **src** folder of project location.

Adding Syncfusion packages

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

You can choose the component that you want to install.

To install Document Editor component, use the following command

,

```
npm install @syncfusion/ej2-react-documenteditor --save
```

,

Adding CSS reference

Add Document Editor component and its dependent component styles as given below in **src/App.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-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-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import "../node_modules/@syncfusion/ej2-documenteditor/styles/material.css";
`
```

To know about individual component CSS, please refer to

[Individual Component theme files](#) section.

Adding Component

You can add **DocumentEditorContainer** Component with predefined toolbar and properties pane options or **DocumentEditor** component with customize options.

Note: Starting from v19.3.0.x, we have optimized the accuracy of text size measurements such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API [to disable it and retain the document pagination behavior of older versions](#).

DocumentEditor component

DocumentEditor Component is used to create , view and edit word documents. In this , you can customize the UI options based on your requirements to modify the document.

Adding DocumentEditor component

Now, you can start adding DocumentEditor component in the application. For getting started, add the DocumentEditor component in **src/App.tsx** file using following code.

Add the below code in the **src/App.tsx** to initialize the DocumentEditor.

```
`ts
import * as React from 'react';

import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport, Selection, Search,
Editor, ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog } from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);

function App() {

return (<DocumentEditorComponent id="container" height={'330px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
isReadOnly={false} enablePrint={true} enableSelection={true} enableEditor={true}
```

```

enableEditorHistory={true} enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
enableImageResizer={true} enableListDialog={true} enablePageSetupDialog={true}
enableSfdtExport={true} enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true} enableTablePropertiesDialog={true} enableTextExport={true}
enableWordExport={true} />;
}

export default App

```

Run the DocumentEditor application

The [create-react-app](#) will pre-configure the project to compile and run the application in browser. Use the following command to run the application.

```
npm start
```

Document Editor output will be displayed as follows.

INDEX.JSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
function Default() {
    return (<DocumentEditorComponent id="container" height={'330px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" isReadOnly={false} enablePrint={true}
enableSelection={true} enableEditor={true} enableEditorHistory={true}
enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
enableHyperlinkDialog={true} enableImageResizer={true}
enableListDialog={true} enablePageSetupDialog={true} enableSfdtExport={true}
enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
enableTextExport={true} enableWordExport={true} />);

```

```

}
export default Default;
ReactDOM.render(<Default />, document.getElementById('sample'));

```

INDEX.TSX

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, DocumentEditor, RequestNavigateEventArgs,
  ViewChangeEventArgs,
  Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
  ImageResizer, EditorHistory,
  ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
  TableOfContentsDialog,
  PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
  BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
  CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
  Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
  OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
  TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
  ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
  CellOptionsDialog, StylesDialog);
function Default() {
  return (
    <DocumentEditorComponent id="container" height={'330px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-
      services/api/documenteditor/" isReadOnly={false} enablePrint={true}
        enableSelection={true} enableEditor={true}
      enableEditorHistory={true}
        enableContextMenu={true} enableSearch={true}
      enableOptionsPane={true}
        enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
      enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
      enableHyperlinkDialog={true} enableImageResizer={true}
      enableListDialog={true}
        enablePageSetupDialog={true} enableSfdtExport={true}
        enableStyleDialog={true} enableTableOfContentsDialog={true}
        enableTableOptionsDialog={true}
      enableTablePropertiesDialog={true}
        enableTextExport={true} enableWordExport={true} />
  );
}
export default Default
ReactDOM.render(<Default />, document.getElementById('sample'));

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>

```

```

<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

[DocumentEditorContainer component](#)

DocumentEditorContainer Component is also used to create, view and edit word document. But here, you can use predefined toolbar and properties pane to view and modify word document.

[Adding DocumentEditorContainer component](#)

Now, you can start adding DocumentEditorContainer component in the application. For getting started, add the DocumentEditorContainer component in `src/App.tsx` file using following code.

Add the below code in the `src/App.tsx` to initialize the DocumentEditor.

```
{% raw %}
```

```
`ts
```

```
import * as React from 'react';
```

```
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  return (<DocumentEditorContainerComponent id="container" style={{ 'height': '590px' }}
  serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
  enableToolbar={true}/>);
}
export default App
```

```
{% endraw %}
```

Run the DocumentEditorContainer application

The [create-react-app](#) will pre-configure the project to compile and run the application in browser. Use the following command to run the application.

```
npm start
```

DocumentEditorContainer output will be displayed as follows.

INDEX.JSX

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function Default() {
  return (<DocumentEditorContainerComponent id="container" height={'590px'}
  serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/" enableToolbar={true}/>);
}
export default Default;
ReactDOM.render(<Default />, document.getElementById('sample'));
```

INDEX.TSX

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent, Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function Default() {
  return (
    <DocumentEditorContainerComponent id="container" height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/" enableToolbar={true} />);
}
```

```
export default Default
ReactDOM.render(<Default />, document.getElementById('sample'));
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Frequently Asked Questions

- [How to localize the Documenteditor container.](#)
- [How to load the document by default.](#)
- [How to customize tool bar.](#)

- [How to resize Document editor component.](#)

Creating a Next.js Application Using Syncfusion React Components

This section provides a step-by-step guide for setting up a Next.js application and integrating the Syncfusion React Document Editor component.

What is Next.js?

[Next.js](#) is a React framework that makes it easy to build fast, SEO-friendly, and user-friendly web applications. It provides features such as server-side rendering, automatic code splitting, routing, and API routes, making it an excellent choice for building modern web applications.

Prerequisites

Before getting started with the Next.js application, ensure the following prerequisites are met:

- [Node.js 16.8](#) or later.
- The application is compatible with macOS, Windows, and Linux operating systems.

Create a Next.js application

To create a new **Next.js** application, use one of the commands that are specific to either NPM or Yarn.

NPM

```
npx create-next-app@latest
```

YARN

```
yarn create next-app
```

Using one of the above commands will lead you to set up additional configurations for the project as below:

1. Define the project name: Users can specify the name of the project directly. Let's specify the name of the project as **ej2-nextjs-documenteditor**.

CMD

```
√ What is your project named? » ej2-nextjs-documenteditor
```

2. Select the required packages.

CMD

```
√ What is your project named? ... ej2-nextjs-documenteditor
√ Would you like to use TypeScript? ... No / `Yes`
√ Would you like to use ESLint? ... No / `Yes`
√ Would you like to use Tailwind CSS? ... `No` / Yes
√ Would you like to use `src/` directory? ... No / `Yes`
√ Would you like to use App Router? (recommended) ... No / `Yes`
√ Would you like to customize the default import alias? ... `No` / Yes
Creating a new Next.js app in D:\ej2-nextjs-documenteditor.
```

3. Once complete the above mentioned steps to create `ej2-nextjs-documenteditor`, navigate to the directory using the below command:

CMD

```
cd ej2-nextjs-documenteditor
```

The application is ready to run with default settings. Now, let's add Syncfusion components to the project.

Install Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

Here, the [React Document Editor component](#) is used in the project. To install the React Document Editor component, use the following command:

NPM

```
npm install @syncfusion/ej2-react-documenteditor --save
```

YARN

```
yarn add @syncfusion/ej2-react-documenteditor
```

Import Syncfusion CSS styles

Syncfusion React components come with [built-in themes](#), which are available in the installed packages. It's easy to adapt the Syncfusion React components to match the style of your application by referring to one of the built-in themes.

Import the **Material** theme into the `src/app/globals.css` file and removed the existing styles in that file, as shown below:

GLOBALS.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-navigations/styles/material.css';
@import '../node_modules/@syncfusion/ej2-splitbuttons/styles/material.css';
@import '../node_modules/@syncfusion/ej2-dropdowns/styles/material.css';
@import "../node_modules/@syncfusion/ej2-react-documenteditor/styles/material.css";
```

To know more about built-in themes and CSS reference for individual components, refer to the [themes](#) section.

Add Syncfusion React component

Follow the below steps to add the React Document Editor component to the Next.js project:

1. Before adding the Document Editor component to your markup, import the Document Editor component in the **src/app/page.tsx** file.

PAGE.TSX

```
'use client'
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
```

2. Then, define the Document Editor component in the **src/app/page.tsx** file, as shown below:

PAGE.TSX

```
'use client'
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
export default function Home() {
  return (
    <>
    <h2>Syncfusion React Document Editor Component</h2>
    <DocumentEditorContainerComponent id="container" height={ '590px' }
    serviceUrl="https://services.syncfusion.com/react/production/api/documenteditor/" enableToolbar={true}>
    </DocumentEditorContainerComponent>
    </>
  )
}
```

Run the application

To run the application, use the following command:

NPM

```
npm run dev
```

YARN

```
yarn run dev
```

To learn more about the functionality of the Document Editor component, refer to the [documentation](#).

[View the NEXT.js Document Editor sample in the GitHub repository.](#)

Feature module in React Document editor component

Document Editor features are segregated into individual feature-wise modules to enable selective referencing. By default, the document editor displays the document in read-only mode. The required modules should be injected to extend its functionality. The following are the selective modules of document editor that can be included as required:

- **Print** - Prints the document.
- **SfdtExport** - Exports the document as Syncfusion Document Text (.SFDt) file.
- **Selection** - Selects a portion of the document and copy it to the clipboard.

- **Search** - Searches specific text and navigate between the results.
- **WordExport** - Exports the document as Word Document (.DOCX) file.
- **TextExport** - Exports the document as Text Document (.TXT) file.
- **Editor** - Performs all kind of editing operations.
- **EditorHistory** - Maintains the history of editing operations so that you can perform undo and redo at any time.
- User interface options such as context menu, options pane, image resizer, and dialog are available as individual modules.

In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a document editor instance.

Refer to the following table.

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

Module	Dependent modules to be injected for extending the functionality of document editor in your application	Property to enable the functionality for a document editor instance
--------	---	---

```
|ImageResizer| DocumentEditor.Inject(Selection, Editor,
ImageResizer)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableImageResizer= {true }>|
```

```
|HyperlinkDialog| DocumentEditor.Inject(Selection, Editor,
HyperlinkDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableHyperlinkDialog= {true }>|
```

```
|TableDialog| DocumentEditor.Inject(Selection, Editor,
TableDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableTableDialog= {true }>|
```

```
|FontDialog| DocumentEditor.Inject(Selection, Editor, FontDialog)|<DocumentEditorComponent
isReadOnly= {false} enableEditor= {true} enableFontDialog= {true }>|
```

```
|ParagraphDialog| DocumentEditor.Inject(Selection, Editor,
ParagraphDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableParagraphDialog= {true }>|
```

```
|BookmarkDialog| DocumentEditor.Inject(Selection, Editor,
BookmarkDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableBookmarkDialog= {true }>|
```

```
|PageSetupDialog| DocumentEditor.Inject(Selection, Editor,
PageSetupDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enablePageSetupDialog= {true }>|
```

```
|TableOfContentsDialog| DocumentEditor.Inject(Selection, Editor,
TableOfContentsDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableTableOfContentsDialog= {true }>|
```

```
|ListDialog| DocumentEditor.Inject(Selection, Editor, ListDialog)|<DocumentEditorComponent
isReadOnly= {false} enableEditor= {true} enableListDialog= {true }>|
```

```
|TablePropertiesDialog| DocumentEditor.Inject(Selection, Editor,
TablePropertiesDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableTablePropertiesDialog= {true }>|
```

```
|BordersAndShadingDialog| DocumentEditor.Inject(Selection, Editor,
BordersAndShadingDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor=
{true} enableBordersAndShadingDialog= {true }>|
```

```
|TableOptionsDialog| DocumentEditor.Inject(Selection, Editor,
TableOptionsDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableTableOptionsDialog= {true }>|
```

```
|StylesDialog| DocumentEditor.Inject(Selection, Editor,
StylesDialog,StyleDialog)|<DocumentEditorComponent isReadOnly= {false} enableEditor= {true}
enableStyleDialog= {true ,enableStylesDialog= {true }>|
```

```
|StyleDialog| DocumentEditor.Inject(Selection, Editor, StyleDialog)|<DocumentEditorComponent
isReadOnly= {false} enableEditor= {true} enableStyleDialog= {true }>|
```

These modules should be injected into the documenteditor using the `Inject` directive.

Accessibility in React Document editor component

The accessibility compliance for the Document editor component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

Keyboard interaction

Document editor supports [keyboard shortcuts](#).

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Import in React Document editor component

In Document Editor, the documents are stored in its own format called **Syncfusion Document Text (SFDT)**.

The following example shows how to open SFDT data in Document Editor.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-
documenteditor';
function App() {
  let documenteditor;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function componentDidMount() {
    let sfdt = `{
      "sections": [
        {
          "blocks": [
            {
              "inlines": [
                {
                  "characterFormat": {
                    "bold": true,
                    "italic": true
                  },
                  "text": "Hello World"
                }
              ]
            }
          ]
        }
      ],
      "headersFooters": {
      }
    }`;
    setTimeout(() => {
```

```

        //Open the document in Document Editor.
        documenteditor.open(sfdt);
    });
}
    return (<DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
    DocumentEditorComponent, DocumentEditor
} from '@syncfusion/ej2-react-documenteditor';
function App() {
    let documenteditor: DocumentEditorComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function componentDidMount() {
        let sfdt: string = `{
            "sections": [
                {
                    "blocks": [
                        {
                            "inlines": [
                                {
                                    "characterFormat": {
                                        "bold": true,
                                        "italic": true
                                    },
                                    "text": "Hello World"
                                }
                            ]
                        }
                    ]
                },
                "headersFooters": {
                }
            }
        }`;
        setTimeout(() => {
            //Open the document in Document Editor.
            documenteditor.open(sfdt);
        });
    }
    return (
        <DocumentEditorComponent id="container" height={'330px'} ref={(scope)
=> { documenteditor = scope; }} />
    );
}

```



```
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Import document from local machine

The following example shows how to import document from local machine.

INDEX.JSX

```
{% raw %}
```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, } from '@syncfusion/ej2-react-documenteditor';
function App() {
  let documenteditor;
  function onImportClick() {
    //Open file picker.
    document.getElementById('file_upload').click();
  }
  function onFileChange(e) {
    if (e.target.files[0]) {
      //Get selected file.
      let file = e.target.files[0];
      //Open sfdt document.
      if (file.name.substr(file.name.lastIndexOf('.')) === '.sfdt') {
        let fileReader = new FileReader();
        fileReader.onload = (e) => {
          let contents = e.target.result;
          let proxy = documenteditor;
          //Open the document in Document Editor.
          proxy.open(contents);
        };
        //Read the file as text.
        fileReader.readAsText(file);
        documenteditor.documentName = file.name.substr(0,
file.name.lastIndexOf('.'));
      }
    }
  }
  return (<div>
    <input type='file' id='file_upload' accept='.sfdt'
onChange={onFileChange}/>
    <button onClick={onImportClick}>Import</button>
    <DocumentEditorComponent id="container" ref={scope => {
      documenteditor = scope;
    }} height={'330px'}/>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent,
  DocumentEditor,
} from '@syncfusion/ej2-react-documenteditor';
function App() {
  let documenteditor: DocumentEditorComponent;
  function onImportClick() {
    //Open file picker.

```

```

        document.getElementById('file_upload').click();
    }
    function onFileChange(e: any) {
        if (e.target.files[0]) {
            //Get selected file.
            let file = e.target.files[0];
            //Open sfdt document.
            if (file.name.substr(file.name.lastIndexOf('.')) === '.sfdt') {
                let fileReader: FileReader = new FileReader();
                fileReader.onload = (e: any) => {
                    let contents: string = e.target.result;
                    let proxy: any = documenteditor;
                    //Open the document in Document Editor.
                    proxy.open(contents);
                };
                //Read the file as text.
                fileReader.readAsText(file);
                documenteditor.documentName = file.name.substr(0,
file.name.lastIndexOf('.'));
            }
        }
    }
    return (
        <div>
            <input type='file' id='file_upload' accept='.sfdt'
onChange={onFileChange} />
            <button onClick={onImportClick}>Import</button>
            <DocumentEditorComponent
                id="container"
                ref={scope => {
                    documenteditor = scope;
                }}
                height={'330px'}
            />
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Convert word documents into SFDT

You can convert word documents into SFDT format using the .NET Standard library

[Syncfusion.EJ2.WordEditor.AspNet.Core](#) by the web API service implementation. This library helps you to convert word documents (.dotx,.docx,.docm,.dot,.doc), rich text format documents (.rtf), and text documents (.txt) into SFDT format.

Note: The Syncfusion Document Editor component's document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application. For more information about [why the document pagination \(page-by-page display\) differs from Microsoft Word](#)

Please refer the following example for converting word documents into SFDT.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-documenteditor';
function App() {
    let documenteditor: DocumentEditorComponent;
    function onImportClick() {

```

```

//Open file picker.
document.getElementById('file_upload').click();
}
function onFileChange(e: any) {
  if (e.target.files[0]) {
    //Get selected file.
    let file = e.target.files[0];
    if (file.name.substr(file.name.lastIndexOf('.')) !== '.sfdt') {
      loadFile(file);
    }
  }
}

function loadFile(file: File) {
  let ajax: XMLHttpRequest = new XMLHttpRequest();
  ajax.open('POST', 'https://localhost:4000/api/documenteditor/Import', true);
  ajax.onreadystatechange = () => {
    if (ajax.readyState === 4) {
      if (ajax.status === 200 || ajax.status === 304) {
        // open SFDT text in document editor
        documenteditor.open(ajax.responseText);
      }
    }
  };

  let formData: FormData = new FormData();
  formData.append('files', file);
  ajax.send(formData);
}

return (
  <div>
    <input type="file" id="file_upload" accept=".dotx,.docx,.docm,.dot,.doc,.rtf,.txt,.xml,.sfdt"
      onChange={onFileChange} />
    <button onClick={onImportClick}>Import</button>
    <DocumentEditorComponent id="container" height={'330px'} ref={scope => {

```

```

documenteditor = scope;
}} />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Here's how to handle the server-side action for converting word document in to SFDT.

```

`csharp
[AcceptVerbs("Post")]
public string Import(Microsoft.AspNetCore.Http.IFormCollection data)
{
    if (data.Files.Count == 0)
        return null;

    System.IO.Stream stream = new System.IO.MemoryStream();
    Microsoft.AspNetCore.Http.IFormFile file = data.Files[0];
    int index = file.FileName.LastIndexOf('.');
    string type = index > -1 && index < file.FileName.Length - 1 ?
        file.FileName.Substring(index) : ".docx";
    file.CopyTo(stream);
    stream.Position = 0;

    Syncfusion.EJ2.DocumentEditor.WordDocument document =
        Syncfusion.EJ2.DocumentEditor.WordDocument.Load(stream, GetFormatType(type.ToLower()));
    string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(document);
    document.Dispose();
    return sfdt;
}

internal static Syncfusion.EJ2.DocumentEditor.FormatType GetFormatType(string format)
{
    if (string.IsNullOrEmpty(format))
        throw new System.NotSupportedException("EJ2 DocumentEditor does not support this file format.");
    switch (format.ToLower()) {

```

```

case ".dotx":
case ".docx":
case ".docm":
case ".dotm":
return Syncfusion.EJ2.DocumentEditor.FormatType.Docx;
case ".dot":
case ".doc":
return Syncfusion.EJ2.DocumentEditor.FormatType.Doc;
case ".rtf":
return Syncfusion.EJ2.DocumentEditor.FormatType.Rtf;
case ".txt":
return Syncfusion.EJ2.DocumentEditor.FormatType.Txt;
case ".xml":
return Syncfusion.EJ2.DocumentEditor.FormatType.WordML;
default:
throw new System.NotSupportedException("EJ2 DocumentEditor does not support this file format.");
}
}
,

```

To know about server-side action, please refer this [page](#).

Compatibility with Microsoft Word

Syncfusion Document Editor is a minimal viable Word document viewer/editor product for web applications. As most compatible Word editor, the product vision is adding valuable feature sets of Microsoft Word, and not to cover 100% feature sets of Microsoft Word desktop application. You can even see the feature sets difference between Microsoft Word desktop and their Word online application. So kindly don't misunderstand this component as a complete replacement for Microsoft Word desktop application and expect 100% feature sets of it.

How Syncfusion accepts the feature request for Document Editor

Syncfusion accepts new feature request as valid based on feature value and technological feasibility, then plan to implement unsupported features incrementally in future releases in a phase-by-phase manner.

How to report the problems in Document Editor

You can report the problems with displaying, or editing Word documents in Document Editor component through [support forum](#), [Direct-Trac](#), or [feedback portal](#). Kindly share the Word document for replicating the problem easily in minimal time. If you have confidential data, you can replace it and attach the document.

Why the document pagination differs from Microsoft Word

For your understanding about the Word document structure and the workflow of Word viewer/editor components, the Word document is a flow document in which content will not be preserved page by page; instead, the content will be preserved sequentially like a HTML file. Only the Word viewer/editor paginates the content of the Word document page by page dynamically, when opened for viewing or editing and this page-wise position information will not be preserved in the document level (it is Word file format specification standard). Syncfusion Document Editor component also does the same.

At present there is a known technical limitation related to slight difference in text size calculated using HTML element based text measuring approach. Even though the text size is calculated with correct font and font size values, the difference lies; it is as low as 0.00XX to 0. XXXX values compared to that of Microsoft Word application's display. Hence the document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application.

How Syncfusion address the document pagination difference compared to Microsoft Word

The following table illustrates the reasons for pagination (page-by-page display) difference compared to Microsoft Word in your documents and how Syncfusion address it.

Root causes	How is it solved?
Any mistake (wrong behavior handled) in lay outing the supported elements and formatting	Customer can report to Syncfusion support and track the status through bug report link. Syncfusion fixes the bugs in next possible weekly patch release and service pack or main releases.
Font missing in deployment environment	Customer can either report to Syncfusion support and get suggestion or solve it on their own by installing the missing fonts in their deployment environment.
Any unsupported elements or formatting present in your document	Customer can report to Syncfusion support and track the status through feature request link. Syncfusion implements unsupported features incrementally in future releases based on feature importance, customer interest, efforts involved, and technological feasibility. Also, suggests alternate approach for possible cases.
Technical limitation related to framework	For example, there is a known case with slight fractional difference in text size measured using HTML and Microsoft Word's display. Customer can report to Syncfusion support and track the status through feature request link. Syncfusion does research about alternate approaches to overcome the technical limitation/behaviors and process it same as a feature.
>Note: Here the challenge is, time schedule for implementation varies based on the alternate solution and its reliability.	

See Also

- [Feature modules](#)

Export in React Document editor component

Document Editor exports the document into various known file formats in client-side such as Microsoft Word document (.docx), Microsoft Word Template (.dotx), text document (.txt), and its own format called **Syncfusion Document Text (.sfdt)**.

We are providing two types of save APIs as mentioned below.

API name	Purpose

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

|save(filename,FormatType):void
FormatType: Sfdt or Docx or Txt|Creates the document with specified file name and format type. Then, the created file is downloaded in the client browser by default.

|saveAsBlob(FormatType):Blob|Creates the document in specified format type and returns the created document as Blob.
This blob can be uploaded to your required server, database, or file path.

SFDT export

The following example shows how to export documents in document editor as Syncfusion document text (.sfdt).

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor } from
 '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
function App() {
  let documenteditor;
  return (<div>
    <button onClick={save}>Save</button>
    <DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
    enableSelection={true} enableEditor={true} enableSfdtExport={true}/>
    </div>);
  function save() {
    //Download the document in sfdt format.
    documenteditor.save('sample', 'Sfdt');
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor } from
 '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
function App() {
  let documenteditor: DocumentEditorComponent;
  return (
    <div>
      <button onClick={save}>Save</button>
      <DocumentEditorComponent id="container" height={'330px'}
      ref={(scope) => {documenteditor = scope; }} isReadOnly={false}
      enableSelection={true} enableEditor={true} enableSfdtExport={true} />
    </div>
  );
}
```

```

        </div>
    );
    function save(){
        //Download the document in sfdt format.
        documenteditor.save('sample', 'Sfdt');
    }
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Document Editor features are segregated into individual feature-wise modules. To use SFDT export, inject the `SfdtExport` module using `DocumentEditor.Inject(SfdtExport)`.

To enable SFDT export for a document editor instance, set `enableSfdtExport` to true.

Word export

The following example shows how to export the document as Word document (.docx).

Note: The Syncfusion Document Editor component's document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application. For more information about [why the document pagination \(page-by-page display\) differs from Microsoft Word](#)

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, WordExport, SfdtExport, Selection, Editor }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, WordExport);
function App() {
  let documenteditor;
  function save() {
    //Download the document in docx format.
    documenteditor.save('sample', 'Docx');
  }
  return (<div>
    <button onClick={save}>Save</button>
    <DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
    enableSelection={true} enableEditor={true} enableSfdtExport={true}
    enableWordExport={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, WordExport, SfdtExport, Selection, Editor }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, WordExport);
function App() {
  let documenteditor: DocumentEditorComponent;
  function save() {
    //Download the document in docx format.

```

```

        documenteditor.save('sample', 'Docx');
    }
    return (
        <div>
            <button onClick={save}>Save</button>
            <DocumentEditorComponent id="container" height={'330px'}
ref={(scope) => {documenteditor = scope; }} isReadOnly={false}
enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableWordExport={true} />
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>

```

```

        <div id='sample'>
            <div id='loader'>Loading....</div>
        </div>
    </body>
</html>

```

Document Editor features are segregated into individual feature-wise modules. To use word export, inject the **WordExport** and **SfdtExport** modules using **DocumentEditor.Inject(WordExport, SfdtExport)**.

To enable word export for a document editor instance, set **enableWordExport** to true.

Template export

The following example shows how to export the document as Word Template (.dotx).

Note: The Syncfusion Document Editor component's document pagination (page-by-page display) can't be guaranteed for all the Word documents to match the pagination of Microsoft Word application. For more information about [why the document pagination \(page-by-page display\) differs from Microsoft Word](#)

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor, WordExport }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, WordExport);
function App() {
    let documenteditor;
    function save() {
        //Download the document in txt format.
        documenteditor.save('sample', 'Dotx');
    }
    return (<div>
        <button onClick={save}>Save</button>
        <DocumentEditorComponent id="container" height={'330px'}
        ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
        enableSelection={true} enableEditor={true} enableSfdtExport={true}
        enableWordExport={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';

```

```
import { DocumentEditorComponent, SfdtExport, Selection, Editor, WordExport }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, WordExport);
function App() {
  let documenteditor: DocumentEditorComponent;
  function save() {
    //Download the document in txt format.
    documenteditor.save('sample', 'Dotx');
  }
  return (
    <div>
      <button onClick={save}>Save</button>
      <DocumentEditorComponent id="container" height={'330px'}
ref={(scope) => {documenteditor = scope; }} isReadOnly={false}
enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableWordExport={true} />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
```

```

<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Document Editor features are segregated into individual feature-wise modules. To use word template export, inject the **WordExport** and **SfdtExport** modules using **DocumentEditor.Inject(WordExport, SfdtExport)**.

To enable word template export for a document editor instance, set **enableWordExport** to true.

Text export

The following example shows how to export document as text document (.txt).

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor, TextExport }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, TextExport);
function App() {
  let documenteditor;
  function save() {
    //Download the document in txt format.
    documenteditor.save('sample', 'Txt');
  }
  return (<div>
    <button onClick={save}>Save</button>
    <DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
    enableSelection={true} enableEditor={true} enableSfdtExport={true}
    enableTextExport={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}

```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor, TextExport }
from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, TextExport);
function App() {
    let documenteditor: DocumentEditorComponent;
    function save() {
        //Download the document in txt format.
        documenteditor.save('sample', 'Txt');
    }
    return (
        <div>
            <button onClick={save}>Save</button>
            <DocumentEditorComponent id="container" height={'330px'}
ref={ (scope) => { documenteditor = scope; } } isReadOnly={false}
enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableTextExport={true} />
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />

```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Document Editor features are segregated into individual feature-wise modules. To use text export, inject the `TextExport` and `SfdtExport` modules using the `DocumentEditor.Inject(TextExport, SfdtExport)`.

To enable text export for a document editor instance, set `enableTextExport` to true.

Export as blob

Document Editor also supports API to store the document into a blob. Refer to the following sample to export document into blob in client-side.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, WordExport, SfdtExport } from '@syncfusion/ej2-react-
documenteditor';
```

```
//Inject require modules.
```

```
DocumentEditorComponent.Inject(WordExport, SfdtExport);
```

```
function App() {
```

```
let documenteditor: DocumentEditorComponent;
```

```
function save() {
```

```
//Export the document as Blob object.
```

```
documenteditor.saveAsBlob('Docx').then((exportedDocument: Blob) => {
```

```
// The blob can be processed further
```

```
});
```

```
}
```

```
return (
```

```
<div>
```

```

<button onClick={save}>Save</button>

<DocumentEditorComponent id="container" height={'330px'} ref={{scope} => {documenteditor = scope;
}} enableWordExport={true} enableSfdtExport={true} enableTextExport={true} />
</div>

);
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

For instance, to export the document as Rich Text Format file, implement an ASP.NET MVC web API controller using DocIO library by passing the DOCX blob. Refer to the following code example.

```

`csharp
//API controller for the conversion.
[HttpPost]
public HttpResponseMessage ExportAsRtf()
{
    System.Web.HttpPostedFile data = HttpContext.Current.Request.Files[0];
    //Opens document stream
    WordDocument wordDocument = new WordDocument(data.InputStream);
    MemoryStream stream = new MemoryStream();
    //Converts document stream as RTF
    wordDocument.Save(stream, FormatType.Rtf);
    wordDocument.Close();
    stream.Position = 0;
    return new HttpResponseMessage() { Content = new StreamContent(stream) };
}
`

```

In client-side, you can consume this web service and save the document as Rich Text Format (.rtf) file. Refer to the following example.

```

`ts
onExport(): void {
    //Export the document as Blob object.
    this.documenteditor.saveAsBlob('Docx').then((exportedDocument: Blob) => {
    // The blob can be processed further

```

```
let formData: FormData = new FormData();
formData.append('fileName', 'sample.docx');
formData.append('data', exportedDocument);
this.saveAsRtf(formData);
});
}

saveAsRtf(formData: FormData): void {
let httpRequest: XMLHttpRequest = new XMLHttpRequest();
httpRequest.open('POST', '/api/DocumentEditor/ExportAsRtf', true);
httpRequest.onreadystatechange = () => {
if (httpRequest.readyState === 4) {
if (httpRequest.status === 200 || httpRequest.status === 304) {
if (!!navigator.msSaveBlob) {
navigator.msSaveBlob(httpRequest.response, 'sample.rtf');
} else {
let downloadLink: HTMLAnchorElement = document.createElementNS(
'http://www.w3.org/1999/xhtml',
'a'
) as HTMLAnchorElement;
download(
'sample.rtf',
'rtf',
httpRequest.response,
downloadLink,
'download' in downloadLink
);
}
} else {
console.error(httpRequest.response);
}
}
};
httpRequest.responseType = 'blob';
```

```
httpRequest.send(formData);
}

//Download the document in client side.
download(fileName: string, extension: string, buffer: Blob, downloadLink: HTMLAnchorElement,
hasDownloadAttribute: Boolean): void {
  if (hasDownloadAttribute) {
    downloadLink.download = fileName;
    let dataUrl: string = window.URL.createObjectURL(buffer);
    downloadLink.href = dataUrl;
    let event: MouseEvent = document.createEvent('MouseEvent');
    event.initEvent('click', true, true);
    downloadLink.dispatchEvent(event);
    setTimeout(() : void => {
      window.URL.revokeObjectURL(dataUrl);
      dataUrl = undefined;
    });
  } else {
    if (extension !== 'docx' && extension !== 'xlsx') {
      let url: string = window.URL.createObjectURL(buffer);
      let isPopupBlocked: Window = window.open(url, '_blank');
      if (!isPopupBlocked) {
        window.location.href = url;
      }
    }
  }
}
```

See Also

- [Feature modules](#)
- [How to export the document as pdf?](#)

Web services in React Document editor component

You can deploy web APIs for server-side dependencies of Document Editor component in the following platforms.

- [ASP.NET Core](#)
- [ASP.NET MVC](#)
- [Java](#)

Which operations require server-side interaction

|Operations|When client-server communication will be triggered?|What type of data will be transferred between client and server?|

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

|[Open file formats other than SFDT](#)|When opening the document other than SFDT (Syncfusion Document Editor's native file format), the server-side web API is invoked from client-side script. | **Client:** Sends the input file.
Server: Receives the input file and sends the converted SFDT back to the client.

The server-side web API internally extracts the content from the document (DOCX, DOC, WordML, RTF, HTML) using Syncfusion Word library (DocIO) and converts it into SFDT for opening the document in Document Editor. |

|[Paste with formatting](#)|When pasting the formatted content (HTML/RTF) received from system clipboard. For converting HTML/RTF to SFDT format.

 Note: Whereas plain text received from system clipboard will be pasted directly in the client-side. | **Client:** Sends the input Html or Rtf string.
Server: Receives the input Html or Rtf string and sends the converted SFDT back to the client. |

|[Restrict editing](#)|When protecting the document, for generating hash. | **Client:** Sends the input data for hashing algorithm.
Server: Receives the input data for hashing algorithm and sends the result hash information back to the client. |

|[Spellcheck](#)(default)|When the spellchecker is enabled on client-side Document Editor, and it performs the spell check validation for words in the document. | **Client:** Sends the words (string) with their language for spelling validation.
Server: Receives the words (string) with their language for spelling validation and sends the validation result as JSON back to the client. |

|[SpellCheckByPage](#)|Document editor provides options to spellcheck page by page when loading the documents. By [enabling optimized spell check](#) in client-side, you can perform spellcheck page by page when loading the documents. | **Client:** Sends the words (string) with their language for spelling validation.
Server: Receives the words (string) with their language for spelling validation and sends the validation result as JSON back to the client. |

|[Save as file formats other than SFDT and DOCX](#) (optional API)|You can configure this API, if you want to save the document in file format other than DOCX and SFDT.

 For saving the files as WordML, DOC, RTF, HTML, ODT, Text using Syncfusion Word library (DocIO) and PDF using Syncfusion Word (DocIO) and PDF libraries. |You can transfer document from client to server either as SFDT or DOCX format.

First option (SFDT):
Client: Sends the SFDT.
Server: Receives the SFDT and saves the converted document as any file format supported by [Syncfusion Word library \(DocIO\)](#) in server or sends the saved file to the client browser.

Second option (DOCX):
Client: Sends the DOCX file.
Server: Receives the DOCX file and saves the converted document as any file format supported by [Syncfusion Word library \(DocIO\)](#) in server or sends the saved file to the client browser. |

Note: If you don't require the above functionalities then you can deploy as pure client-side component without any server-side interactions.

Please refer the [example from GitHub](#) to configure the web service and set the [serviceUrl](#).

If your running web service Url is `http://localhost:62869/`, set the serviceUrl like below:

```
`ts
```

```
this.container.serviceUrl = "http://localhost:62869/api/documenteditor/";
```

```
,
```

Required Web API structure

Please check below table for expected web API structure.

Expected method name	Parameters	Return type
Import	Files(IFormCollection)	json(sfdt format)
SystemClipboard	CustomerParameter: content(type string either rtf or html) and type(either .rtf or .html)	json(sfdt format)
RestrictEditing	Parameter of type CustomRestrictParameter	public class CustomRestrictParameter
SpellCheck(default)	Parameter: SpellCheckJsonData	public class SpellCheckJsonData
SpellCheckByPage	Parameter: SpellCheckJsonData	public class SpellCheckJsonData
Save(optional API)	parameter: SaveParameter	public class SaveParameter
ExportSFDt(optional API)	parameter: SaveParameter	public class SaveParameter
Export(optional API)	Files(IFormCollection)	FileStreamResult (to save the document in client-side)

```
public class CustomRestrictParameter
{
    public string passwordBase64 { get; set; }
    public string saltBase64 { get; set; }
    public int spinCount { get; set; }
    public string resultHash { get; set; }
}
```

```
public class SpellCheckJsonData
{
    public int LanguageID { get; set; }
    public string TexttoCheck { get; set; }
    public bool CheckSpelling { get; set; }
    public bool CheckSuggestion { get; set; }
    public bool AddWord { get; set; }
}
```

```
public class SpellCheckJsonData
{
    public int LanguageID { get; set; }
    public string TexttoCheck { get; set; }
    public bool CheckSpelling { get; set; }
    public bool CheckSuggestion { get; set; }
    public bool AddWord { get; set; }
}
```

Note: Document editor provides options to spellcheck page by page when loading the documents. By [enabling optimized spell check](#) in client-side, you can perform spellcheck page by page when loading the documents.

```
public class SaveParameter
{
    public string Content { get; set; }
    public string FileName { get; set; }
}
```

```
public class SaveParameter
{
    public string Content { get; set; }
    public string FileName { get; set; }
}
```

Customize the expected method name

Document editor component provides an option to customize the expected method name for Import, SystemClipboard, RestrictEditing and SpellCheck using [serverActionSettings](#).

The following example code illustrates how to customize the method name using serverActionSettings.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  // Customize the API name
  let settings = { import: 'Import1', systemClipboard: 'SystemClipboard1', spellCheck: 'SpellCheck1',
    restrictEditing: 'RestrictEditing1' };
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      serverActionSettings={settings}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));

```

Add the custom headers to XMLHttpRequest

Document editor component provides an option to add custom headers of XMLHttpRequest using the [headers](#).

```

`ts

```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  // custom headers
  let customHeaders = [{ 'Authorization': 'Bearer YOURACCESSTOKEN' }, { 'Content-Type': 'application/json' }
  ];
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      headers = {customHeaders}
    />
  );
}
export default App
ReactDOM.render(<App />, document.getElementById('root'));

```

Modify the XMLHttpRequest before request send

Document editor component provides an option to modify the XMLHttpRequest object (setting additional headers, if needed) using [beforeXmlHttpRequestSend](#) event and it gets triggered before a server request.

You can customize the required [XMLHttpRequest](#) properties.

The following example code illustrates how to modify the XMLHttpRequest using [beforeXmlHttpRequestSend](#).


```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
  XmlHttpRequestEventArgs,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    // Below action, cancel all server-side interactions expect spell check
    container.beforeXmlHttpRequestSend = (
      args: XmlHttpRequestEventArgs
    ): void => {
      //Here, modifying the request headers
      args.headers = [{ syncfusion: 'true' }];
      args.withCredentials = true;
      switch (args.serverActionType) {
        case 'Import':
        case 'RestrictEditing':
        case 'SystemClipboard':
          args.cancel = true;
          break;
      }
    };
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
    />
  );
}
```

```

}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
}
export default App
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Note: Find the customizable serverActionType values are `'Import'` | `'RestrictEditing'` | `'SpellCheck'` | `'SystemClipboard'`.

Server Deployment

Word processor server docker image overview in React Document editor component

The Syncfusion **Word Processor (also known as Document Editor)** is a component with editing capabilities like Microsoft Word. It is used to create, edit, view, and print Word documents. It provides all the common word processing abilities, including editing text; formatting contents; resizing images and tables; finding and replacing text; importing, exporting, and printing Word documents; and using bookmarks and tables of contents.

This Docker image is the predefined Docker container of Syncfusion's Word Processor backend. You can deploy it quickly to your infrastructure.

Word Processor is a commercial product, and it requires a valid license to use it in a production environment ([request license or trial key](#)).

The Word Processor is supported in the JavaScript, Angular, React, Vue, ASP.NET Core, ASP.NET MVC, and Blazor platforms.

Prerequisites

Have [Docker](#) installed in your environment:

- On Windows, install [Docker for Windows](#).
- On macOS, install [Docker for Mac](#).

How to deploy Word Processor Docker image

Step 1: Pull the word-processor-server image from Docker Hub.

```
docker pull syncfusion/word-processor-server
```

Step 2: Create the docker-compose.yml file with the following code in your file system.

```

version: '3.4'
services:
  word-processor-server:
    image: syncfusion/word-processor-server:latest
    environment:
      Provide your license key for activation
      SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
    ports:
      • "6002:80"

```

Step 3: In a terminal tab, navigate to the directory where you've placed the docker-compose.yml file and execute the following.

```
docker-compose up
```

Now the Word Processor server Docker instance runs in the localhost with the provided port number `http://localhost:6002`. Open this link in a browser and navigate to the Word Processor Web API control `http://localhost:6002/api/documenteditor`. It returns the default get method response.

Step 4: Append the Docker instance running the URL (`http://localhost:6002/api/documenteditor`) to the service URL in the client-side Word Processor control. For more information about how to get started with the Word Processor control, refer to this [getting started page](#).

[How to configure spell checker dictionaries path in Docker compose file](#)

Step 1: In the Docker compose file, mount the local directory as a container volume using the following code.

```

version: '3.4'
services:
  word-processor-server:
    image: syncfusion/word-processor-server:latest
    environment:
      Provide your license key for activation
      SYNCFUSIONLICENSEKEY: YOURLICENSEKEY
    volumes:

```

- `./data:/app/data`

ports:

- `"6002:80"`

,

This YAML definition binds the data folder that is available in the Docker compose file directory.

Step 2: In the data folder, include the dictionary files (.dic, .aff) and JSON file. The JSON file should contain the language based dictionary file configuration in the following format.

,

```
[
{
  "LanguadeID": 1036,
  "DictionaryPath": "fr_FR.dic",
  "AffixPath": "fr_FR.aff",
  "PersonalDictPath": "customDict.dic"
},
{
  "LanguadeID": 1033,
  "DictionaryPath": "en_US.dic",
  "AffixPath": "en_US.aff",
  "PersonalDictPath": "customDict.dic"
}
]
```

,

Note: By default, the json file name should be "spellcheck.json". You can also use different file name by mounting the file name to 'SPELLCHECKJSONFILENAME' attribute in Docker compose file as below,

,

version: '3.4'

services:

word-processor-server:

image: syncfusion/word-processor-server:latest

environment:

[Provide your license key for activation](#)

`SYNCFUSIONLICENSEKEY: YOURLICENSEKEY`

SPELLCHECKDICTIONARYPATH: data

SPELLCHECKJSONFILENAME: spellcheck1.json

volumes:

- ./data:/app/data

ports:

- "6002:80"

,

Step 3: For handling the personal dictionary, place an empty .dic file (e.g., customDict.dic file) in the data folder.

Step 4: Provide the configured volume path to the environment variable like in the following in the Docker compose file.

,

version: '3.4'

services:

word-processor-server:

image: syncfusion/word-processo -server:latest

environment:

[Provide your license key for activation](#)

SYNCFUSIONLICENSEKEY: YOURLICENSEKEY

SPELLCHECKDICTIONARYPATH: data

volumes:

- ./data:/app/data

ports:

- "6002:80"

,

[How to copy template Word documents to Docker image](#)

You can copy the required template Word documents into docker container while deploying the docker image to server. You can open these Word documents present in the server by passing the document path (name with relative path) to LoadDocument() web API.

Note: Place the word files in the data folder mentioned in the volumes section(i.e., C:/Docker/Data) of the docker-compose.yml file. All the files present in the folder path (C:/Docker/Data) mentioned in the volumes section of 'docker-compose.yml' file will be copied to the respective folder (/app/Data) of

docker container. The Word documents copied to docker container can be processed using the 'LoadDocument' web API.

The following code example shows how to use LoadDocument() API in document editor.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-documenteditor';
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function componentDidMount() {
    var dataContext = documenteditor;
    var uploadDocument = new FormData();
    uploadDocument.append('DocumentName', 'Getting Started.docx');
    var baseUrl = 'http://localhost:6002/api/documenteditor/LoadDocument';
    var httpRequest = new XMLHttpRequest();
    httpRequest.open('POST', baseUrl, true);
    httpRequest.onreadystatechange = () => {
      if (httpRequest.readyState === 4) {
        if (httpRequest.status === 200 || httpRequest.status === 304) {
          //Open the document in Document Editor.
          dataContext.open(httpRequest.responseText);
        }
      }
    };
    httpRequest.send(uploadDocument);
  }
  return (
    <DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor = scope;
    }} />
  );
}
```

```
export default App;  
ReactDOM.render(<App />, document.getElementById('sample'));  
`
```

Refer to these getting started pages to create a Word Processor in [Typescript](#), [Angular](#), [Vue](#), [ASP.NET MVC](#), [ASP.NET Core](#), and [Blazor](#).

How to deploy word processor server docker container in azure app service in React Document editor component

Prerequisites

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

```
`  
az login  
`
```

Step 1: Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named documenteditorresourcegroup in the eastus location.

```
`  
az group create --name documenteditorresourcegroup --location "East US"  
`
```

Step 2: Create an Azure App Service plan.

Create an App Service plan in the resource group with the [az appservice plan create](#) command.

The following example creates an App Service plan named documenteditorappservice in the Standard pricing tier (--sku S1) and in a Linux container (--is-linux).

```
`  
az appservice plan create --name documenteditorappservice --resource-group  
documenteditorresourcegroup --sku S1 --is-linux  
`
```

Step 3: Create a Docker Compose app.

Create a multi-container [web app](#) in the documenteditorappservice App Service plan with the [az webapp create](#) command. The following command creates the web app using the provided Docker compose file. Please look into the section for getting started with Docker compose to create the Docker compose file for the document editor server and use the created Docker compose file here.

```
az webapp create --resource-group documenteditorresourcegroup --plan documenteditorappservice --
name documenteditor-server --multicontainer-config-type compose --multicontainer-config-file
documenteditor-server-compose.yml
```

Step 4: Browse to the app.

Browse to the deployed app at `http://<app_name>.azurewebsites.net`, i.e. `http://documenteditor-server.azurewebsites.net`. Browse this link and navigate to the Document Editor Web API control `http://documenteditor-server.azurewebsites.net/api/documenteditor`. It returns the default get method response.

Append the app service running the URL `http://documenteditor-server.azurewebsites.net/api/documenteditor/` to the service URL in the client-side Document Editor control. For more information about the Document Editor control, refer to this [getting started page](#).

For more information about the app container service, please look deeper into the [Microsoft Azure Container Service](#) for a production-ready setup.

How to deploy word processor server docker container in azure kubernetes service in React Document editor component

Prerequisites

- Have [Azure account](#) and [Azure CLI](#) setup in your environment.
- Run the following command to open the Azure login page. Sign into your [Microsoft Azure account](#).

```
az login
```

Step 1: Create a resource group.

Create a resource group using the [az group create](#) command.

The following example creates a resource group named `documenteditorresourcegroup` in the `eastus` location.

```
az group create --name documenteditorresourcegroup --location "East US"
```

Step 2: Create AKS cluster.

Use the [az aks create](#) command to create an AKS cluster. The following example creates a cluster named `documenteditorcluster` with one node.

```
az aks create --resource-group documenteditorresourcegroup --name documenteditorcluster --node-
count 1
```


`

Step 3: Connect to the cluster.

Install the [kubecti](#) into the workspace using the following command.

`

```
az aks install-cli
```

`

To configure kubectl to connect to your Kubernetes cluster, use the [az aks get-credentials](#) command. This command downloads credentials and configures the Kubernetes CLI to use them.

`

```
az aks get-credentials --resource-group documenteditorresourcegroup --name documenteditorcluster
```

`

Step 4: Create Kubernetes Services and Deployments

[Kubernetes Services](#) and [Deployments](#) can be configured in a file. To run the Document Editor server, you must define a Service and a Deployment documenteditorserver. To do this, create the documenteditor-server.yml file in the current directory using the following code.

`

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
labels:
```

```
app: documenteditorserver
```

```
name: documenteditorserver
```

```
spec:
```

```
replicas: 1
```

```
selector:
```

```
matchLabels:
```

```
app: documenteditorserver
```

```
strategy: {}
```

```
template:
```

```
metadata:
```

```
labels:
```

```
app: documenteditorserver
```

```
spec:
```

```
containers:
```

- image: syncfusion/word-processor-server:latest

name: documenteditorserver

ports:

- containerPort: 80

env:

- name: SYNCFUSIONLICENSEKEY

value: "YOURLICENSEKEY"

apiVersion: v1

kind: Service

metadata:

labels:

app: documenteditorserver

name: documenteditorserver

spec:

ports:

- port: 80

targetPort: 80

selector:

app: documenteditorserver

type: LoadBalancer

,

Step 5: To create all Services and Deployments needed to run the Document Editor server, execute the following.

,

kubectl create -f ./documenteditor-server.yml

,

Run the following command to get the Kubernetes cluster deployed service details and copy the external IP address of the Document Editor service.

,

kubectl get all

,

Browse the copied external IP address and navigate to the Document Editor Web API control `http://<external-ip>/api/documenteditor`. It returns the default get method response.

Step 6: Append the Kubernetes service running the URL `http://<external-ip>/api/documenteditor/` to the service URL in the client-side Document Editor control. For more information about the Document Editor control, refer to this [getting started page](#).

For more details about the Azure Kubernetes service, please look deeper into [Microsoft Azure Kubernetes Service](#) for a production-ready setup.

How to publish documenteditor web api application in azure app service from visual studio in React Document editor component

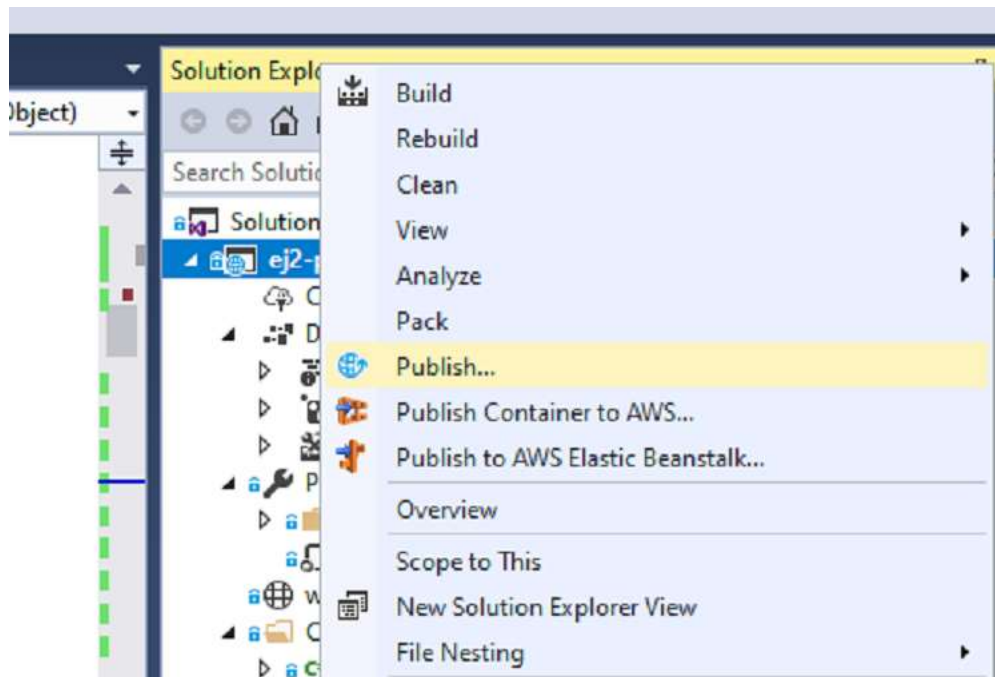
Prerequisites

- Visual Studio 2017 or 2019.
- An [Azure subscription](#).
- The Document Editor Web API controller application from [here](#).

Make sure you build the project using the Build > Build Solution menu command before following the deployment steps.

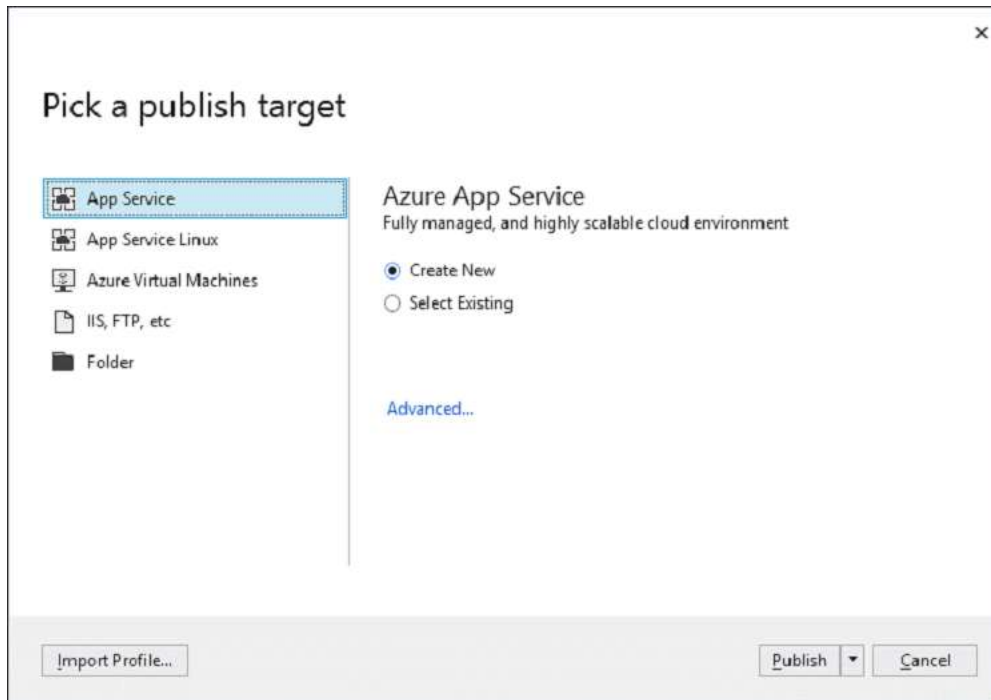
Publish to Azure App Service

Step 1: In Solution Explorer, right-click the project and click Publish (or use the Build > Publish menu item).



Step 2: If you have previously configured any publishing profiles, the Publish pane appears, in which case select Create new profile.

Step 3: In the Pick a publish target dialog box, select App Service.



Step 4: Select Publish. The Create App Service dialog box appears. Sign in with your Azure account, if necessary, and then the default app service settings populate the fields.

App Name
ej2-documenteditor-server20200514102909

Subscription
Microsoft Azure Enterprise

Resource Group
cloud-shell-storage-centralindia (centralindia) [New...](#)



Hosting Plan
ej2-documenteditor-server20200514102909P* (South Centra [New...](#)

Application Insights
None

Explore additional Azure services

- [Create a SQL Database](#)
- [Create a storage account](#)

Clicking the Create button will create the following Azure resources

- Hosting Plan - ej2-documenteditor-server202005141...  
- App Service - ej2-documenteditor-server20200514102909

Export... [Create](#) [Cancel](#)

Step 5: Select Create. Visual Studio deploys the app to your Azure App Service, and the web app loads in your browser with the app name at `http://<app_name>.azurewebsites.net` (i.e. `http://ej2-documenteditor-server20200514102909.azurewebsites.net`).

Step 6: Navigate to Document Editor Web API control `http://ej2-documenteditor-server20200514102909.azurewebsites.net/api/documenteditor`. It returns the default get method response.

Append the app service running the URL `http://ej2-documenteditor-server20200514102909.azurewebsites.net/api/documenteditor` to the service URL in the client-side Document Editor control. For more information about how to get started with the Document Editor control, refer to this [getting started page](#).

For more information about the app container service, please look deeper into the [Microsoft Azure App Service](#) for a production-ready setup.

Collaborative Editing

Allows multiple users to work on the same document simultaneously. This can be done in real-time, so that collaborators can see the changes as they are made. Collaborative editing can be a great way to improve efficiency, as it allows team members to work together on a document without having to wait for others to finish their changes.

Prerequisites

The following are needed to enable collaborative editing in Document Editor.

- SignalR
- Microsoft SQL Server

How to enable collaborative editing in client side

Step 1: Enable collaborative editing in Document Editor

To enable collaborative editing, inject `CollaborativeEditingHandler` and set the property `enableCollaborativeEditing` to true in the Document Editor, like in the code snippet below.

```
`typescript
import { DocumentEditorContainerComponent, CollaborativeEditingHandler,
DocumentEditorComponent } from '@syncfusion/ej2-react-documenteditor';

public collaborativeEditingHandler?: CollaborativeEditingHandler;

// Inject collaborative editing module.
DocumentEditor.Inject(CollaborativeEditingHandler);

public componentDidMount(): void {
  if (this.container) {
    this.container.documentEditor.enableCollaborativeEditing = true;
    this.collaborativeEditingHandler = this.container.documentEditor.collaborativeEditingHandlerModule;
  }
  if (!this.connection) {
    this.initializeSignalR();
    this.loadDocumentFromServer();
  }
}
```

```

}
render() {
return (<div className='control-pane'>
<div>
<div id='documenteditor_titlebar' className="e-de-ctn-title"></div>
<div id="documenteditorcontainerbody">
<DocumentEditorContainerComponent id="container" ref={{(scope) => { this.container = scope; }}}
style={{ 'display': 'block' }}
height={'590px'} currentUser={this.currentUser} serviceUrl={this.serviceUrl + 'api/documenteditor'}
enableToolbar={true} locale='en-US' >
<Inject services={{[Toolbar]}} />
</DocumentEditorContainerComponent>
</div>
</div>
</div>);
}
`

```

Step 2: Configure SignalR to send and receive changes

To broadcast the changes made and receive changes from remote users, configure SignalR like below.

```

`typescript
import { HubConnectionBuilder, HttpTransportType, HubConnectionState, HubConnection } from
'@microsoft/signalr';

public connectionId: string = "";
public connection?: HubConnection;

public initializeSignalR = (): void => {
// SignalR connection
this.connection = new HubConnectionBuilder().withUrl(this.serviceUrl + 'documenteditorhub', {
skipNegotiation: true,
transport: HttpTransportType.WebSockets
}).withAutomaticReconnect().build();
// Event handler for signalR connection
this.connection.on('dataReceived', this.onDataRecived.bind(this));
this.connection.onclose(async () => {
if (this.connection && this.connection.state === HubConnectionState.Disconnected) {

```

```
alert('Connection lost. Please reload the browser to continue.');
```

```
}  
});  
  
public onDataReceived(action: string, data: any) {  
  if (this.collaborativeEditingHandler) {  
    debugger;  
    if (action == 'connectionId') {  
      // Update the current connection id to track other users  
      this.connectionId = data;  
    } else if (this.connectionId != data.connectionId) {  
      if (this.titleBar) {  
        if (action == 'action' || action == 'addUser') {  
          // Add the user to title bar when user joins the room  
          this.titleBar.addUser(data);  
        } else if (action == 'removeUser') {  
          // Remove the user from title bar when user leaves the room  
          this.titleBar.removeUser(data);  
        }  
      }  
    }  
    // Apply the remote action in DocumentEditor  
    this.collaborativeEditingHandler.applyRemoteAction(action, data);  
  }  
}  
  
public connectToRoom(data: any) {  
  try {  
    if (this.connection) {  
      // start the connection.  
      this.connection.start().then(() => {  
        // Join the room.  
        if (this.connection) {  
          this.connection.send('JoinGroup', { roomName: data.roomName, currentUser: data.currentUser });  
        }  
      });  
    }  
  }  
}
```

```

}
console.log('server connected!!!');
});
}
} catch (err) {
console.log(err);
// Attempting to reconnect in 5 seconds
setTimeout(this.connectToRoom, 5000);
}
};
`

```

Step 3: Join SignalR room while opening the document

When opening a document, we need to generate a unique ID for each document. These unique IDs are then used to create rooms using SignalR, which facilitates sending and receiving data from the server.

```

`typescript
public openDocument(responseText: string, roomName: string): void {
showSpinner(document.getElementById('container') as HTMLElement);
let data = JSON.parse(responseText);
if (this.container) {
this.collaborativeEditingHandler = this.container.documentEditor.collaborativeEditingHandlerModule;
// Update the room and version information to collaborative editing handler.
this.collaborativeEditingHandler.updateRoomInfo(roomName, data.version, this.serviceUrl +
'api/CollaborativeEditing/');
// Open the document
this.container.documentEditor.open(data.sfdt);
setTimeout(() => {
if (this.container) {
// connect to server using signalR
this.connectToRoom({ action: 'connect', roomName: roomName, currentUser:
this.container.currentUser });
}
});
}
hideSpinner(document.getElementById('container') as HTMLElement);

```



```
}
`
```

Step 4: Broadcast current editing changes to remote users

Changes made on the client-side need to be sent to the server-side to broadcast them to other connected users. To send the changes made to the server, use the method shown below from the document editor using the `contentChange` event.

```
`typescript
this.container.contentChange = (args: ContainerContentChangeEventArgs) => {
if (this.collaborativeEditingHandler) {
// Send the editing action to server
this.collaborativeEditingHandler.sendActionToServer(args.operations as Operation[])
}
}
`
```

How to enable collaborative editing in ASP.NET Core

Step 1: Configure SignalR in ASP.NET Core

We are using Microsoft SignalR to broadcast the changes. Please add the following configuration to your application's Program.cs file.

```
`csharp
using Microsoft.Azure.SignalR;

.....

builder.Services.AddSignalR();

.....

.....

.....

app.MapHub<DocumentEditorHub>("/documenteditorhub");

.....

.....
`
```

Step 2: Configure SignalR hub to create room for collaborative editing session

To manage groups for each document, create a folder named "Hub" and add a file named "DocumentEditorHub.cs" inside it. Add the following code to the file to manage SignalR groups using room names.

Join the group by using unique id of the document by using `JoinGroup` method.

```
`csharp
static Dictionary<string, ActionInfo> userManager = new Dictionary<string, ActionInfo>();
```

```
internal static Dictionary<string, List<ActionInfo>> groupManager = new Dictionary<string,
List<ActionInfo>>>();

// Join to the specified room name
public async Task JoinGroup(ActionInfo info)
{
    if (!userManager.ContainsKey(Context.ConnectionId))
    {
        userManager.Add(Context.ConnectionId, info);
    }
    info.ConnectionId = Context.ConnectionId;
    //Add the current connected use to the specified group
    await Groups.AddToGroupAsync(Context.ConnectionId, info.RoomName);
    if (groupManager.ContainsKey(info.RoomName))
    {
        await Clients.Caller.SendAsync("dataReceived", "addUser", groupManager[info.RoomName]);
    }
    lock (groupManager)
    {
        if (groupManager.ContainsKey(info.RoomName))
        {
            groupManager[info.RoomName].Add(info);
        }
        else
        {
            List<ActionInfo> actions = new List<ActionInfo>
            {
                info
            };
            groupManager.Add(info.RoomName, actions);
        }
    }
    // Notify other users in the group about new user joined the collaborative editing session.
    Clients.GroupExcept(info.RoomName, Context.ConnectionId).SendAsync("dataReceived", "addUser",
    info);
}
```

```

}
,

Handle user disconnection using SignalR.

`csharp
//Handle disconnection from group.
public override Task OnDisconnectedAsync(Exception? e)
{
    string roomName = userManager[Context.ConnectionId].RoomName;
    if (groupManager.ContainsKey(roomName))
    {
        groupManager[roomName].Remove(userManager[Context.ConnectionId]);
        if (groupManager[roomName].Count == 0)
        {
            groupManager.Remove(roomName);
            //If all user disconnected from current room. Auto save the change to source document.
            CollaborativeEditingController.UpdateOperationsToSourceDocument(roomName,
            "<<documentpath>>", false);
        }
    }
    if (userManager.ContainsKey(Context.ConnectionId))
    {
        //Notify other user in the group about user exit the collaborative editing session
        Clients.OthersInGroup(roomName).SendAsync("dataReceived", "removeUser", Context.ConnectionId);
        Groups.RemoveFromGroupAsync(Context.ConnectionId, roomName);
        userManager.Remove(Context.ConnectionId);
    }
    return base.OnDisconnectedAsync(e);
}
,

```

Step 3: Configure Microsoft SQL database connection string in application level

Configure the SQL database that stores temporary data for the collaborative editing session. Provide the SQL database connection string in `appsettings.json` file.

```

`json
.....

```

```
"ConnectionStrings": {
  "DocumentEditorDatabase": "<SQL server connection string>"
}
.....
\
```

Step 4: Configure Web API actions for collaborative editing

Import File

1. When opening a document, create a database table to store temporary data for the collaborative editing session. 2. If the table already exists, retrieve the records from the table and apply them to the WordDocument instance using the `UpdateActions` method before converting it to the SFDT format.

```
`csharp
public string ImportFile([FromBody] FileInfo param)
{
  .....
  .....
  DocumentContent content = new DocumentContent();
  .....
  //Get source document from database/file system/blob storage
  WordDocument document = GetDocumentFromDatabase(param.fileName, param.documentOwner);
  .....
  //Get temporary records from database
  List<ActionInfo> actions = CreatedTable(param.fileName);
  if(actions!=null)
  {
    //Apply temporary data to the document.
    document.UpdateActions(actions);
  }
  string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
  content.version = 0;
  content.sfdt = json;
  return Newtonsoft.Json.JsonConvert.SerializeObject(content);
}
\
```

Update editing records to database.

Each edit operation made by the user is sent to the server and is pushed to the database. Each operation receives a version number after being inserted into the database.

After inserting the records to the server, the position of the current editing operation must be transformed against any previous editing operations not yet synced with the client using the TransformOperation method.

After performing the transformation, the current operation is broadcast to all connected users within the group.

```
`csharp
```

```
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
```

```
{
```

```
try
```

```
{
```

```
ActionInfo modifiedAction = AddOperationsToTable(param);
```

```
//After transformation broadcast changes to all users in the group
```

```
await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",  
modifiedAction);
```

```
return modifiedAction;
```

```
}
```

```
catch
```

```
{
```

```
return null;
```

```
}
```

```
}
```

```
private ActionInfo AddOperationsToTable(ActionInfo action)
```

```
{
```

```
int clientVersion = action.Version;
```

```
string tableName = action.RoomName;
```

```
.....
```

```
.....
```

```
.....
```

```
.....
```

```
List<ActionInfo> actions = GetOperationsQueue(table);
```

```
foreach (ActionInfo info in actions)
```

```
{
```

```

if (!info.IsTransformed)
{
    CollaborativeEditingHandler.TransformOperation(info, actions);
}
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
,

```

[Add Web API to get previous operation as a backup to get lost operations](#)

On the client side, messages broadcasted using SignalR may be received in a different order, or some operations may be missed due to network issues. In these cases, we need a backup method to retrieve missing records from the database.

Using the following method, we can retrieve all operations after the last successful client-synced version and return all missing operations to the requesting client.

```

`csharp
public async Task<ActionInfo> UpdateAction([FromBody] ActionInfo param)
{
    try
    {
        ActionInfo modifiedAction = AddOperationsToTable(param);
        //After transformation broadcast changes to all users in the group
        await _hubContext.Clients.Group(param.RoomName).SendAsync("dataReceived", "action",
            modifiedAction);
        return modifiedAction;
    }
    catch
    {
        return null;
    }
}

private ActionInfo AddOperationsToTable(ActionInfo action)
{

```

```

int clientVersion = action.Version;
string tableName = action.RoomName;
.....
.....
.....
.....
List<ActionInfo> actions = GetOperationsQueue(table);
foreach (ActionInfo info in actions)
{
if (!info.IsTransformed)
{
CollaborativeEditingHandler.TransformOperation(info, actions);
}
}
action = actions[actions.Count - 1];
action.Version = updateVersion;
//Return the transformed operation to broadcast it to other clients.
return action;
}
`

```

Full version of the code discussed about can be found in below GitHub location.

Github Example: [Collaborative editing examples](#)

Image in React Document editor component

Document Editor supports common raster format images like PNG, BMP, JPEG, SVG and GIF. You can insert an image file or online image in the document using the [insertImage\(\)](#) method. Refer to the following sample code.

```

{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, ImageResizer, EditorHistory
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, ImageResizer, EditorHistory);

```

```
function App() {
  let documenteditor: DocumentEditorComponent;

  function insertPicture() {
    let pictureUpload: HTMLInputElement = document.getElementById("insertImageButton") as
    HTMLInputElement;
    pictureUpload.value = "";
    //Open file picker.
    pictureUpload.click();
  }

  function onInsertImage(args: any) {
    let proxy: any = documenteditor;

    if (navigator.userAgent.match('Chrome') || navigator.userAgent.match('Firefox') ||
    navigator.userAgent.match('Edge') || navigator.userAgent.match('MSIE') ||
    navigator.userAgent.match('.NET')) {
      if (args.target.files[0]) {
        //Get selected file.
        let path = args.target.files[0];
        let reader = new FileReader();
        reader.onload = function (frEvent: any) {
          let base64String = frEvent.target.result;
          let image = document.createElement('img');
          image.addEventListener('load', function () {
            //Insert image in Document Editor.
            proxy.editor.insertImage(base64String, proxy.editor.width, proxy.editor.height);
          })
          image.src = base64String;
        };
        //Convert image into base64 string..
        reader.readAsDataURL(path);
      }
      //Safari does not Support FileReader Class
    } else {
      let image = document.createElement('img');
      image.addEventListener('load', function () {
```



```
//Insert image in Document Editor.
proxy.editor.insertImage(args.target.value);
})
image.src = args.target.value;
}
}
return (
<div>
<input type="file" id="insertImageButton" style={{ "position": "fixed", "left": "-110em" }}
accept=".jpg,.jpeg,.png,.bmp" onChange={onInsertImage} />
<button onClick={insertPicture}>Dialog</button>
<DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableImageResizer={true}
enableEditorHistory={true} />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
、
{% endraw %}
```

Image files will be internally converted to base64 string. Whereas, online images are preserved as URL.

Note: EMF and WMF images can't be inserted, but these types of images will be preserved in Document Editor when using ASP.NET MVC Web API.

[Image resizing](#)

Document Editor provides built-in image resizer that can be injected into your application based on the requirements. This allows you to resize the image by dragging the resizing points using mouse or touch interactions. This resizer appears as follows.



Changing size

Document Editor expose API to get or set the size of the selected image. Refer to the following sample code.

```
`ts
documenteditor.selection.imageFormat.width = 800;
documenteditor.selection.imageFormat.height = 800;
`
```

Note: Images are stored and processed(read/write) as base64 string in DocumentEditor. The online image URL is preserved as a URL in DocumentEditor upon saving.

Text wrapping style

Text wrapping refers to how images fit with surrounding text in a document. Please [refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the image

DocumentEditor preserves the position properties of the image and displays the image based on position properties. It does not support modifying the position properties. Whereas the image will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

See Also

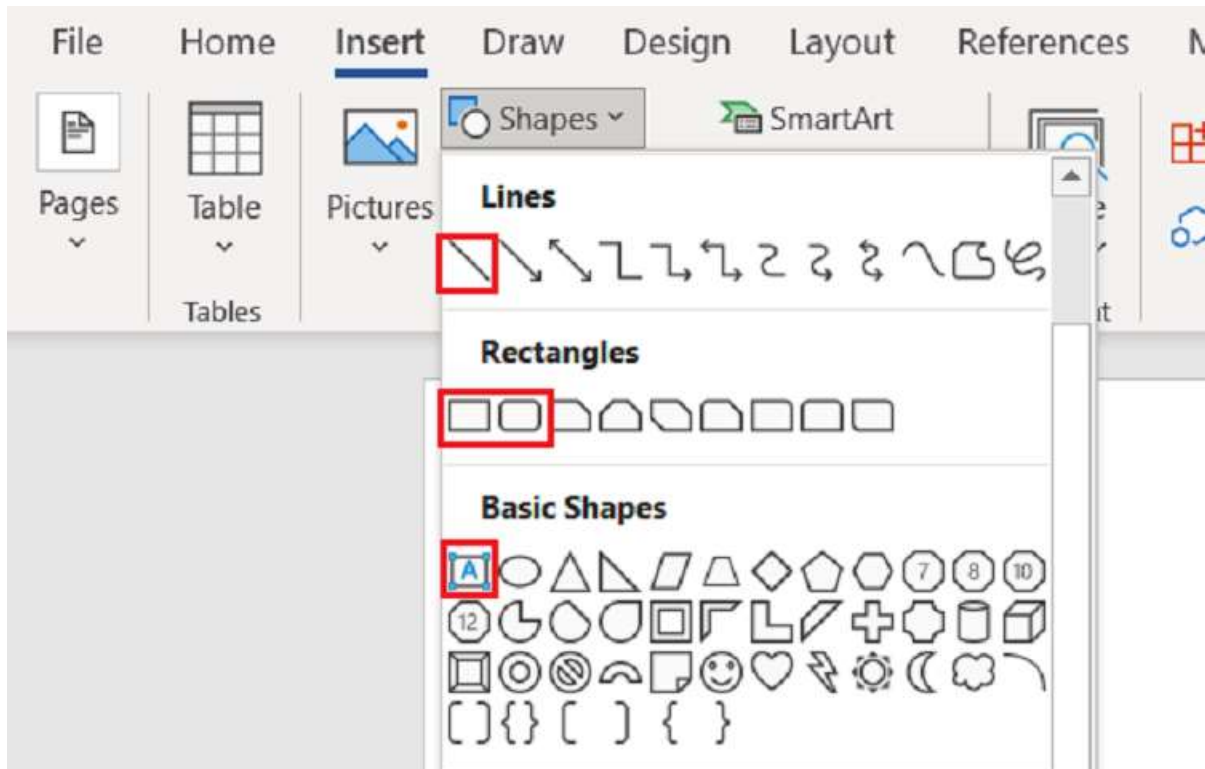
- [Feature modules](#)

Shapes in React Document editor component

Shapes are drawing objects that include a text box, rectangles, lines, curves, circles, etc. It can be preset or custom geometry. At present, Document Editor does not have support to insert shapes. however, if the document contains a shape while importing, it will be preserved properly.

Supported shapes

The Document Editor has preservation support for Text box, Rectangle, Rounded Rectangle and Line shapes.



Note: When using ASP.NET MVC service, the unsupported shapes will be converted as image and preserved as image.

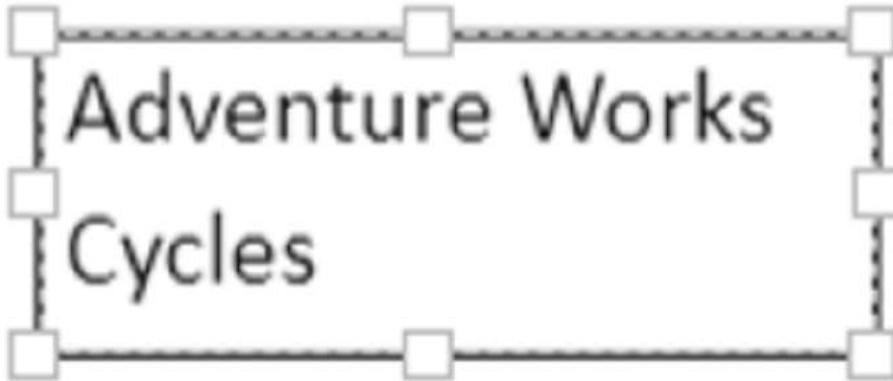
Text box Shape

A text box is a rectangular area on the document where you can enter text. When you click in a text box, a flashing cursor will display indicating that you can begin typing. It allows you to enter multiple lines of text with all text formatting.

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal **Adventure Works Cycles** and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams are located throughout their market base.

Shape Resizer

The Document Editor also supports a built-in shape resizer to resize the shapes present in the document. The shape resizer accepts both touch and mouse interactions.



Text wrapping style

Text wrapping refers to how shapes fit with surrounding text in a document. Please [refer to this page](#) for more information about text wrapping styles available in Word documents.

Positioning the shape

Document Editor preserves the position properties of the shape and displays the shape based on position properties. It does not support modifying the position properties. Whereas the shape will be automatically moved along with text edited if it is positioned relative to the line or paragraph.

Text wrapping style in React Document editor component

Text wrapping refers to how images and shapes are fit with surrounding text in a document. Currently, DocumentEditor has only preservation support for image and textbox shape with below wrapping styles.

In-Line with Text

In this option, the image or shape is placed on the same line surrounding with text like any other word or letter. This image or shape will be automatically moved along with the text while editing, whereas the other options denote that the image or shape stays in a fixed position while the text shifts and wraps around it.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company



manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in

In Front of Text

In this option, the image or shape is placed in front of the text. This can be used to place an image around some text or to add shape to highlight the part in a paragraph.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales offices are located and shipped in throughout their market base.



Note: Starting from v18.2.0.x, the in front of wrapping styles are supported.

Top and Bottom

In this option, Text wraps above and below the image or shape. No text is to the left or right of the image or shape. This can be used for larger images or shapes that occupy most of the width in a document.

Note: Starting from v19.1.0.x, the top and bottom wrapping style is supported.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company



manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290

Behind

In this option, the image or shape is placed behind the text. This can be used when you need to add a watermark or background image to a document.

Adventure Works Cycles, the fictitious company on which the AdventureWorks sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams and manufacturer and located and shipped in throughout their market base.



Note: Starting from v19.2.0.x, behind text wrapping styles are supported.

Square

In this option, Text wraps around the image or text box in a square shape.

Note: Tight and Through styles will be preserved as square wrapping style in DocumentEditor which is supported from v19.2.0.x.

Adventure Works Cycles, the fictitious company on which the Adventure Works sample databases are based, is a large, multinational manufacturing company. The company manufactures and sells metal North American, European and While its base operation is with 290 employees, several regional sales teams are located throughout their market base.

Adventure works cycles
company.

and composite bicycles to
Asian commercial markets.
located in Bothell, Washington

Bookmark in React Document editor component

Bookmark is a powerful tool that helps you to mark a place in the document to find again easily. You can enter many bookmarks in the document and give each one a unique name to identify easily.

Document Editor provides built-in dialog to add, delete, and navigate bookmarks within the document. To add a bookmark, select a portion of text in the document. After that, jump to the location or add links to it within the document using built-in hyperlink dialog. You can also delete bookmarks from a document.

Bookmark names need to begin with a letter. They can include both numbers and letters, but not spaces. To separate the words, use an underscore.

Bookmark names starting with an underscore are called hidden bookmarks. For example, bookmarks generated for table of contents.

Add bookmark

Using [insertBookmark](#) method, Bookmark can be added to the selected text.

```
`csharp
this.container.documentEditor.editor.insertBookmark("Bookmark1");
`
```

Select Bookmark

You can select the bookmark in the document using [selectBookmark](#) method by providing Bookmark name to select as shown in the following code snippet.

```
`csharp
this.container.documentEditor.selection.selectBookmark("Bookmark1", true);
`
```

Note: Second parameter is optional parameter and it denotes is exclude bookmark start and end from selection. If true, excludes bookmark start and end from selection.

Delete Bookmark

You can delete bookmark in the document using [deleteBookmark](#) method as shown in the following code snippet.

```
`csharp
this.container.documentEditor.editor.deleteBookmark("Bookmark1");
`
```

Get Bookmark from document

You can get all the bookmarks in the document using [getBookmarks](#) method as shown in the following code snippet.

```
`csharp
this.container.documentEditor.getBookmarks(false);
`
```

Note: Parameter denotes is include hidden bookmarks. If false, ignore hidden bookmark.

Get Bookmark from selection

You can get bookmarks in current selection in the document using [getBookmarks](#) method as shown in the following code snippet.

```
`csharp
this.container.documentEditor.selection.getBookmarks(false);
`
```

Replace bookmark content

You can replace bookmark content without removing the bookmark start and end for backtracking the bookmark content.

```
`csharp
```

```
this.container.documentEditor.selection.selectBookmark("Bookmark1", true);
```

```
this.container.documentEditor.editor.insertText('Hello World')
```

```
,
```

You can replace content by removing the bookmark start and end, thus the bookmark content can't be tracked in future.

```
`csharp
```

```
this.container.documentEditor.selection.selectBookmark("Bookmark1");
```

```
this.container.documentEditor.editor.insertText('Hello World')
```

```
,
```

Show or Hide bookmark

You can show or hide the show square brackets around bookmarked items in Document editor component.

The following example code illustrates how to show or hide square brackets around bookmarked items.

```
`typescript
```

```
this.container.documentEditorSettings.showBookmarks = true;
```

```
,
```

Bookmark Dialog

The following example shows how to open bookmark dialog in document editor.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, SfdtExport, Selection, Editor, BookmarkDialog } from  
'@syncfusion/ej2-react-documenteditor';
```

```
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, BookmarkDialog);
```

```
function Default() {
```

```
let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
```

```
function showBookmarkDialog() {
```

```
//Open bookmark dialog.
```

```
documenteditor.showDialog('Bookmark');
```

```
}
```

```
return (
```



```

<div>
<button onClick={showBookmarkDialog}>Dialog</button>

<DocumentEditorComponent id="container" height={'330px'} isReadOnly={false} enableSelection={true}
enableEditor={true} enableSfdtExport={true} enableBookmarkDialog={true} />
</div>

);
}

export default Default

ReactDOM.render(<Default />, document.getElementById('sample'));
`

```

See Also

- [Feature modules](#)
- [Bookmark dialog](#)

Link in React Document editor component

Document Editor supports hyperlink field. You can link a part of the document content to Internet or file location, mail address, or any text within the document.

Navigate a hyperlink

Document Editor triggers 'requestNavigate' event whenever user clicks Ctrl key or tap a hyperlink within the document. This event provides necessary details about link type, navigation URL, and local URL (if any) as arguments, and allows you to easily customize the hyperlink navigation functionality.

Add the requestNavigate event for DocumentEditor

The following example illustrates how to add requestNavigate event for DocumentEditor.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection } from
'@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfdtExport);
function App() {
  let documenteditor;
  // Add event listener for requestNavigate event to customize hyperlink
  navigation functionality
  let requestNavigate = (args) => {
    if (args.linkType !== 'Bookmark') {
      let link = args.navigationLink;
      if (args.localReference.length > 0) {
        link += '#' + args.localReference;
      }
      //Navigate to the specified URL.
      window.open(link);
      args.isHandled = true;
    }
  }
}

```



```

    };
    return (<DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} enableSelection={true}
    enableSfdtExport={true} requestNavigate={requestNavigate}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));
  {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, RequestNavigateEventArgs
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfdtExport);
function App() {
  let documenteditor: DocumentEditorComponent;
  // Add event listener for requestNavigate event to customize hyperlink
  // navigation functionality
  let requestNavigate = (args: RequestNavigateEventArgs) => {
    if (args.linkType !== 'Bookmark') {
      let link: string = args.navigationLink;
      if (args.localReference.length > 0) {
        link += '#' + args.localReference;
      }
      //Navigate to the specified URL.
      window.open(link);
      args.isHandled = true;
    }
  };
  return (
    <DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => {documenteditor = scope; }} enableSelection={true}
    enableSfdtExport={true} requestNavigate={requestNavigate} />
  );
}
export default App
ReactDOM.render(<App/>, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
  documenteditor/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Add the requestNavigate event for DocumentEditorContainer component

The following example illustrates how to add requestNavigate event for DocumentEditorContainer component.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
  RequestNavigateEventArgs
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
export class Default extends React.Component {
  onCreate() {

```

```
// Add event listener for requestNavigate event to customize hyperlink navigation functionality
this.container.documentEditor.requestNavigate = (args: RequestNavigateEventArgs) => {
  if (args.linkType !== 'Bookmark') {
    let link: string = args.navigationLink;
    if (args.localReference.length > 0) {
      link += '#' + args.localReference;
    }
    //Navigate to the selected URL.
    window.open(link);
    args.isHandled = true;
  }
};

render() {
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        this.container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={this.onCreated.bind(this)}
    />
  );
}

ReactDOM.render(<Default />, document.getElementById('sample'));
```

If the selection is in hyperlink, trigger this event by calling 'navigateHyperlink' method of 'Selection' instance. Refer to the following example.

```
`ts
```

```
documenteditor.selection.navigateHyperlink();
```

```
,
```

Copy link

Document Editor copies link text of a hyperlink field to the clipboard if the selection is in hyperlink. Refer to the following example.

```
`ts
```

```
documenteditor.selection.copyHyperlink();
```

```
,
```

Add hyperlink

To create a basic hyperlink in the document, press **ENTER** / **SPACEBAR** / **SHIFT + ENTER** / **TAB** key after typing the address, for instance <http://www.google.com>. Document Editor automatically converts this address to a hyperlink field. The text can be considered as a valid URL if it starts with any of the following.

```
`http://`<br>
```

```
`https://`<br>
```

```
file:///<br>
```

```
www.<br>
```

```
mailto:<br>
```

Refer to the following example.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfddExport, Selection, Editor } from
 '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfddExport, Editor);
function App() {
  let documenteditor;
  // Add event listener for requestNavigate event to customize hyperlink
  navigation functionality.
  let requestNavigate = (args) => {
    if (args.linkType !== 'Bookmark') {
      let link = args.navigationLink;
      if (args.localReference.length > 0) {
        link += '#' + args.localReference;
      }
      //Navigate to the specified URL.
      window.open(link);
      args.isHandled = true;
    }
  };
  return (<DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
    enableSelection={true} enableSfddExport={true} enableEditor={true}
    requestNavigate={requestNavigate}/>);
}
```

```

}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor,
  RequestNavigateEventArgs
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfdtExport, Editor);
function App() {
  let documenteditor: DocumentEditorComponent;
  // Add event listener for requestNavigate event to customize hyperlink
  // navigation functionality.
  let requestNavigate = (args: RequestNavigateEventArgs) => {
    if (args.linkType !== 'Bookmark') {
      let link: string = args.navigationLink;
      if (args.localReference.length > 0) {
        link += '#' + args.localReference;
      }
      //Navigate to the specified URL.
      window.open(link);
      args.isHandled = true;
    }
  };
  return (
    <DocumentEditorComponent id="container" height={'330px'} ref={(scope)
=> { documenteditor = scope; }} isReadOnly={false} enableSelection={true}
enableSfdtExport={true} enableEditor={true}
    requestNavigate={requestNavigate} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Customize screen tip

You can customize the screen tip text for the hyperlink by using below sample code.

```
`ts
```

```
documenteditor.insertHyperlink('https://www.google.com', 'Google', '<<Screen tip text>>');
```

```
,
```

Screen tip text can be modified through UI by using the [Hyperlink dialog](#)

Remove hyperlink

To remove link from hyperlink in the document, press Backspace key at the end of a hyperlink. By removing the link, it will be converted as plain text. You can use 'removeHyperlink' method of 'Editor' instance if the selection is in hyperlink. Refer to the following example.

```
`ts
```

```
documenteditor.editor.removeHyperlink();
```

Hyperlink dialog

Document Editor provides dialog support to insert or edit a hyperlink. Refer to the following example.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor,
HyperlinkDialog } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfdtExport, Editor,
HyperlinkDialog);
function App() {
  let documenteditor;
  //Click the hyperlink button, the hyperlink dialog will open
  function showHyperlinkDialog() {
    //Open hyperlink dialog.
    documenteditor.showDialog('Hyperlink');
  }
  return (<div>
    <button onClick={showHyperlinkDialog}>Dialog</button>
```

```

        <DocumentEditorComponent id="container" height={'330px'}
        ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
        enableSelection={true} enableSfdtExport={true} enableEditor={true}
        enableHyperlinkDialog={true}/>
      </div>;
    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('sample'));
    {% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, HyperlinkDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Selection, SfdtExport, Editor,
HyperlinkDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  //Click the hyperlink button, the hyperlink dialog will open
  function showHyperlinkDialog() {
    //Open hyperlink dialog.
    documenteditor.showDialog('Hyperlink');
  }
  return (
    <div>
      <button onClick={showHyperlinkDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'}
      ref={(scope) => {documenteditor = scope; }} isReadOnly={false}
      enableSelection={true} enableSfdtExport={true} enableEditor={true}
      enableHyperlinkDialog={true} />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />

```



```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

You can use the following keyboard shortcut to open the hyperlink dialog if the selection is in hyperlink.

Key Combination	Description
----- -----	
Ctrl + K	Open hyperlink dialog that allows you to create or edit hyperlink

See Also

- [Feature modules](#)
- [Hyperlink dialog](#)

Table in React Document editor component

Tables are an efficient way to present information. Document Editor can display and edit the tables. You can select and edit tables through keyboard, mouse, or touch interactions. Document Editor exposes a rich set of APIs to perform these operations programmatically.

Create a table

You can create and insert a table at cursor position by specifying the required number of rows and columns.

Refer to the following sample code.

```
`ts
documenteditor.editor.insertTable(3,3);
`
```

The maximum size of row and column is limited to 32767 and 63 respectively.

Insert rows

You can add a row (or several rows) above or below the row at cursor position by using the [insertRow](#) method. This method accepts the following parameters:

Parameter | Type | Description

left(optional) | boolean | This is optional and if omitted, it takes the value as false and inserts to the right of column at cursor position.

count(optional) | number | This is optional and if omitted, it takes the value as 1.

Refer to the following sample code.

```
`ts
//Insert a column to the right of the column at cursor position.
documenteditor.editor.insertColumn();
//Insert a column to the left of the column at cursor position.
documenteditor.editor.insertColumn(false);
//Insert two columns to the left of the column at cursor position.
documenteditor.editor.insertColumn(false, 2);
`
```

Select an entire table

If the cursor position is inside a table, you can select the entire table by using the following sample code.

```
`ts
documenteditor.selection.selectTable();
`
```

Select row

You can select the entire row at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectRow();
`
```

If current selection spans across cells of different rows, all these rows will be selected.

Select column

You can select the entire column at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectColumn();
`
```

,

If current selection spans across cells of different columns, all these columns will be selected.

Select cell

You can select the cell at cursor position by using the following sample code.

```
`ts
documenteditor.selection.selectCell();
```

,

Delete table

Document Editor allows you to delete the entire table. You can use the [deleteTable\(\)](#) method of editor instance, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteTable();
```

,

Delete row

Document Editor allows you to delete the selected number of rows. You can use the [deleteRow\(\)](#) method of editor instance to delete the selected number of rows, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteRow();
```

,

Delete column

Document Editor allows you to delete the selected number of columns. You can use the [deleteColumn\(\)](#) method of editor instance to delete the selected number of columns, if selection is in table. Refer to the following sample code.

```
`ts
documenteditor.editor.deleteColumn();
```

,

Merge cells

You can merge cells vertically, horizontally, or combination of both to a single cell. To vertically merge the cells, the columns within selection should be even in left and right directions. To horizontally merge the cells, the rows within selection should be even in top and bottom direction.

Refer to the following sample code.

```
`ts
documenteditor.editor.mergeCells()
```

,

Positioning the table

Document Editor preserves the position properties of the table and displays the table based on position properties. It does not support modifying the position properties. Whereas the table will be automatically moved along with text edited if it is positioned relative to the paragraph.

How to work with tables

The following sample demonstrates how to delete the table row or columns, merge cells and how to bind the API with button.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, Selection, Editor, EditorHistory, ContextMenu, TableDialog, } from
 '@syncfusion/ej2-react-documenteditor';
import { ToolbarComponent, ItemDirective, ItemsDirective, } from '@syncfusion/ej2-react-navigations';
//Inject require modules.
DocumentEditorComponent.Inject(Selection, Editor, EditorHistory, ContextMenu, TableDialog);
function App() {
let documenteditor: DocumentEditorComponent;
React.useEffect(() => {
ComponentDidMount();
}, []);
function ComponentDidMount() {
documenteditor.editor.insertTable(2, 2);
}
function toolbarButtonClick(arg) {
switch (arg.item.id) {
case 'table':
//Insert table API to add table
documenteditor.editor.insertTable(3, 2);
break;
case 'insert_above':
//Insert the specified number of rows to the table above to the row at cursor position
documenteditor.editor.insertRow(true, 2);
break;
case 'insert_below':
//Insert the specified number of rows to the table below to the row at cursor position
```

```
documenteditor.editor.insertRow();
break;
case 'insert_left':
//Insert the specified number of columns to the table left to the column at cursor position
documenteditor.editor.insertColumn(true, 2);
break;
case 'insert_right':
//Insert the specified number of columns to the table right to the column at cursor position
documenteditor.editor.insertColumn();
break;
case 'delete_table':
//Delete the entire table
documenteditor.editor.deleteTable();
break;
case 'delete_rows':
//Delete the selected number of rows
documenteditor.editor.deleteRow();
break;
case 'delete_columns':
//Delete the selected number of columns
documenteditor.editor.deleteColumn();
break;
case 'merge_cell':
//Merge the selected cells into one (both vertically and horizontally)
documenteditor.editor.mergeCells();
break;
case 'table_dialog':
//Opens insert table dialog
documenteditor.showDialog('Table');
break;
}
}
return (
```

```
<div>
<ToolbarComponent clicked={toolbarButtonClick}>
<ItemsDirective>
<ItemDirective id="table" prefixIcon="e-de-ctnr-table e-icons" />
<ItemDirective type="Separator" />
<ItemDirective id="insert_above" prefixIcon="e-de-ctnr-insertabove e-icons" />
<ItemDirective id="insert_below" prefixIcon="e-de-ctnr-insertbelow e-icons" />
<ItemDirective type="Separator" />
<ItemDirective id="insert_left" prefixIcon="e-de-ctnr-insertleft e-icons" />
<ItemDirective id="insert_right" prefixIcon="e-de-ctnr-insertright e-icons" />
<ItemDirective type="Separator" />
<ItemDirective id="delete_table" prefixIcon="e-de-delete-table e-icons" />
<ItemDirective id="delete_rows" prefixIcon="e-de-ctnr-deleterows e-icons" />
<ItemDirective id="delete_columns" prefixIcon="e-de-ctnr-deletecolumns e-icons" />
<ItemDirective type="Separator" />
<ItemDirective text="Dialog" />
</ItemsDirective>
</ToolbarComponent>
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
enableEditorHistory={true}
enableContextMenu={true}
enableTableDialog={true}
/>
</div>
);
```

```

}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

See Also

- [Feature modules](#)
- [Insert table dialog](#)

Table of contents in React Document editor component

The table of contents in a document is same as the list of chapters at the beginning of a book. It lists each heading in the document and the page number, where that heading starts with various options to customize the appearance.

Inserting table of contents

Document Editor exposes an API to insert table of contents at cursor position programmatically. You can specify the settings for table of contents explicitly. Otherwise, the default settings will be applied.

[TableOfContentsSettings](#) contain the following properties:

- **startLevel:** Specifies the start level for constructing table of contents.
- **endLevel:** Specifies the end level for constructing table of contents.
- **includeHyperlink:** Specifies whether the link for headings is included.
- **includePageNumber:** Specified whether the page number of the headings is included.
- **rightAlign:** Specifies whether the page number is right aligned.
- **tabLeader:** Specifies the tab leader styles such as none, dot, hyphen, and underscore.
- **includeOutlineLevels:** Specifies whether the outline levels are included.

The following code illustrates how to insert table of content in document editor.

```

`ts
let tocSettings: TableOfContentsSettings =
{
startLevel: 1, endLevel: 3, includeHyperlink: true, includePageNumber: true, rightAlign: true
};
this.documenteditor.editor.insertTableOfContents(tocSettings);
`

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor, } from '@syncfusion/ej2-react-
documenteditor';

```

```
//Inject require module.
```

```
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
```

```
function App() {
```

```
let documenteditor: DocumentEditorComponent;
```

```
React.useEffect(() => {
```

```
componentDidMount()
```

```
, []);
```

```
function componentDidMount() {
```

```
let documentString: string = '{"sections":[{"blocks":[{"paragraphFormat":{"styleName":"Heading 1"},"inlines":[{"text":"Headin"}, {"name":"GoBack","bookmarkType":0}, {"name":"GoBack","bookmarkType":1}, {"text":"g1"}]}, {"paragraphFormat":{"styleName":"Heading 2"},"inlines":[{"text":"Heading2"}]}, {"paragraphFormat":{"styleName":"Heading 3"},"inlines":[{"text":"Heading3"}]}, {"paragraphFormat":{"styleName":"Heading 4"},"inlines":[{"text":"Heading4"}]}, {"paragraphFormat":{"styleName":"Heading 5"},"inlines":[{"text":"Heading5"}]}, {"paragraphFormat":{"styleName":"Heading 6"},"inlines":[{"text":"Heading6"}]}, {"paragraphFormat":{"styleName":"Normal"},"inlines":[{"text":"Normal"}]}], "headersFooters": {}, "sectionFormat": {"headerDistance":36.0, "footerDistance":36.0, "pageWidth":612.0, "pageHeight":792.0, "leftMargin":72.0, "rightMargin":72.0, "topMargin":72.0, "bottomMargin":72.0, "differentFirstPage":false, "differentOddAndEvenPages":false}}, "characterFormat":{"fontSize":11.0, "fontFamily":"Calibri"}, "paragraphFormat":{"afterSpacing":8.0, "lineSpacing":1.0791666507720947, "lineSpacingType":"Multiple", "background":{"color":"#FFFFFF"}, "styles":[{"type":"Paragraph", "name":"Normal", "next":"Normal"}, {"type":"Paragraph", "name":"Heading 1", "basedOn":"Normal", "next":"Normal", "link":"Heading 1 Char", "characterFormat":{"fontSize":16.0, "fontFamily":"Calibri Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":12.0, "afterSpacing":0.0, "outlineLevel":"Level1"}}, {"type":"Paragraph", "name":"Heading 2", "basedOn":"Normal", "next":"Normal", "link":"Heading 2 Char", "characterFormat":{"fontSize":13.0, "fontFamily":"Calibri Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afterSpacing":0.0, "outlineLevel":"Level2"}}, {"type":"Paragraph", "name":"Heading 3", "basedOn":"Normal", "next":"Normal", "link":"Heading 3 Char", "characterFormat":{"fontSize":12.0, "fontFamily":"Calibri Light", "fontColor":"#1F3763FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afterSpacing":0.0, "outlineLevel":"Level3"}}, {"type":"Paragraph", "name":"Heading 4", "basedOn":"Normal", "next":"Normal", "link":"Heading 4 Char", "characterFormat":{"italic":true, "fontFamily":"Calibri Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afterSpacing":0.0, "outlineLevel":"Level4"}}, {"type":"Paragraph", "name":"Heading 5", "basedOn":"Normal", "next":"Normal", "link":"Heading 5 Char", "characterFormat":{"fontFamily":"Calibri Light", "fontColor":"#2F5496FF"}, "paragraphFormat":{"beforeSpacing":2.0, "afterSpacing":0.0, "outlineLevel":"Level5"}}, {"type":"Paragraph", "name":"Heading 6", "basedOn":"Normal", "next":"Normal", "link":"Heading 6 Char", "characterFormat":{"fontFamily":"Calibri
```



```

Light", "fontColor": "#1F3763FF"}, "paragraphFormat": {"beforeSpacing": 2.0, "afterSpacing": 0.0, "outlineLevel": "Level6"}}, {"type": "Character", "name": "Default Paragraph Font"}, {"type": "Character", "name": "Heading 1 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"fontSize": 16.0, "fontFamily": "Calibri Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 2 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"fontSize": 13.0, "fontFamily": "Calibri Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 3 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"fontSize": 12.0, "fontFamily": "Calibri Light", "fontColor": "#1F3763FF"}}, {"type": "Character", "name": "Heading 4 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 5 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496FF"}}, {"type": "Character", "name": "Heading 6 Char", "basedOn": "Default Paragraph Font", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763FF"}}}]]];

```

```
//Open the document in Document Editor.
```

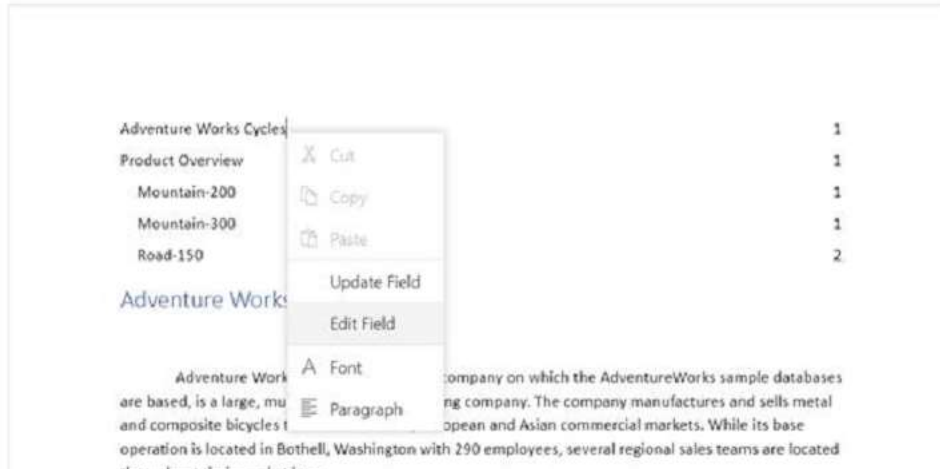
```

documenteditor.open(documentString);
}
return (
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

[Update or edit table of contents](#)

You can update or edit the table of contents using the built-in context menu shown up by right-clicking it. Refer to the following screenshot.



- **Update Field:** Updates the headings in table of contents with same settings by searching the entire document.
- **Edit Field:** Opens the built-in table of contents dialog and allows you to modify its settings.

You can also do it programmatically by using the exposed API. Refer to the following sample code.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, SfdtExport, Selection, Editor, TableOfContentsSettings, } from '@syncfusion/ej2-react-documenteditor';
```

```
//Inject require modules.
```

```
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
```

```
function App() {
```

```
let documenteditor: DocumentEditorComponent;
```

```
React.useEffect(() => {
```

```
componentDidMount()
```

```
, []);
```

```
function componentDidMount() {
```

```
//Open any existing document/
```

```
documenteditor.open("");
```

```
//Table of content settings.
```

```
let tocSettings: TableOfContentsSettings = {
```

```
startLevel: 1,
```

```
endLevel: 3,
```

```
includeHyperlink: true,
```

```

includePageNumber: true,
rightAlign: true,
};
//Insert table of content with specified settings.
documenteditor.editorModule.insertTableOfContents(tocSettings);
}
return (
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Same method is used for inserting, updating, and editing table of contents. This will work based on the current element at cursor position and the optional settings parameter. If table of contents is present at cursor position, the update operation will be done based on the optional settings parameter. Otherwise, the insert operation will be done.

[See Also](#)

- [Table of contents dialog](#)

Header footer in React Document editor component

Document Editor supports headers and footers in its document. Each section in the document can have the following types of headers and footers:

- First page: Used only on the first page of the section.
- Even pages: Used on all even numbered pages in the section.

- Default: Used on all pages of the section, where first or even pages are not applicable or not specified.

You can define this by setting format properties of the corresponding section using the following sample code.

```
`ts
//Defines whether different header footer is required for first page of the section
documenteditor.selection.sectionFormat.differentFirstPage= true;
//Defines whether different header footer is required for odd and even pages in the section
documenteditor.selection.sectionFormat.differentOddAndEvenPages= true;
`
```

[Go to header footer region](#)

Double click in header or footer region to move the selection into it. You can also do this by using the following code.

```
`ts
documenteditor.selection.goToHeader();
`

`ts
documenteditor.selection.goToFooter();
`
```

[Link to previous](#)

Link to previous is enabled by default when document has more than one section. If you're using different headers and footers such as different first page or different odd and even pages, they can't be linked together because they're all separate.

Before setting or getting the link to previous value, use the ['goToHeader'](#) or ['goToFooter'](#) API to move the current selection to the header or footer region.

You can get or set the default header footer link to previous value of a section at cursor position by using the following sample code.

```
`ts
this.container.documentEditor.selection.sectionFormat.oddPageHeader.linkToPrevious = false;
this.container.documentEditor.selection.sectionFormat.oddPageFooter.linkToPrevious = false;
`
```

In case the document has different header and footer types, such as different first page, odd, and even pages.

```
`ts
// Different first page
this.container.documentEditor.selection.sectionFormat.firstPageHeader.linkToPrevious = false;
```

```

this.container.documentEditor.selection.sectionFormat.firstPageFooter.linkToPrevious = false;
//Even page
this.container.documentEditor.selection.sectionFormat.firstPageHeader.linkToPrevious = false;
this.container.documentEditor.selection.sectionFormat.firstPageFooter.linkToPrevious = false;
`

```

Note: When there is more than one section in the document, the Link to Previous option becomes available. By default, this feature is disabled state in UI and set to return false for the first section.

Header and footer distance

You can define the distance of header region content from the top of the page. Refer to the following sample code.

```

`ts
documenteditor.selection.sectionFormat.headerDistance= 36;
`

```

Same way, you can define the distance of footer region content from the bottom of the page. Refer to the following sample code.

```

`ts
documenteditor.selection.sectionFormat.footerDistace=36;
`

```

Close header footer region

Move the selection to the document body from header or footer region by double clicking or tapping the document area. You can also perform this by using the following sample code.

```

`ts
documenteditor.selection.closeHeaderFooter()
`

```

See Also

- [Working with Section Formatting](#)

Text format in React Document editor component

Document Editor supports several formatting options for text like bold, italic, font color, highlight color, and more. This section describes how to modify the formatting for selected text in detail.

Bold

The bold formatting for selected text can be get or set by using the following sample code.

```

`ts
//Gets the value for bold formatting of selected text.
let bold : boolean = documenteditor.selection.characterFormat.bold;
//Sets bold formatting for selected text.

```

```
documenteditor.selection.characterFormat.bold = true;
```

```
,
```

You can toggle the bold formatting based on existing value at selection. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.toggleBold();
```

```
,
```

Italic

The Italic formatting for selected text can be get or set by using the following sample code.

```
`ts
```

```
//Gets the value for italic formatting of selected text.
```

```
let italic : boolean = documenteditor.selection.characterFormat.italic;
```

```
//Sets italic formatting for selected text.
```

```
documenteditor.selection.characterFormat.italic= true|false;
```

```
,
```

You can toggle the Italic formatting based on existing value at selection. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.toggleItalic();
```

```
,
```

Underline property

The underline style for selected text can be get or set by using the following sample code.

```
`ts
```

```
//Gets the value for underline formatting of selected text.
```

```
let underline : Underline = documenteditor.selection.characterFormat.underline;
```

```
//Sets underline formatting for selected text.
```

```
documenteditor.selection.characterFormat.underline='Single' | 'None';
```

```
,
```

You can toggle the underline style of selected text based on existing value at selection by specifying a value. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.toggleUnderline('Single');
```

```
,
```

Strikethrough property

The strikethrough style for selected text can be get or set by using the following sample code.

```
`ts
```

```
//Gets the value for strikethrough formatting of selected text.
```

```
let strikethrough : Strikethrough = documenteditor.selection.characterFormat.strikethrough;
```

```
//Sets strikethrough formatting for selected text.
```

```
documenteditor.selection.characterFormat.strikethrough='Single' | 'Normal';
```

```
,
```

You can toggle the strikethrough style of selected text based on existing value at selection by specifying a value. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.toggleStrikethrough();
```

```
,
```

Superscript property

The selected text can be made superscript by using the following sample code.

```
`ts
```

```
//Gets the value for baselineAlignment formatting of selected text.
```

```
let baselineAlignment : BaselineAlignment =  
documenteditor.selection.characterFormat.baselineAlignment;
```

```
//Sets baselineAlignment formatting for selected text.
```

```
documenteditor.selection.characterFormat.baselineAlignment='Superscript';
```

```
,
```

Toggle the selected text as superscript or normal using the following sample code.

```
`ts
```

```
documenteditor.editor.toggleSuperscript();
```

```
,
```

Subscript property

The selected text can be made subscript by using the following sample code.

```
`ts
```

```
//Gets the value for baselineAlignment formatting of selected text.
```

```
let baselineAlignment : BaselineAlignment =  
documenteditor.selection.characterFormat.baselineAlignment;
```

```
//Sets baselineAlignment formatting for selected text.
```

```
documenteditor.selection.characterFormat.baselineAlignment='Subscript';
```

```
,
```

Toggle the selected text as subscript or normal using the following sample code.

```
`ts
```

```
documenteditor.editor.toggleSubscript();  
`ts
```

You can make a subscript or superscript text as normal using the following code.

```
`ts  
documenteditor.selection.characterFormat.baselineAlignment='Normal';  
`ts
```

Size

The size of selected text can be get or set using the following code.

```
`ts  
//Gets the value for fontSize formatting of selected text.  
let fontSize : number = documenteditor.selection.characterFormat.fontSize;  
//Sets fontSize formatting for selected text.  
documenteditor.selection.characterFormat.fontSize= 32;  
`ts
```

Color

The color of selected text can be get or set using the following code.

```
`ts  
//Gets the value for fontColor formatting of selected text.  
let fontColor : string = documenteditor.selection.characterFormat.fontColor;  
//Sets fontColor formatting for selected text.  
documenteditor.selection.characterFormat.fontColor= 'Pink';  
documenteditor.selection.characterFormat.fontColor= '#FFC0CB';  
`ts
```

Font

The font style of selected text can be get or set using the following sample code.

```
`ts  
//Gets the value for fontFamily formatting of selected text.  
let baselineAlignment : string = documenteditor.selection.characterFormat.fontFamily;  
//Sets fontFamily formatting for selected text.  
documenteditor.selection.characterFormat.fontFamily= 'Arial';  
`ts
```

Highlight color

The highlight color of the selected text can be get or set using the following sample code.

```
`ts
```



```
//Gets the value for highlightColor formatting of selected text.
```

```
let highlightColor : HighlightColor = documenteditor.selection.characterFormat.highlightColor;
```

```
//Sets highlightColor formatting for selected text.
```

```
documenteditor.selection.characterFormat.highlightColor= 'Pink';
```

```
,
```

[Toolbar with options for text formatting](#)

Refer to the following example.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, Selection, Editor, EditorHistory, ContextMenu } from  
'@syncfusion/ej2-react-documenteditor';
```

```
import { ToolbarComponent, ItemDirective, ItemsDirective } from '@syncfusion/ej2-react-navigations';
```

```
import { ComboBoxComponent } from '@syncfusion/ej2-react-dropdowns';
```

```
import { ColorPickerComponent } from '@syncfusion/ej2-react-inputs';
```

```
DocumentEditorComponent.Inject(Selection, Editor, EditorHistory, ContextMenu);
```

```
function App() {
```

```
let documenteditor: DocumentEditorComponent;
```

```
React.useEffect(() => {
```

```
componentDidMount()
```

```
, []);
```

```
function componentDidMount() {
```

```
documenteditor.selectionChange = () => {
```

```
setTimeout(() => { onSelectionChange(); }, 20);
```

```
};
```

```
}
```

```
function toolbarButtonClick(arg) {
```

```
switch (arg.item.id) {
```

```
case 'bold':
```

```
//Toggles the bold of selected content
```

```
documenteditor.editor.toggleBold();
```

```
break;
```

```
case 'italic':
```

```
//Toggles the Italic of selected content
```

```
documenteditor.editor.toggleItalic();
break;
case 'underline':
//Toggles the underline of selected content
documenteditor.editor.toggleUnderline('Single');
break;
case 'strikethrough':
//Toggles the strikethrough of selected content
documenteditor.editor.toggleStrikethrough();
break;
case 'subscript':
//Toggles the subscript of selected content
documenteditor.editor.toggleSubscript();
break;
case 'superscript':
//Toggles the superscript of selected content
documenteditor.editor.toggleSuperscript();
break;
}
}
//To change the font Style of selected content
function changeFontFamily(args): void {
documenteditor.selection.characterFormat.fontFamily = args.value;
documenteditor.focusIn();
}
//To Change the font Size of selected content
function changeFontSize(args): void {
documenteditor.selection.characterFormat.fontSize = args.value;
documenteditor.focusIn();
}
//To Change the font Color of selected content
function changeFontColor(args) {
documenteditor.selection.characterFormat.fontColor = args.currentValue.hex;
```

```
documenteditor.focusIn();
}

//Selection change to retrieve formatting
function onSelectionChange() {
  if (documenteditor.selection) {
    enableDisableFontOptions();
  }
}

function enableDisableFontOptions() {
  var characterformat = documenteditor.selection.characterFormat;
  var properties = [characterformat.bold, characterformat.italic, characterformat.underline,
    characterformat.strikethrough];
  var toggleBtnId = ["bold", "italic", "underline", "strikethrough"];
  for (let i = 0; i < properties.length; i++) {
    changeActiveState(properties[i], toggleBtnId[i]);
  }
}

function changeActiveState(property, btnId) {
  let toggleBtn: HTMLElement = document.getElementById(btnId);
  if ((typeof (property) == 'boolean' && property == true) || (typeof (property) == 'string' && property !==
    'None'))
    toggleBtn.classList.add("e-btn-toggle");
  else {
    if (toggleBtn.classList.contains("e-btn-toggle"))
      toggleBtn.classList.remove("e-btn-toggle");
  }
}

let fontStyle: string[] = ['Algerian', 'Arial', 'Calibri', 'Cambria', 'Cambria Math', 'Candara', 'Courier New',
  'Georgia', 'Impact', 'Segoe Print', 'Segoe Script', 'Segoe UI', 'Symbol', 'Times New Roman', 'Verdana',
  'Wdings'];

let fontSize: string[] = ['8', '9', '10', '11', '12', '14', '16', '18',
  '20', '22', '24', '26', '28', '36', '48', '72', '96'];
```

```
function contentTemplate1() {
return (<ColorPickerComponent showButtons={true} value='#000000'
change={changeFontColor}></ColorPickerComponent>);
}
function contentTemplate2() {
return (<ComboBoxComponent dataSource={fontStyle} change={changeFontFamily} index={2}
allowCustom={true} showClearButton={false} ></ComboBoxComponent>);
}
function contentTemplate3() {
return (<ComboBoxComponent dataSource={fontSize} change={changeFontSize} index={2}
allowCustom={true} showClearButton={false} ></ComboBoxComponent>);
}
return (
<div>
<ToolbarComponent id='toolbar' clicked={toolbarButtonClick}>
<ItemsDirective>
<ItemDirective id="bold" prefixIcon="e-de-icon-Bold" tooltipText="Bold" />
<ItemDirective id="italic" prefixIcon="e-de-icon-Italic" tooltipText="Italic" />
<ItemDirective id="underline" prefixIcon="e-de-icon-Underline" tooltipText="Underline" />
<ItemDirective id="strikethrough" prefixIcon="e-de-icon-Strikethrough" tooltipText="Strikethrough" />
<ItemDirective id="subscript" prefixIcon="e-de-icon-Subscript" tooltipText="Subscript" />
<ItemDirective id="superscript" prefixIcon="e-de-icon-Superscript" tooltipText="Superscript" />
<ItemDirective type="Separator" />
<ItemDirective template={contentTemplate1} />
<ItemDirective type="Separator" />
<ItemDirective template={contentTemplate2} />
<ItemDirective template={contentTemplate3} />
</ItemsDirective>
</ToolbarComponent>
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
```

```
  }}  
  readOnly={false}  
  enableSelection={true}  
  enableEditor={true}  
  enableEditorHistory={true}  
  enableContextMenu={true}  
/>  
</div>  
);  
}  
  
export default App;  
  
ReactDOM.render(<App />, document.getElementById('sample'));  
`
```

See Also

- [Feature modules](#)
- [Font dialog](#)
- [Keyboard shortcuts](#)

Paragraph format in React Document editor component

Document Editor supports various paragraph formatting options such as text alignment, indentation, paragraph spacing, and more.

Indentation

You can modify the left or right indentation of selected paragraphs using the following sample code.

```
`ts  
documenteditor.selection.paragraphFormat.leftIndent= 24;  
documenteditor.selection.paragraphFormat.rightIndent= 24;  
`
```

Special indentation

You can define special indent for first line of the paragraph using the following sample code.

```
`ts  
documenteditor.selection.paragraphFormat.firstLineIndent= 24;  
`
```

Increase indent

You can increase the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`ts
```

```
documenteditor.editor.increaseIndent()
```

```
,
```

Decrease indent

You can decrease the left indent of selected paragraphs by a factor of 36 points using the following sample code.

```
`ts
```

```
documenteditor.editor.decreaseIndent()
```

```
,
```

Text alignment

You can get or set the text alignment of selected paragraphs using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.textAlignment= 'Center' | 'Left' | 'Right' | 'Justify';
```

```
,
```

You can toggle the text alignment of selected paragraphs by specifying a value using the following sample code.

```
`ts
```

```
documenteditor.editor.toggleTextAlignment('Center' | 'Left' | 'Right' | 'Justify');
```

```
,
```

Line spacing and its type

You can define the line spacing and its type for selected paragraphs using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.lineSpacingType='AtLeast';
```

```
documenteditor.selection.paragraphFormat.lineSpacing= 6;
```

```
,
```

Paragraph spacing

You can define the spacing before or after the paragraph by using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.beforeSpacing= 24;
```

```
documenteditor.selection.paragraphFormat.afterSpacing= 24;
```

```
,
```

You can also set automatic spacing before and after the paragraph by using the following sample code.

```
`ts
```

```
documenteditor.selection.paragraphFormat.spaceBeforeAuto = true;
```

```
documenteditor.selection.paragraphFormat.spaceAfterAuto = true;
```

Note: If auto spacing property is enabled, then value defined in the `beforeSpacing` and `afterSpacing` property will not be considered.

Paragraph Border

You can apply borders to the paragraphs in a Word document. Using borders, decorate the paragraphs to set them apart from other paragraphs in the document.

The following example code illustrates how to apply box border for the selected paragraphs.

```
`ts
// left
documenteditor.selection.paragraphFormat.borders.left.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.left.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.left.color = "#000000";
//right
documenteditor.selection.paragraphFormat.borders.right.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.right.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.right.color = "#000000";
//top
documenteditor.selection.paragraphFormat.borders.top.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.top.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.top.color = "#000000";
//bottom
documenteditor.selection.paragraphFormat.borders.bottom.lineStyle = 'Single';
documenteditor.selection.paragraphFormat.borders.bottom.lineWidth = 3;
documenteditor.selection.paragraphFormat.borders.bottom.color = "#000000";
```

Note: At present, the Document editor component displays all the border styles as single line. But you can apply any border style and get the proper display in Microsoft Word app when opening the exported Word document.

Pagination properties

You can enable or disable the following pagination properties for the paragraphs in a Word document.

- Widow/Orphan control - whether the first and last lines of the paragraph are to remain on the same page as the rest of the paragraph when paginating the document.
- Keep with next - whether the specified paragraph remains on the same page as the paragraph that follows it while paginating the document.
- Keep lines together - whether all lines in the specified paragraphs remain on the same page while paginating the document.

The following example code illustrates how to enable or disable these pagination properties for the selected paragraphs.

```
`ts
documenteditor.selection.paragraphFormat.widowControl = false;
documenteditor.selection.paragraphFormat.keepWithNext = true;
documenteditor.selection.paragraphFormat.keepLinesTogether = true;
`
```

Show or Hide Paragraph marks

You can show or hide the hidden formatting symbols like spaces, tab, paragraph marks, and breaks in Document editor component. These marks help identify the start and end of a paragraph and all the hidden formatting symbols in a Word document.

The following example code illustrates how to show or hide paragraph marks.

```
`ts
documenteditor.documentEditorSettings.showHiddenMarks = true;
`
```

Toolbar with paragraph formatting options

The following sample demonstrates the paragraph formatting options using a toolbar.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, Selection, Editor, EditorHistory, ContextMenu
} from '@syncfusion/ej2-react-documenteditor';
import { ToolbarComponent, ItemDirective } from '@syncfusion/ej2-react-navigations';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
//Inject require modules.
DocumentEditorComponent.Inject(Selection, Editor, EditorHistory, ContextMenu);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  let items: ItemModel[] = [
    {
      text: 'Single',
```



```
,
{
text: '1.15',
},
{
text: '1.5',
},
{
text: 'Double',
},
];

function contentTemplate1() {
return (<DropDownButtonComponent items={items} iconCss="e-de-icon-LineSpacing"
select={lineSpacingAction} ></DropDownButtonComponent>);
}

function componentDidMount(): void {
documenteditor.selectionChange = () => {
setTimeout(() => { onSelectionChange(); }, 20);
};
}

function toolbarButtonClick(arg): void {
switch (arg.item.id) {
case 'AlignLeft':
//Toggle the Left alignment for selected or current paragraph
documenteditor.editor.toggleTextAlignment('Left');
break;
case 'AlignRight':
//Toggle the Right alignment for selected or current paragraph
documenteditor.editor.toggleTextAlignment('Right');
break;
case 'AlignCenter':
//Toggle the Center alignment for selected or current paragraph
documenteditor.editor.toggleTextAlignment('Center');
```

```
break;
case 'Justify':
//Toggle the Justify alignment for selected or current paragraph
documenteditor.editor.toggleTextAlignment('Justify');
break;
case 'IncreaseIndent':
//Increase the left indent of selected or current paragraph
documenteditor.editor.increaseIndent();
break;
case 'DecreaseIndent':
//Decrease the left indent of selected or current paragraph
documenteditor.editor.decreaseIndent();
break;
case 'ClearFormat':
documenteditor.editor.clearFormatting();
break;
case 'ShowParagraphMark':
//Show or hide the hidden characters like spaces, tab, paragraph marks, and breaks.
documenteditor.documentEditorSettings.showHiddenMarks =
!documenteditor.documentEditorSettings.showHiddenMarks;
break;
}
}
//Change the line spacing of selected or current paragraph
function lineSpacingAction(args: any) {
let text: string = args.item.text;
switch (text) {
case 'Single':
documenteditor.selection.paragraphFormat.lineSpacing = 1;
break;
case '1.15':
documenteditor.selection.paragraphFormat.lineSpacing = 1.15;
break;
```

```
case '1.5':
documenteditor.selection.paragraphFormat.lineSpacing = 1.5;
break;
case 'Double':
documenteditor.selection.paragraphFormat.lineSpacing = 2;
break;
}
setTimeout(() : void => {
documenteditor.focusIn();
}, 30);
}
// Selection change to retrieve formatting
function onSelectionChange() {
if (documenteditor.selection) {
var paragraphFormat = documenteditor.selection.paragraphFormat;
var toggleBtnId = ['AlignLeft', 'AlignCenter', 'AlignRight', 'Justify', 'ShowParagraphMark'];
//Remove toggle state.
for (var i = 0; i < toggleBtnId.length; i++) {
let toggleBtn: HTMLElement = document.getElementById(toggleBtnId[i]);
toggleBtn.classList.remove('e-btn-toggle');
}
//Add toggle state based on selection paragraph format.
if (paragraphFormat.textAlignment === 'Left') {
document.getElementById('AlignLeft').classList.add('e-btn-toggle');
} else if (paragraphFormat.textAlignment === 'Right') {
document.getElementById('AlignRight').classList.add('e-btn-toggle');
} else if (paragraphFormat.textAlignment === 'Center') {
document.getElementById('AlignCenter').classList.add('e-btn-toggle');
} else {
document.getElementById('Justify').classList.add('e-btn-toggle');
}
if (documenteditor.documentEditorSettings.showHiddenMarks) {
document.getElementById('ShowParagraphMark').classList.add('e-btn-toggle');
```

```
}  
// #endregion  
}  
}  
return (  
  <div>  
    <ToolbarComponent id='toolbar' clicked={toolbarButtonClick}>  
      <ItemDirective id="AlignLeft" prefixIcon="e-de-ctnr-alignleft e-icons" tooltipText="Align Left" />  
      <ItemDirective id="AlignCenter" prefixIcon="e-de-ctnr-aligncenter e-icons" tooltipText="Align Center" />  
      <ItemDirective id="AlignRight" prefixIcon="e-de-ctnr-alignright e-icons" tooltipText="Align Right" />  
      <ItemDirective id="Justify" prefixIcon="e-de-ctnr-justify e-icons" tooltipText="Justify" />  
      <ItemDirective type="Separator" />  
      <ItemDirective id="IncreaseIndent" prefixIcon="e-de-ctnr-increaseindent e-icons" tooltipText="Increase  
Indent" />  
      <ItemDirective id="DecreaseIndent" prefixIcon="e-de-ctnr-decreaseindent e-icons"  
tooltipText="Decrease Indent" />  
      <ItemDirective type="Separator" />  
      <ItemDirective template={contentTemplate1} />  
      <ItemDirective id="ClearFormat" prefixIcon="e-de-ctnr-clearall e-icons" tooltipText="Clear Formatting"  
/>  
      <ItemDirective type="Separator" />  
      <ItemDirective id="ShowParagraphMark" prefixIcon="e-de-e-paragraph-mark e-icons"  
tooltipText="Show the hidden characters like spaces, tab, paragraph marks, and breaks.(Ctrl + *)" />  
    </ToolbarComponent>  
    <DocumentEditorComponent  
id="container"  
ref={scope => {  
  documenteditor = scope;  
}}  
isReadOnly={false}  
enableSelection={true}  
enableEditor={true}  
enableEditorHistory={true}  
enableContextMenu={true}
```

```
enableTableDialog={true}
height={'330px'}
/>
</div>
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

[See Also](#)

- [Feature modules](#)
- [Paragraph dialog](#)
- [Keyboard shortcuts](#)

Styles in React Document editor component

Styles are useful for applying a set of formatting consistently throughout the document. In document editor, styles are created and added to a document programmatically or via the built-in Styles dialog.

Styles definition overview

A Style in document editor should have the following properties:

- **name:** Name of the style. All styles in a document have a unique name, which is used as an identifier when applying the style.
- **type:** Specifies the document elements that the style will target. For example, paragraph or character.
- **next:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined.
- **link:** Provides a relation between the paragraph and character style.
- **characterFormat:** Specifies the properties of paragraph and character style.
- **paragraphFormat:** Specifies the properties of paragraph style.
- **basedOn:** Specifies that the current style inherits the style set to this property. This is how hierarchical styles are defined. It can be optional.

The style type should match the inherited style type. For example, it is not possible to have a character style inherit a paragraph style.

Default style

The default style for span and paragraph properties is normal. It internally inherits the default style of the document loaded or document editor component.

Style hierarchy

Each style initially checks its local value for the property that is being evaluated and turns to the style it is based on. If no local value is found, it turns to its default style.

Style inheritance of different styles are listed as follows:

Character style

Character styles are based only on other character styles.

The inheritance is: Character properties are inherited from the base character style.

Paragraph style

Paragraph styles are based on other paragraph styles or on linked styles.

When a paragraph style is based on another paragraph style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the base paragraph style.
- Span properties are inherited from the base paragraph style.

When a paragraph style is based on a linked style, the inheritance of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Linked style

Linked styles are composite styles and their components are paragraph and character styles with link between them. To apply paragraph properties, take the properties from the linked paragraph style. Similarly, to apply character properties, take the properties from linked character style.

Linked styles are based on other linked styles or on paragraph styles.

When a linked style is based on a paragraph style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the 'basedOn' paragraph style.
- Character properties are inherited from the 'basedOn' paragraph style.

When a linked style is based on another linked style, the hierarchy of the properties is as follows:

- Paragraph properties are inherited from the paragraph style part in its base linked style.
- Span properties are inherited from the span style part in its base linked style.

Defining new styles

New Styles are defined and added to the style collection of the document. In this way, they will be discovered by the default UI and applied to the parts of a document.

Defining a character style

The following example shows how to programmatically create a character style.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, SfdtExport, Selection, Editor } from '@syncfusion/ej2-react-documenteditor';
```

```
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function componentDidMount() {
    let styleJson: any = {
      type: 'Character',
      name: 'New CharacterStyle',
      basedOn: 'Default Paragraph Font',
      characterFormat: {
        fontSize: 16.0,
        fontFamily: 'Calibri Light',
        fontColor: '#2F5496',
        bold: true,
        italic: true,
        underline: 'Single',
      },
    };
    //Create style in Document Editor.
    documenteditor.editor.createStyle(JSON.stringify(styleJson));
  }
  return (
    <DocumentEditorComponent
      id="container"
      height={'330px'}
      ref={scope => {
        documenteditor = scope;
      }}
      isReadOnly={false}
      enableSelection={true}
```

```

enableEditor={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Defining a paragraph style

The following example shows how to programmatically create a paragraph style.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor } from '@syncfusion/ej2-react-
documenteditor';

//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);

function App() {
let documenteditor: DocumentEditorComponent;
React.useEffect(() => {
componentDidMount()
}, []);
function componentDidMount() {
let styleJson: any = {
type: 'Paragraph',
name: 'New ParagraphStyle',
basedOn: 'Normal',
characterFormat: {
fontSize: 16.0,
fontFamily: 'Calibri Light',
fontColor: '#2F5496',
bold: true,
italic: true,
underline: 'Single',
},

```



```

paragraphFormat: {
  leftIndent: 0.0,
  rightIndent: 0.0,
  firstLineIndent: 0.0,
  beforeSpacing: 12.0,
  afterSpacing: 0.0,
  lineSpacing: 1.0791666507720947,
  lineSpacingType: 'Multiple',
  textAlignment: 'Left',
  outlineLevel: 'Level1',
},
};
//Create style in Document Editor.
documenteditor.editor.createStyle(JSON.stringify(styleJson));
}
return (
<DocumentEditorComponent
  id="container"
  height={'330px'}
  ref={scope => {
    documenteditor = scope;
  }}
  isReadOnly={false}
  enableSelection={true}
  enableEditor={true}
/>
);
}
export default App();
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Defining a linked style

The following example shows how to programmatically create linked style.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, DocumentEditor, SfdtExport, Selection, Editor } from
 '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
function App() {
let documenteditor: DocumentEditorComponent;
React.useEffect(() => {
componentDidMount()
}, []);
function componentDidMount() {
let styleJson: any = {
type: 'Paragraph',
name: 'New Linked',
basedOn: 'Normal',
next: 'Normal',
link: 'New Linked Char',
characterFormat: {
fontSize: 16.0,
fontFamily: 'Calibri Light',
fontColor: '#2F5496',
},
paragraphFormat: {
leftIndent: 0.0,
rightIndent: 0.0,
firstLineIndent: 0.0,
beforeSpacing: 12.0,
afterSpacing: 0.0,
lineSpacing: 1.0791666507720947,
lineSpacingType: 'Multiple',
textAlignment: 'Left',
```

```

outlineLevel: 'Level1',
},
};
documenteditor.editor.createStyle(JSON.stringify(styleJson));
}
return (
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Applying a style

The styles are applied using the **applyStyle** method of **editorModule**, the parameter should be passed is the **Name** of the Style.

The styles of the **Character** type is applied to the currently selected part of the document. If there is no selection, the values that will be applied to the word at caret position. The styles of **Paragraph** type follow the same logic and are applied to all paragraphs in the selection or the current paragraph.

When there is no selection, styles of **Linked** type will change the values of the paragraph, and apply both the Paragraph and Character properties. When there is selection, Linked Style changes only the character properties of the selected text.

For example, the following line will apply the "New Linked" to the current paragraph.

```

`ts
this.documenteditor.editorModule.applyStyle('New Linked');
//Clear direct formatting and apply the specified style
this.documenteditor.editorModule.applyStyle('New Linked', true);

```

List format in React Document editor component

Document Editor supports both the single-level and multilevel lists. Lists are used to organize data as step-by-step instructions in documents for easy understanding of key points. You can apply list to the paragraph either using supported APIs.

Create bullet list

Bullets are usually used for unordered lists. To apply bulleted list for selected paragraphs, use the following method of 'Editor' instance.

```
applyBullet(bullet, fontFamily);
```

Parameter	Type	Description
-----------	------	-------------

bullet	string	Bullet character.
--------	--------	-------------------

fontFamily	string	Bullet font family.
------------	--------	---------------------

Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
```

Create numbered list

Numbered lists are usually used for ordered lists. To apply numbered list for selected paragraphs, use the following method of 'Editor' instance.

```
applyNumbering(numberFormat,listLevelPattern)
```

Parameter	Type	Description
-----------	------	-------------

numberFormat	string	“%n” representations in ‘numberFormat’ parameter will be replaced by respective list level’s value. “%1” will be displayed as “1”
--------------	--------	---

listLevelPattern(optional)	string	Default value is 'Arabic'.
----------------------------	--------	----------------------------

Refer to the following example.

```
`ts
```

```
documenteditor.editor.applyNumbering('%1', 'UpRoman');
```

Clear list

You can also clear the list formatting applied for selected paragraphs. Refer to the following sample code.

```
`ts
```

```
documenteditor.editor.clearList();
```

Working with lists

The following sample demonstrates how to create bullet and numbering lists in document editor.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Selection, Editor, EditorHistory,
ContextMenu, TableDialog } from '@syncfusion/ej2-react-documenteditor';
import { ToolbarComponent, ItemDirective, ItemsDirective, } from
 '@syncfusion/ej2-react-navigations';
DocumentEditorComponent.Inject(Selection, Editor, EditorHistory, ContextMenu,
TableDialog);
function App() {
  let documenteditor;
  function toolbarButtonClick(args) {
    switch (args.item.id) {
      case 'Bullets':
        //To create bullet list
        documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
        break;
      case 'Numbering':
        //To create numbering list
        documenteditor.editor.applyNumbering('%1', 'UpRoman');
        break;
      case 'clearlist':
        //To clear list
        documenteditor.editor.clearList();
        break;
    }
  }
  return (
    <div>
      <ToolbarComponent id="toolbar" clicked={toolbarButtonClick}>
        <ItemsDirective>
          <ItemDirective id="Bullets" prefixIcon="e-de-ctnr-bullets
e-icons" tooltipText="Bullets"/>
          <ItemDirective id="Numbering" prefixIcon="e-de-ctnr-
numbering e-icons" tooltipText="Numbering"/>
          <ItemDirective id="clearlist" text="Clear"
tooltipText="Clear List"/>
        </ItemsDirective>
      </ToolbarComponent>
      <DocumentEditorComponent id="container" height={'330px'}
ref={scope => { documenteditor = scope; }} isReadOnly={false}
enableSelection={true} enableEditor={true} enableEditorHistory={true}/>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
```

```

import * as React from 'react';
import { DocumentEditorComponent, Selection, Editor, EditorHistory,
ContextMenu, TableDialog } from '@syncfusion/ej2-react-documenteditor';
import { ToolbarComponent, ItemDirective, ItemsDirective, } from
 '@syncfusion/ej2-react-navigations';
DocumentEditorComponent.Inject(Selection, Editor, EditorHistory, ContextMenu,
TableDialog);
function App() {
    let documenteditor: DocumentEditorComponent;
    function toolbarButtonClick(args) {
        switch (args.item.id) {
            case 'Bullets':
                //To create bullet list
                documenteditor.editor.applyBullet('\uf0b7', 'Symbol');
                break;
            case 'Numbering':
                //To create numbering list
                documenteditor.editor.applyNumbering('%1', 'UpRoman');
                break;
            case 'clearlist':
                //To clear list
                documenteditor.editor.clearList();
                break;
        }
    }
    return (
        <div>
            <ToolbarComponent id="toolbar" clicked={toolbarButtonClick} >
                <ItemsDirective>
                    <ItemDirective id="Bullets" prefixIcon="e-de-ctnr-bullets
e-icons" tooltipText="Bullets" />
                    <ItemDirective id="Numbering" prefixIcon="e-de-ctnr-
numbering e-icons" tooltipText="Numbering" />
                    <ItemDirective id="clearlist" text="Clear"
tooltipText="Clear List" />
                </ItemsDirective>
            </ToolbarComponent>
            <DocumentEditorComponent id="container" height={'330px'}
ref={scope => { documenteditor = scope; }} isReadOnly={false}
enableSelection={true} enableEditor={true} enableEditorHistory={true} />
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />

```

```
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
<style>
  /** Document editor sample level font icons*/
@font-face {
  font-family: 'Sample brower icons';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSi8AAAEoAAAAVmNtYXDq0OvsAAACfAAAAALBnbHlma57A
AgAAA6wAAC90aGVhZBF0mKkAAADQAAAAANmhoZWEIURAAAAAArAAAACRobXR4/AAAAAAAYAAAD8b
G9jYXhXbWAAAMsAAAAGl1heHABZQE/AAABCAAAACBuYWllCHiPawAAMyAAAAL9cG9zdAI/4kAAD
YgAAADOQABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAPwABAAAAAQAGurlbV8PPPU
ACwQAAAAAANcxqdwAAAAA1zGp3AAAAAAD8wP0AAAACAACAAAAAAAAAAAAEAAAA/ATMAHAAAAAAAgAA
AAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAB6wAyAQgAAAIABQMAAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnPQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
ABAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAAB
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAA
AAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAA
QAAAAEAAAAAAAAAgAAAAAMAAUAMAAQAAABQABACcAAAABAAEAAEAOC9//8AAOC//8AAAABAAQ
AAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAEGATABQAFQAWABcAGAAZABoAGwAc
AB0AHgAfACAAIQAIACMAJAALACYAJwAoACkAKgArACwALQAUAC8AMAAxADIAMwA0ADUANgA3ADgAO
QA6ADsAPAA9AD4AAAAAFoAngDuAg4CWAJ4ApoCxcgMGA9QD8gVgBcoGSgAMByoHYggKCLII3AkICb
wJ3An4CjIKvAr4C8QL4AwADEIM6g0MDawNxxg42DoIOpA8yD2YPhA+2EFgQdhEWEcAR2BI4EyYTXhO
```

```
UE8AUPhRWFJAUnhVAFeqWMBdiF4IXugAOAAAAAAPzA7UAAwAHAAsADwATABcAGwAfACMAJwArArAC8A
MwA3AAALmZUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFmZUjBTM1IyUhNSEFITUhJTM1IwUzNSMHMzUjB
zM1IwO1Pz+7fX36fX36fX36fX0C7vr6/on6+v6J+voB9AH0/gz+DAF3/okC7vr6/si7u/p9fbw/P0
t9fX19fX19fX19fX19fX19fHx8fX19fX19fX0AAAAACAAAAAN2A/MABAArAAABEwkBEQMfCTM/BAK
BHwYzPwkRIQM4Af7H/sG/AQIDBQYGCakJCQkJCQkIBwEKAQsFBQUGBgYGDawFCQgGBgUDAgH9EgO1
/JUBZ/6aA2r8lgoJCAgHBwUEAwEBawQFBwEx/s4FBAMDagEBAgIEBQYICAgJCgOpAAAAABQAAAAAD8
wPUAAQACAAAnAC4AMgAAJRujJzcHITU3JQ8DHQEfBj8GNS8GDwETEScHAQcRAYERIQO1j51SDf192g
HdAgICAgICBAUGBgYGBgYFBAMBAQMCBwUGCwKFrtp9/sfaPwPo/Bh9E5xR7c7bQgIDBQYHBgYFBQq
DAQEBAQMEBAUGCwoFAgYDAQECAwE9/UDzfQE42gIG/JYDqAAAAIAAAAA/MD8wB/AQUAAAEVDx0r
AS8dPQE/HTsBHx0FFR8HAQ8DHwgZPwQBHWc/Hy8fDx4DtQECAwMFBQUGBwgICQkKCsLDawNDQ4OD
g4PDw8QEBAQE8QDw8ODg4ODQwNDASLCgoJCQgIBgcFBQCEAwIBAQIDBAQFBQYHCAgJCQoKCwsMDA
0NDg4ODg8PDxAQEBAQDxApDw4ODg4NDA0MCwsKCgkJCAgGBwUFBAQDAgH9UQEEBgcKDA4P/s8GBQI
BAgMGCAQFBGsMDAwLBQUFAS0YGBobHB0dHhMTExiTERIREBAQDw8ODg0MDAsLCQkJBwcGBQQDAwEB
AQEDAwQFBgcHCQkJCwsMDA0ODg8PEBAQERIRExiTExmTExmMSExESEERARDw8PDg4NDawLCwoJCAcHB
gUEBAIBAn0QEBAPDw8ODw0ODQ0MDAsLCgoJCQgIBwYFBQUdAwIBAQIDAwUFBQYHCAgJCQoKCwsMDA
0NDg0PDg8PDxAQEBAQDxApDw4ODg4NDA0LDAsKCgkJCAcHBwUFBAQDAgEBAgMEBAUFBwcHCAkJCgo
LDAsNDA0ODg4ODw8QDxAQDw8dHRwbGhgY/s4KCgSLCwsKCQUDBAQCAgQEAwUBLRAODAoHBgQBAQED
AwQFBgcHCAoJCwsMDA0ODg8PDxEQERIRExiTExmTExmMSExESERAQE8PDg4NDawLCwkJCQcHBgUEA
wMBAQEBAwMEBQYHBwkJCQsLDawNDg4PDxAQEBESEhISExMAAAsAAAAA9QD1AADAaCACwAPABMAFw
AbAB8AIwApAC8AACUzNSM1MzUjNzM1IwcZNSMHMzUjBzM1IwcZNSM3MzUjNTM1IyczIREhESMRfSE
RIQHhPj4+Pvo+Pn0+Pn0+Pn0+Pn0+Pvo+Pj4++j4CcFzUPgOo/FjnPj8+Pz4+Pj4+Pj4+Pj8+Pz59
/NQDLpzUPgOoAAAEAAAAAAPzA/MAAwAHAAsADwAANyE1ITUhNSE1ITUhNSE1IQwD6PwYA+j8GAPo/
BgD6PwYDD/6Pvo++j8AAAAAQAAAAADtQO1AAsAABMJARcJATcJAScJAUsBiF53LAGJAYks/ncBiS
z+d/53A4n+d/53LAGJ/ncsAYkBiSz+dwGJAAAFAAAAAAPzA/MAAwAHA0AEQAVAAA3ITUhJSE1ISU
XNyc3JxchNSE1ITUhDAPo/BgBOQKv/VH+x5IqaWkqpWkv/VH+xwPo/BgMP/o+fZwscHAsH76PwAA
BwAAAAADhNzPzAAMABwATABcAGwAfACsAACUzNSMHMzUjNjYmVMxUzNTM1IzUjSE1ISUzNSMHMzUjF
yMVMxUzNTM1IzUjAn0+Pvo/P30+Pj8+Pj/+DAPo/BgCtT4++j8/ft4+Pz4+P8g+Pj4/P/r6Pz59Pr
w+Pj4+Pz4+P/oAAAAEAAAAAAPzA/MAMAAzAGkApwAAJRUPdi8OPQE/Bx8GAQcnBQ8JFR8OPw81Lwk
BFQkCJwcXByEBNT8GOWEfBhEzETUvDg8OA6sBAGMDAwUEBgUGBwYHBwgHBwcHBgYGBQUEBAMCAgEB
AgYJChINDRsMCwkIBAL+pOriAsMBNBUJCggHBQMBAwMFBgcJCQsLDA0NDg4PDw8ODQ0MCwoKCAcGB
QQCAQMEBgwJCgoVEzT94/7HAVgBloUwYBX98QECAQIDAwUFBgcGBgUFAwMCAT4CAgMEBQUGBwcICA
kJCQkKCQkJCAgHBwYFBQQDAgKuCQkICAgHBwcFBQUEAwIBAQEBAgMEBQUFBwcHBwkICQkHCQgTFRU
fFRQpFRUVExiJAQ3i4iMCSCQSExQTExmRERAPDw4ODAsLCQgHBQQDAQEEDBAUHCAkLCwwODg8PEAgR
ExMTHRMTIEiAcQgHUcP67/qgBh6AodBQBDIoGBgUFBAMCAgMEBQUGBv7nARkKCQkJCAcIBgYGBAQDA
wEBAQEDAwQEBgYGCACJCAkJAAAAAgAAAAAD8wPzAAMADAAANyE1ISUnBwkBJwCRiwwD6PwYAfTkLA
EvAS8s4z8MP+blLP7OATIs5QLDAAAABgAAAAAD8wPzAB8AXwCfAOIA5QEYAAABFQ8FKwEvBj8GOWE
fBQcVHW4/Dy8OIw8OFw8PLw8/Dx8OJyMPAycHFw8EJwcfBacXNx8DBxc3HwE/Ahc3Jz8DFzcnPwUn
By8DNycHLwM1IycjNSURHw8hNSEjLwU1ETU/BTMhFTMVMz0BLw8hDw4DEgICAwQEBAUFBQQDAwMBA
QEBAwMDBAUFBQQEBAMCAm8CAgMDBQFwYFBgYICQkJCAkIBwCHBgYFBAQDAgEBAQECAwQEBQYGBw
CHCAkICQkJCAgIBwYHBQUFAwMCAT4BAGMFBQCICQkLCwwMDQ0ODg4MDQwLCgoJBwcGBQMCAQECAw
GBwcJCgoLDA0MDg4ODQ0MDAsLCQkIBwUFwKIAhQTEhIiKiIJCwoIBDMKNAEDBQYvHDAODg8TFDQU
FBQPDwkUNBQSDw0QMBwvBQUEAQEOCjMICAQIioiFRESFTgQkP30AQECBAQEAgYGCACICQkJCgGW/
moGBgYEBAMCAgMEBQUGBGCW+j4BAwMEBAYG1gYICAgJCAoJ/mUKCQkJCAcIBgYGBAQEAQEBBQUEBA
QDAgICAgMEBAQFBQUEAwMDAQEDAwMEBQUJCAkIBwcHBgYFBAQDAgEBAQECAwQEBQYGBwcHCAkICQk
JCAgHCAYGBgUEBAMCAgEBAgMEBAUGBgYIBwgICQkODQ0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCwsM
DA0NDg4NDQ0MCwoKCQCHBgQEAgEBAgQEBgcHCQoKCwwNDQ22BAYICikkkQoQERILCTcKGBQTEhsxH
A4NCww3FDgDAQECATgTOAoLDBECMBwNERMTDQk4CRQQEBQpJCKLBwYENvqPDFzUCgkJCAkHCAYGBg
QEAWMBAT8CAwQFBQYGAywGBgUFBAMC+nyCCQkJCQgIBwfVBwUFBAMCAQEBAQIEBAQGBgYIBwgJCQk
AAAAABAAAAAADdgPzAAMABwAiAFMAADchNSEBFQc1AQ8KHQEHnZUvCSM7AR8PBzVMVnZUzJz8PMzUj
FSE1I4kC7v0SAbZ+ATIGBgOIBwUFAwMCAf6JAQIBAwQEBQcICGyECgoSEQ4MDAoIBwCFawMDAQEBA
m76bQIBAQICAwQFBggICgsNDhESFD/9kD8MfQF3UESUATgGBg0NDg4ODg8PDxBfYA8PDw4PDg4NDg
0MAwQFBwgJCgsLDQ4ODhAPIH76jW1+IA8QDg4ODQsLCgkIBwUEA7x9fQACAAAAAAPzA7UAVABgAAA
BDwUVPwY7AR8JFQ8QFTM1Iz8SLw8HBQkBFwkBNwkBJwkBA1cODg0MDQwMDAwMDQwNDACNDaoJBAMD
AgEBAgQGBwkrRDDcODAsKCAyCAgH6tAEBAQECwXAGQ8MBQQEBAICAQEBAgIEBQUHdBwGCwMDAwNE
PylATH+zzzIBJGEmMf7QATAx/tr+2gOzAwMFBgcIOQoJBwYEBACBAUHBQQGBQCgdGwMGcwoKDgorCw
wMDQ4PCAgIJJTMHBQYFBGsLMBUPDwgICAKJCA8LDAsLCgkICACGBQCEAwIBAQEm/nH+cCYBgV5/JQG
QAY8m/n4BggAACgAAAAAD8wPzAAMABwALAA8AEwAXABsAHWAjACgAAAAEVIzUjFSM1IxUjNQEVIZUj
```


FSM1IxUjNQEVIZUjFSM1IxUjNQMpAREhA7X6Pvo++gNq+j76PvoDavo++j76PwE5Aq/8GAFF+vr6+vr6ATj6+vr6+voBOPr6+vr6+vxXA+gAAAAAAQAAAAAD8wPzAIoAABMBNwEhMx8dHQEPHSsBFTM/Hy8eIyEBJwWbJsn+ygIQDw4ODg0ODQwNDawLCwsKCgkJCAgHBWYGBQUDawMCAQECAwMDBQUGBgHCAGJCQoKCwsLDawNDA0ODQ4ODg9eXhIREREREBAQDw8ODg4NDawLCwoKCQgIBWYFBQQDAgEBAQECAwQFBQYHCAgJCgoLCwwMDQ4ODg8PEBAQERERERL99wEtKQKY/q0vAQkCAQMDBAQFBgYHBWgICQoJCgsLDAsMDQ0NDQ4NDg8ODw4ODg0ODQ0MDawLCwsKCgkJCAgIBGcFBQUDBAICAT8BAQIDBAUFBgICAKKCgsLDawNDg4ODw8QEBAREREREhIREREREBAQDw8ODg0NDQwLCwoKCQgHBWcFBQMDawEBcI8AAAAUAAAAA/MD8wALAA8AEwAXACcAACUjFTMVmZUzNSM1IwEVIzUjFSM1IxUjNQMhESMVIzUjFSM1IxUjNSMCAH19P3x8PwG1+j76Pvo/A+g/+j76Pvo/yD99fT99AXb6+vr6+vr+yAJx+vr6+vr6AAAAAUAUA9QAawAHAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAVwBbAF8AYwBnAGsAbwAAJTM1IwczNSMHMzUjBzM1IwczNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSMhMzUjBzM1IwczNSMFMzUjBzM1IwczNSMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjNSE1IQOWPj59Pz99Pz+7Pj68Pz99Pz98Pj4Daj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4BtT4+AbU+Pn0/P30/P/6JPz99Pz98Pj4BtT4+AbU+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+A6j8WCw+Pj4+Pj4+Pj4+Pj4+Pj8/Pz8/Pj8/Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+PgAFAAAAAAOWA/MAAwAfACIAQACFAAABBYM3JyMVMwcjFTMHFzcBxc3MzUjNzM1IzcnByM3JyUjNScVMxEPBiMhIy8GET8GMwcRFR80IT8ONRE1Lw8hDw4CRxJ8EjZwZxJVTA0+DnwMPPQ5vZhJVTA0+DnwMPPQGIjz76AQIDBAQGBQf9kAcFBgQEAWIBAQIDBAQGBQdeAgIDBAUFBgHCAGJCQkKAnAKCQkJCAgHBWYFBQQDAgICAgMEBQUG1gcHCAgJCQkKJ/mUKCQkJCAgHBWYFBQQDAgIBwn19Pj59P1kJY1kJYj59P1kJY1kJmI8s+v2vBgYFBQQDAgIDBAUFBgYDLAYGBQUEAwIf/NQKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAKJCgJXCQkJCQgIBwfVBgYFBAMCAQEBAQMDBAQGBgYIBwkICQkAAAADAAAAAAPzA/MACAAMABUAACUXNxExERc3JyUhNSE1JwcXNycHESMBgypTP1Mqnf3tA+j8GAH0UyqcnCpTPvYvTP75AQdML419Pq9ML42NL0wBBwAFAA AAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISUXBxc3JwUhNSE1ITUhDAPo/BgBOQKv/VH+x29vLJu baAQ0Cr/1R/scD6PWyDD/6Puxvbybmx4++j8AAwAAAAADGQO1ACMARgCbAAABOWEfDg8OKwEREx8PDw8jEQcVESE/GzUvDzU/DzUvECEBzQ0NGRgVFBIQDw0LCQgGBQIBAQIEBgJCwwODhERExUVF5F7FQSERAOdQwKCQgGBQMCAQECEBAYHCAsLDg4PERITFBZtawEKHx4dDg0NDQwMDAsLCwoKCQgHBWYGBQ QEAWICAQECEBQYICQsNDw8REhMUFhYSERAPDg0MCwoIBWYFAwIBAwQGBAUFBg0PERMVfHcZGxz+7gHiAgMEBgHCQsLDQ4PEBITeHeQDw4NDQsKCAgGBAQCAToBdwEBAwMFBQcHCQkLCwwODhASEQ8PDg0LCwoIBWUFAwIBARudP/3OAMGAwQFBQYGBWcICAKJCgoKCgsLDawMDQwODQ4WFRQTEhAQDw0MCgoHBgUDAwYHCQkKCw0NDg8PEBAREhILFRUTCQkICRAPDQ0KCQcFAwIAAAAAABAAAAAD8wPzAAMABwALAA8AADchNSE1ITUhNSE1ITUhNSEMAq/9UQPo/BgCr/1RA+j8GAw/+j76Pvo/AAAAAAMAAAAA7UD8wADAACaCwAANYE1IQERIREDIRehyAJw/ZACr/0SPgNq/Ja9vAI8/JYDavxXA+gABQAAAAAD8wPzAAMABwATABcAJwAAARUjNRMVIZUFIxUzFTM1MzUjNSMnFSM1ITMVIxUzFSMVMxUjFSERIQI/+vr6AfN9fT99fT/5+v7H+vr6+vr6AnH9jwFF+voBOPr6Pz59fT59+vr6+j76Pvo/A+gAAAAAUAUAAN2A/MAAwB4AAA3ITUhExUfHj8eNREjEQcVDxQrAS8UNQMjiQLu/RI/AQIDAwQFBgYHCAgJCQoKCwsMDA0NDQ4PDg8PDxAQEBAQEAE8PDw4PDg0NDQwMCwsKCgkJCAgHBgYFBAMDAgE+AQICAwMEBQUMDQ8RExMWFgwmDAwNDA0NDA0MDAwMCwsWExMRDw0MCgQDAwICAT4MPwF3EQ8QDw8PDw4ODg0MDQsMCwoKCQgJBWcGBgUEBAICAQEBAQICBAQFBgYHBwkICQoKCwwLDQwNDg4ODw8PDxAPEQIy/c4NDQwNDAsMDAsVFBIRDw4LCgQEAgMBAQEBAwIEBAQGCw4PERIUFRcMCwwNDA0CPwAFAAAAAAAPzA/MAAwAHABMAFwAoAAABFSM1ExUjNQUjFTMVmZUzNSM1IyUVIZUDKQE1IzUzNSM1MzUjNTM1IQK7+fn5/sd9fT98fD8CMvk/ATgBOfR6+vr6+v2PAUT5+QE5+vo/Pn19Pn36+vr8Vz/6Pvo++j8AAAADAAAAAAN2A/MAJQBIAK8AAAEhOWEfBRURFQ8FIyEjLwU1ETU/BTM1FSM1Pw47AR8NBRUjDw8RHw8hPw8RLw8jNS8PDw4BRQF2XgYGBgQEAWICAwQFBQYG/c4GBgYEBAMCAgMEBQUGBgGW+gECAwQFBggICQkLCgwMDA0NDawMCgsJCQgIBGU EAwL+yV4KCQkJCACIBgYGBAQEAQEBAQECEBAQEAgYGCACICQkJCgIyCgkJCQgHCAYGBgQEBAIBAQEBAgQEBAYGBggHCAkJCQpeAQIFBgKcG0NDhAQERITExMTEhEQEA4NDQoKCAYFAGI+AgMEBAYFB/5LBgYFBQQDAgIDBAUFBgYBtQcFBgQEAWL6u7sNDawMCwoKCQgHBgUFAwICAwUFBgICQoKCwwMDA27AQECBAMFBgYGBWgICQkJCv5LCgkJCQgHCAYGBgQEBAIBAQEBAgQEBAYGBggHCAkJCQoBtQoJCQkICACGBWUFBMCAQG7ExMSEREPEdg4MCwkIBgUDAEQBQYICQsMDg4PERESEwADAAAAA01A/MAAwAHAAsABMhNSE1ESERAYERICgCcP2QAq/9Ej4DavyWAoe8cvyWA2r8VwPoAAMAAAAA5YDtQADAACADwAAJTMRIyUhNSERIEZeSE1IQHhPj7+iQMs/NQBdz4Bd/zUSwE4Pz4Bd/7HATk+AAADAAAAAAPzA7UADA AQACcAACUHIy8DPQE/AyUJAw8HHwghNQUJAQIUP9GyAwICAgIDLQK0/qX+1AFb/bYGBQQDAWIBAQEBAgMDBAUGxQMK/joBxv57xD2tAwQEQUEBASRWP6xASEBUP4fBgYHCAgICAgICAgIBWcGBr8+AgG3AXcAAAAAUAUA9QAawAHAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAVwBbAF8AYwBnAGsAbwAAJTM1IwczNSMHMzUjBzM1IwczNSMHMzUjJTM1IwUzNSM1MzUjBTM1IyUzNSMHMzUjBzM1IwczNSMHMzUjBzM1IwczNSM1MzUjBTM1IyUzNSMFMzUjATMRIwczNSMHMzUjBzM1IwczNSMHMzUjBzM1IwMzPz99Pz+7Pj68Pz99Pz98Pj4BtT4+/ks+PgG1Pj7+Sz4+Au0/P30/P30/P30/P30/P30/P30/P3w+PgG1Pj7+Sz4+AbU+Pv5LPj4Daj4+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+P

j4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj4+Pn0/Pz8+Pz8//NQDqD4+Pj4+Pj4+Pj4+PgAAAAAEAAAAAA
PzA/MAAwAHAAsADwAAJSE1ISUhNSE1ITUhJSE1IQFFAq/9Uf7HA+j8GAE5Aq/9Uf7HA+j8GAw/+j7
6Pvo/AAMAAAAAA/MDtQASAD0AgAAAATmFBRUHAYETPwQzAx8LMyEfBxUhDwcDETU/BgcRIRM/Ai8L
Iz0BLw0jIS8LkWiPDQOWBgQFBgYDAQGu/VjSAwIDCAgEQgUFBQV7BgCHBwCICAgBCAcFBgQEAWIB/
lENDQwLCgoIA7ECAwQFBQYXGgMiWAQBAQICBQUHCAoJCwsMBmMCAGMEBQUGBwCICAKJCQR++AUFBQ
V7BgCHBwGHCaigCgkJCAkHCAYGBgQEBAIBA j4BAgUGCAgFBf5zAaQEAWMFAgE5AQECA2IEBQMDAGI
BAQIDA wUFBgZeAQMEBgCJCwX+nwJqBgYFBQMDAGef/PMBtQwMCwMCwoKCQgGBQQCAV4JCQkJCAgH
BwYFBQQDAgIBAQIDYgUEAWMCAGECAGMEBQUGBwCICAKJCQAAAwAAAAAD8wPzAAMABwALAAA3ITUhN
SE1ITUhNSEMA+j8GAPo/BgD6PwYDD/6u/r6AAAAAUAAAAA/MD8wADACMAKwAvAE8AAAEVITUnDw
MfBz8HLwYrAQ8BJREjNSEVIxEBSERAYsBDwCVazMVITUzAzUvBysBESECU/6KswQDAQEBAgIEBQY
FBgYGBQUEAWIBAQIDBAQGBQcGBQYDHRv+DLsCcP6KP7sHBgYLCgkGBQIB+gH0+gECAGYHCgoMBge7
/gwBRfr6sgUFBgYGBgUFBAMBAQEBAWQFBQYGBgYFBQQDAgIDQ/6Ku7sBdgF3/sgBOP7IAQIFBgkKC
wYG/kR9fQG8BgYGCgHBgQBAXcAAAAABwAAAAAD8wPzAAMABwALAA8AEwAlADEAAAEVIZUjFSM1Ix
UjNQEVIZUTFSM1ITMVIxUzFSM1IxUjNSMRIREhBRCHFzcxNyc3JwcnA7X6Pvo++gNq+vr6/unZ+vr
6Pvo/A+j9sP5ochAsCHAtCHAtCHABRfr6+vr6+gE4+voBOPr6+j76+vrd/awD6CxcwC1wcC1wcCw
cAADAaaaaAN2A/MAAwAGAA4AADchNSEBIRMBMzchFzMBI4kC7v0SAf3+84f+yE5OATHOTv7vTwX9A
bUBd/1R+voC7gAAABUAAAAA9QD1AADAACACwAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMAUQ
BVAfKAXQAAJTM1IwczNSMHMzUjBTM1IwczNSMHMzUjJTM1IwUzNSMlMzUjBTM1IwEzNSMFMzUjJTM
1IwUzNSMlMzUjBzM1IwczNSMHQEHFSERMxEhNSERIwczNSMHMzUjBzM1IwOWPj59Pz99Pz/+iT8/
fT8/fT8/A2s+PvyVPz8Daz4+/JU/PwNrPj781T8/A2s+PvyVPz8Daz4+fT8/fT8/u/5KAbY+Abb+S
j68Pz99Pz99Pz8sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/PwE4Pz8/Pj8/Pz4+Pj4+Pj59+j7+SwG1PgG1Pj
4+Pj4+AAAABAAAAAD8wPzAAMADwATABsAAAEVITUBFwcXNxc3JzcnBycBFSE1ByMRMxEhESEDtf6
J/c5wcCxcwC1wcC1wCAN9/ok+Pj4B9P4MAUX6+gEMCHAsCHAsCHAsCHABOPr6+v6K/scD6AACAAAA
AAMvA/MAAwAMAAA3ITUhNychHCQEnBxExj5wIy/c765CwBLwEvLQO+DD/m5Sz+zwExLOUCwWAAAAAEA
AAAAAPzA/QAAwAHAAsAGQAAJSE1IREhNSERITUhBRc3ESchFzcnBxEXNycBgwJx/Y8Ccf2PAnH9j/
6JKlNTKpydKlNTKp2JPwE4PgE5Pk8uS/z6Sy6Oji5LAWZLLo4AAAAAGwAAAAAD1APUAAMABwALAA8
AEwAXABsAHwAjACcAKwAvADMANwA7AD8AQwBHAESATwBTAFcAWwBfAGMAZwBrAAALMzUjBzM1Iwcz
NSMFMzUjBzM1IwczNSMlMzUjBTM1IyUzNSMFMzUjJTM1IwczNSMHMzUjBTM1IwczNSMHMzUjJTM1I
wUzNSMlMzUjBTM1IyUzNSMHMzUjBzM1IwMzESMHMzUjBzM1IwczNSMDlj4+fT8/fT8//ok/P30/P3
w+PgNqPj78lj4+A2o+PvyWPj4Daj4+fT8/fT8//ok/P30/P3w+PgNqPj78lj4+A2o+PvyWPj4Daj4
+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+Pj4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj59Pz8/Pj8/
Pz4+Pj4+PvxYA6g+Pj4+Pj4AAgAAAAAD8wPzAAGADAAExc3ETMRfzcBJSE1IbIs5D7kLP7R/isD6
PwYAhYs5v08AsPlLAExbj8AAAAAQAAAAAD8wPzAIoAAAKBISMPHh8fMzUrAS8dPQE/HTMhARcJAQ
JAAS399xIREREREBAQDw8ODg4NDAwLCwoKCQgIBwYFBQQDAgEBAQECAwQFBQYHCAgJCgoLCwWMDQ4
ODg8PEBAQERERERJeXg8ODg4NDg0MDQwMCwsLCgoJCQgIBwCGBgUFAwQCAgEBAgIEAwUFBgYHBwgI
CQkKCgsLCwWMDQwNDg0ODg4PAhD+yygBj5f1A8X+9gEDAwMFBQcHBwgJCgoLCwWMDQ0ODw4QDxAQE
REREReSEREREREAQEA8PDg4ODQwMCwsKCgkICACGBQUEAWIBAT8BAgMDAwUFBgYHBwgICQkKCgsLCw
wMDQwNDg0ODg4PDg8ODQ4NDQ0NDALCwsKCgkJCAgHBwYGBQQEAWMCAf73LwFTAVwAAAAcAAAAAP
UA9QAawAHAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAVwBbAF8AYwBnAGsA
bwAANYe1ISUZNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSMhMzUjBzM1IwczNSMFMzUjBzM1IwczN
SMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjJTM1IwczNSMHMzUjBTM1IwczNSMHMzUjBTM1Iy
wDqPxYA2o+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+AbU+PgG1Pj59Pz99Pz/+iT8/fT8/fD4+AbU+PgG
1Pj7+Sz4+/ks+PgNqPj7+Sz4+/ks+PgNqPj59Pz99Pz/+iT8/fT8/fD4+AbU+Piw+Pj8/Pz8/Pj8/
Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+Pj4+Pj4+Pj4+Pj4+PgAAQAAAAAD1APUAAsAA
AEhFSERMxEhNSERIwHh/koBtj4Btv5KPgIfPv5KAbY+AbYAAwAAAAADdgPzAAcAJABIAAABFSE1Mx
EhESUfBxUzFSE1Mz0BPwg7ARcnDwsjESERIy8ODwIBBgH0Pv2QAVUGBQQHBQIDAX3+in0BAWMEBgU
HCQsNEAdHBQYKCGwLBwMHAWIB+gLu+gECawUFBgGMDgoLCwWMDQwNDAM4fX39EwLteQMEBQOLBg4N
Nj8/JxYKCGkIBwCFBAMBNQIDBwCMDgoGEQsNDPjVA2sMDQsMCwoKDasHBQQEAgEBAgMAAAABgAAA
AAD8wPzAAMAQwBHAICAiWDLAAALITUhBR8PPw8vDw8OASE1KQEfDz8PLw8PDgEhNSE1Hw8/Dy8PDw
4BRQkv/VAH+xwEBAGYEBAYGBgGHCakJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgo
JCQkIBwGGBgYEBAQCAQE4Aq/9Uf7HAQECBAMFBgYGBwgICQkJCgkKCQgJBwGGBgYEBAMDAQEBAQMD
BAQGBgYIBwkICQoJCgkJCQgIBwYGBgUDBAIBATgCr/1R/scBAQIEAwUGBgYHCAgJCQkKCQoJCAkHC
AYGBgQEAWMBAQEBAWMEBAYGBgGHCQgJCgkKCQkJCAgHBgYGBQMEAgFLPh8KCQkICQcIBgYGBAQDAw
EBAQEDAwQEBgYGCACJCAkJCgoJCQgJBwGGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkBTj4KCQkICQc
IBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgoJCQgJBwGGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkB
Lj8fCgkJCAkHCAYGBgQEAWMBAQEBAWMEBAYGBgGHCQgJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEB
gYGCACJCAkJAAIAAAAAAPzA/MAAwAHAAsAEQAVABkAHQAhAAABFSM1IxUjNSMVIzUTMyEVITUBFS

2239

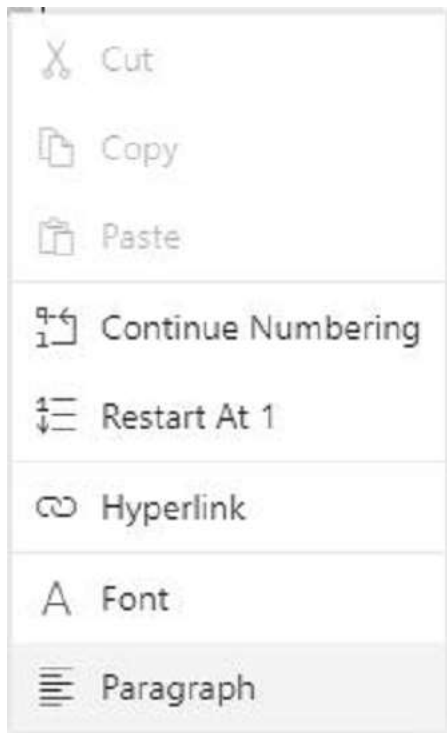
```

1YXN1SW5kZW50FVBpeGVsQWxpZ25DZW50ZXJUYWJsZQ9CYWNRZ3JvdW5kQ29sb3ILQWxpZ25Cb3R0
b20JUGFnZVNldHVwDkhpb2hSaWdodENvbG9yC1N1cGVYc2NyaXB0BVRhYmx1BFVuZG8LSW5zZXJ0Q
mVsb3cJVG9wQm9yZGVyC1BhZ2V0dW1iZXIQQWxpZ25DZW50ZXJUYWJsZQ5JbmNyZWZzZUluZGVudA
RCB2xkCUFsaWduTGVMdAZGb290ZXILSW5zZXJ0UmlnaHQJVW5kZXJsaW5lCkluc2VydExlZnQETG9
jawZIZWFkZXINU3RyaWtldGhyb3VnaAhDbGVhckFsbAtSaWdodEJvcmlRlCgpBbGlnb1JpZ2h0BE9w
ZW4KU3Ryb2t1U2l6ZQVQcmIudAtEZWxldGVUYWJsZQ1Gb250Q29sb3INSW5zaWRlQm9yZGVYcWpEZ
WxldGVsb3dzCERvd25sb2FkC0xpbmVTcGFjaW5nFE1uc2lkZVZlcnRpY2FsQm9yZGVYCEFsaWduVE
9wBFJlZG8MQm90dG9tQm9yZGVyA05ldwVQYXN0ZQdCdWxsZXRxZBENlbGwNRGVsZXRLQ29sdWlucWp
BbGxCb3JkZXJzCVN1YnNjcmlwdBBTaG93SGlkZVByb3BlcnR5DlRhYmx1T2ZDb250ZW50Bkl0YWxp
YxZJbnNpZGVib3Jpem9uZGFsYm9yZGVyC0x1ZnRCb3JkZXJzCU51bWJlcmluZwRMaW5rC0FsaWduQ
2VudGVyC0luc2VydEFib3ZlAAAAAA=) format('truetype');
font-weight: normal;
font-style: normal;
}
[class^="e-de-icon-"],
[class*=" e-de-icon-"] {
font-family: 'Sample browser icons';
}
.e-de-icon-Bullets:before {
content: "\e730";
}
.e-de-icon-Numbering:before {
content: "\e73a";
}
}
</style>

```

Editing numbered list

Document Editor restarts the numbering or continue numbering for a numbered list. These options are found in the built-in context menu, if the list value is selected. Refer to the following screenshot.



See Also

- [List dialog](#)

Table format in React Document editor component

Document Editor customizes the formatting of table, or table cells such as table width, cell margins, cell spacing, background color, and table alignment. This section describes how to customize these formatting for selected cells, rows, or table in detail.

Cell margins

You can customize the cell margins by using the following sample code.

```
`ts
//To change the left margin
documenteditor.selection.cellFormat.leftMargin=5.4;
//To change the right margin
documenteditor.selection.cellFormat.rightMargin=5.4;
//To change the top margin
documenteditor.selection.cellFormat.topMargin=5.4;
//To change the bottom margin
documenteditor.selection.cellFormat.bottomMargin=5.4;
`
```

You can also define the default cell margins for a table. If the specific cell margin value is not defined explicitly in the cell formatting, the corresponding value will be retrieved from default cells margin of the table. Refer to the following sample code.

```
`ts
//To change the left margin
documenteditor.selection.tableFormat.leftMargin=5.4;
//To change the right margin
documenteditor.selection.tableFormat.rightMargin=5.4;
//To change the top margin
documenteditor.selection.tableFormat.topMargin=5.4;
//To change the bottom margin
documenteditor.selection.tableFormat.bottomMargin=5.4;
`
```

Background color

You can explicitly set the background color of selected cells using the following sample code.

```
`ts
```

```
documenteditor.selection.cellFormat.background='#E0E0E0';
```

```
,
```

Refer to the following sample code to customize the background color of the table.

```
`ts
```

```
documenteditor.selection.tableFormat.background='#E0E0E0';
```

```
,
```

Cell spacing

Refer to the following sample code to customize the spacing between each cell in a table.

```
`ts
```

```
documenteditor.selection.tableFormat.cellSpacing=2;
```

```
,
```

Cell vertical alignment

The content is aligned within a table cell to 'Top', 'Center', or 'Bottom'. You can customize this property of selected cells. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.cellFormat.verticalAlignment='Bottom';
```

```
,
```

Table alignment

The tables are aligned in document editor to 'Left', 'Right', or 'Center'. Refer to the following sample code.

```
`ts
```

```
documenteditor.selection.tableFormat.tableAlignment='Center';
```

```
,
```

Cell width

Set the desired width of table cells that will be considered when the table is layouted. Refer to the following sample code.

```
`ts
```

```
documentEditor.selection.cellFormat.preferredWidth = 100;
```

```
,
```

Table width

You can set the desired width of a table in 'Point 'or 'Percent' type. Refer to the following sample code.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, SfddExport, Selection, Editor } from '@syncfusion/ej2-react-documenteditor';
```

```
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    createTable();
  }, []);
  function createTable() {
    documenteditor.editor.insertTable(2, 2);
    //To change the width of a table
    documenteditor.selection.tableFormat.preferredWidthType = 'Point';
    documenteditor.selection.tableFormat.preferredWidth = 300;
  }
  return (
    <DocumentEditorComponent
      id="container"
      height={'330px'}
      ref={scope => {
        documenteditor = scope;
      }}
      isReadOnly={false}
      enableSelection={true}
      enableEditor={true}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Apply borders

Document Editor exposes API to customize the borders for table cells by specifying the settings. Refer to the following sample code.

```
`ts
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor, BorderSettings, } from
 '@syncfusion/ej2-react-documenteditor';

//Inject require module.

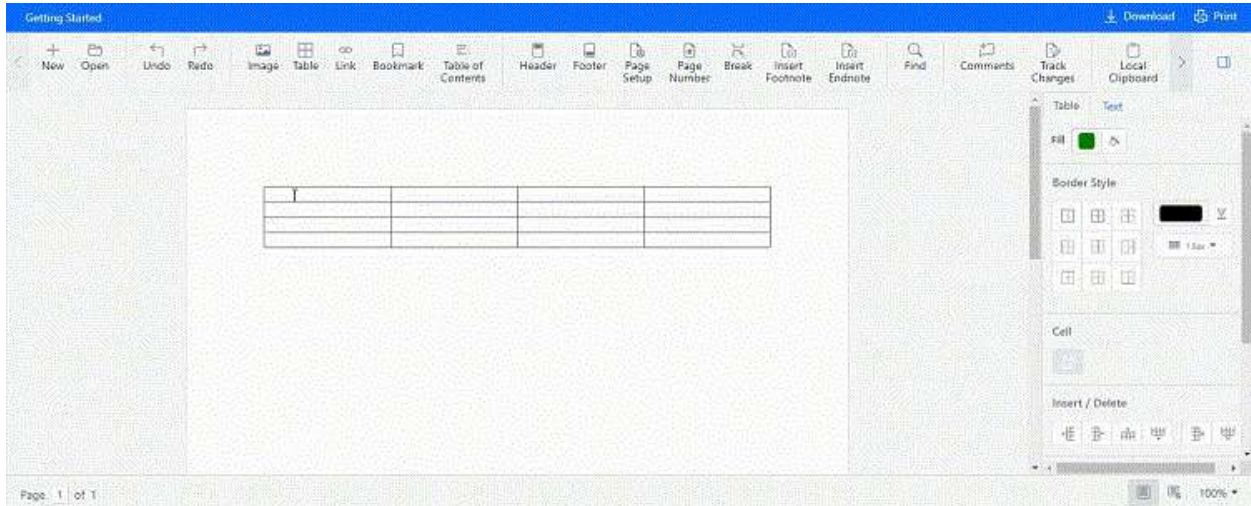
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);

function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function componentDidMount() {
    documenteditor.editor.insertTable(2, 2);
    //Border settings.
    let borderSettings: BorderSettings = {
      type: 'AllBorders',
      lineWidth: 12,
    };
    //Apply border.
    documenteditor.editor.applyBorders(borderSettings);
  }
  return (
    <DocumentEditorComponent
      id="container"
      height={'330px'}
      ref={scope => {
        documenteditor = scope;
      }}
      isReadOnly={false}
      enableSelection={true}
      enableEditor={true}
    />
  );
}
```


export default App;

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Please check below gif which illustrates how to apply border for selected cells through properties pane options - border color, line size and no border:



Working with row formatting

Document Editor allows various row formatting such as height and repeat header.

Row height

You can customize the height of a table row as 'Auto', 'AtLeast', or 'Exactly'. Refer to the following sample code.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor } from '@syncfusion/ej2-react-documenteditor';

//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);

function App() {
  let documenteditor: DocumentEditorComponent;

  React.useEffect(() => {
    componentDidMount()
  }, []);

  function componentDidMount() {
    //Insert table.
    documenteditor.editor.insertTable(2, 2);
```

```
//To change the width of a table
documenteditor.selection.rowFormat.heightType = 'Exactly';
documenteditor.selection.rowFormat.height = 20;
}
return (
<DocumentEditorComponent
id="container"
ref={scope => {
documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Header row

The header row describes the content of a table. A table can optionally have a header row. Only the first row of a table can be the header row. If the cursor position is at first row of the table, then you can define whether it as header row or not, using the following sample code.

```
`ts
documenteditor.selection.rowFormat.isHeader=true;
`
```

Allow row break across pages

This property is valid if a table row does not fit in the current page during table layout. It defines whether a table row can be allowed to break. If the value is false, the entire row will be moved to the start of next page. You can modify this property for selected rows using the following sample code.

```
`ts
documenteditor.selection.rowFormat.allowRowBreakAcrossPages=false;
`
```

Title

Document Editor expose API to get or set the table title of the selected table. Refer to the following sample code to set title.

```
`ts
documenteditor.selection.tableFormat.title = 'Shipping Details';
`
```

Description

Document Editor expose API to get or set the table description of the selected image. Refer to the following sample code to set description.

```
`ts
documenteditor.selection.tableFormat.description = 'Freight cost and shipping details';
`
```

See Also

- [Table properties dialog](#)

Section format in React Document editor component

Document Editor supports various section formatting such as page size, page margins, and more.

Page size

You can get or set the size of a section at cursor position by using the following sample code.

```
`ts
documenteditor.selection.sectionFormat.pageWidth = 500;
documenteditor.selection.sectionFormat.pageHeight = 600;
`
```

You can change the orientation of the page by swapping the values of page width and height respectively.

Page margins

Left and right page margin defines the gap between the document content from left and right side of the page respectively. Top and bottom page margins defines the gap between the document content from header and footer of the page respectively.

Refer to the following sample code.

```
`ts
documenteditor.selection.sectionFormat.leftMargin = 10;
documenteditor.selection.sectionFormat.rightMargin = 10;
documenteditor.selection.sectionFormat.bottomMargin = 10;
documenteditor.selection.sectionFormat.topMargin = 10;
`
```

Header distance

You can define the distance of header content from the top of the page by using the following sample code.

```
`ts
```

```
documentEditor.selection.sectionFormat.headerDistance = 72;
```

```
,
```

Footer distance

You can define the distance of footer content from the bottom of the page by using the following sample code.

```
`ts
```

```
documentEditor.selection.sectionFormat.footerDistance = 72;
```

```
,
```

Columns

You can define the number of columns, column width, and space between columns for the pages in a section.

The following code example illustrates how to define the two columns layout for the pages in a section.

```
`typescript
```

```
var column = new SelectionColumnFormat(documentEditor.selection);
```

```
column.width = 216;
```

```
column.space = 36;
```

```
documentEditor.selection.sectionFormat.columns = [column, column];
```

```
documentEditor.selection.sectionFormat.lineBetweenColumns = true;
```

```
,
```

Breaks

You can insert Column break

The following code indicate that the text following the column break will begin in the next column

```
`typescript
```

```
documentEditor.editor.insertColumnBreak();
```

```
,
```

You can insert next page section break to start the new section on the next page

The following code example illustrates how to insert a next page section break

```
`typescript
```

```
documentEditor.editor.insertSectionBreak(SectionBreakType.NewPage);
```

```
,
```

You can insert continuous section break to start the new section on the same page

The following code example illustrates how to insert a continuous section break

```
`typescript
```

```
documentEditor.editor.insertSectionBreak(SectionBreakType.Continuous);
```

`

See Also

[*Page setup dialogLink to the Video](#)

Comments in React Document editor component

Document Editor allows you to add comments to documents. You can add, navigate and remove comments in code and from the UI.

To know more about the comments in `DocumentEditor` component, you can check the video below.

Add a new comment

Comments can be inserted to the selected text.

`ts

```
documentEditor.editor.insertComment("Test comment");
```

`

Comment navigation

Next and previous comments can be navigated using the below code snippet.

`ts

```
//Navigate to next comment
```

```
documentEditor.selection.navigateNextComment();
```

```
//Navigate to previous comment
```

```
documentEditor.selection.navigatePreviousComment();
```

`

Delete comment

Current comment can be deleted using the below code snippet.

`ts

```
documentEditor.editor.deleteComment();
```

`

Delete all comment

All the comments in the document can be deleted using the below code snippet.

`ts

```
documentEditor.editor.deleteAllComments();
```

`

Protect the document in comments only mode

Document Editor provides support for protecting the document with `CommentsOnly` protection. In this protection, user allowed to add or edit comments alone in the document.

Document editor provides an option to protect and unprotect document using [enforceProtection](#) and [stopProtection](#) API.

The following example code illustrates how to enforce and stop protection in Document editor container.

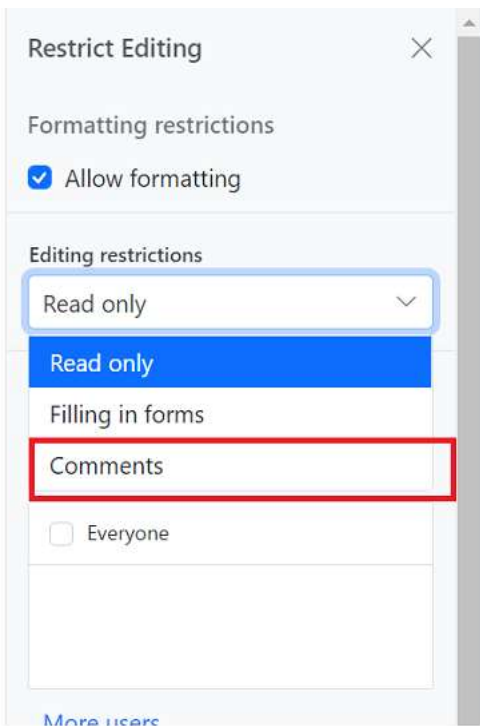
```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    EnforceProtection();
    StopProtection();
  }, []);
  function EnforceProtection(){
    //enforce protection
    container.documentEditor.editor.enforceProtection('123','CommentsOnly');
  }
  function StopProtection(){
    //stop the document protection
    container.documentEditor.editor.stopProtection('123');
  }
  return (
    <div>
      <button onClick={EnforceProtection}>enforceProtection</button>
      <button onClick={StopProtection}>stopProtection</button>
      <DocumentEditorContainerComponent
        id="container"
        ref={(scope) => {
          container = scope;
        }}
      />
    </div>
  );
}
```

```

height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
/>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Comment only protection can be enabled in UI by using [Restrict Editing pane](#)



Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are `NoProtection` | `ReadOnly` | `FormFieldsOnly` | `CommentsOnly`. In stop protection method, parameter denotes the password.

Mention support in Comments

Mention support displays a list of items that users can select or tag from the suggested list. To use this feature, type the @ character in the comment box and select or tag the user from the suggestion list.

The following example illustrates how to enable mention support in the Document Editor

```

`ts
import * as ReactDOM from 'react-dom';

```

```
import * as React from 'react';

import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';

DocumentEditorContainerComponent.Inject(Toolbar);

function App() {
  let mentionData: any = [
    { "Name": "Selma Rose", "Eimg": "3", "EmailId": "selma@mycompany.com" },
    { "Name": "Russo Kay", "Eimg": "8", "EmailId": "russo@mycompany.com" },
    { "Name": "Camden Kate", "Eimg": "9", "EmailId": "camden@mycompany.com" },
    { "Name": "Mary Kate", "Eimg": "4", "EmailId": "marry@mycompany.com" },
    { "Name": "Ursula Ann", "Eimg": "2", "EmailId": "ursula@mycompany.com" },
    { "Name": "Margaret", "Eimg": "5", "EmailId": "margaret@mycompany.com" },
    { "Name": "Laura Grace", "Eimg": "6", "EmailId": "laura@mycompany.com" },
    { "Name": "Robert", "Eimg": "8", "EmailId": "robert@mycompany.com" },
    { "Name": "Albert", "Eimg": "9", "EmailId": "albert@mycompany.com" },
    { "Name": "Michale", "Eimg": "10", "EmailId": "michale@mycompany.com" },
    { "Name": "Andrew James", "Eimg": "7", "EmailId": "james@mycompany.com" },
    { "Name": "Rosalie", "Eimg": "4", "EmailId": "rosalie@mycompany.com" },
    { "Name": "Stella Ruth", "Eimg": "2", "EmailId": "stella@mycompany.com" },
    { "Name": "Richard Rose", "Eimg": "10", "EmailId": "richard@mycompany.com" },
    { "Name": "Gabrielle", "Eimg": "3", "EmailId": "gabrielle@mycompany.com" },
    { "Name": "Thomas", "Eimg": "7", "EmailId": "thomas@mycompany.com" },
    { "Name": "Charles Danny", "Eimg": "8", "EmailId": "charles@mycompany.com" },
    { "Name": "Daniel", "Eimg": "10", "EmailId": "daniel@mycompany.com" },
    { "Name": "Matthew", "Eimg": "7", "EmailId": "matthew@mycompany.com" },
    { "Name": "Donald Krish", "Eimg": "9", "EmailId": "donald@mycompany.com" },
    { "Name": "Yohana", "Eimg": "1", "EmailId": "yohana@mycompany.com" },
    { "Name": "Kevin Paul", "Eimg": "10", "EmailId": "kevin@mycompany.com" },
    { "Name": "Andrew Fuller", "Eimg": "3", "EmailId": "andrew@mycompany.com" }
  ];

  let settings = {showRuler: true, mentionSettings: { dataSource: mentionData, fields: { text: 'Name' } }};
```



```

return (
  <DocumentEditorContainerComponent
    id="container"
    height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
    enableToolbar={true}
    documentEditorSettings={settings}
  />
);
}

export default App;

ReactDOM.render(<App />, document.getElementById('root'));

```

[Link to the Video`](#)

Track changes in React Document editor component

Track Changes allows you to keep a record of changes or edits made to a document. You can then choose to accept or reject the modifications. It is a useful tool for managing changes made by several reviewers to the same document. If track changes option is enabled, all editing operations are preserved as revisions in Document Editor.

To know more about the track changes in DocumentEditor component, you can check the video below.

Enable track changes in Document Editor

The following example demonstrates how to enable track changes.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import {
  DocumentEditorComponent, DocumentEditor
} from '@syncfusion/ej2-react-documenteditor';

function App() {
  let documentEditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  React.useEffect(() => {
    ComponentDidMount()
  }, []);
  function ComponentDidMount() {
    documentEditor.spellChecker.languageID = 1033;
    //LCID of "en-us";
  }
}

```

```

documentEditor.spellChecker.removeUnderline = false;
documentEditor.spellChecker.allowSpellCheckAndSuggestion = true;
}
return (
  <DocumentEditorComponent id="container" ref={{scope}} => { documentEditor = scope; }}
  enableTrackChanges={true} enableSpellCheck={true} />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Get all tracked revisions

The following example demonstrate how to get all tracked revision from current document.

```

`ts
<DocumentEditorComponent id="container" ref={{scope}} => { this.documenteditor = scope; }}
enableTrackChanges={true}/>
/

```

- Get revisions from the current document

```
*/
```

```
let revisions : RevisionCollection = this.container.documentEditor.revisions;
```

```
`
```

Accept or Reject all changes programmatically

The following example demonstrates how to accept/reject all changes.

```

`ts
<DocumentEditorComponent id="container" ref={{scope}} => { this.documenteditor = scope; }}
enableTrackChanges={true}/>
/

```

- Get revisions from the current document

```
*/
```

```
let revisions : RevisionCollection = this.container.documentEditor.revisions;
```

```
/
```

- Accept all tracked changes

```
*/  
revisions.acceptAll();  
/
```

- Reject all tracked changes

```
*/  
revisions.rejectAll();  
`
```

[Accept or reject a specific revision](#)

The following example demonstrates how to accept/reject specific revision in the document editor.

```
`ts  
/  
  
    • Get revisions from the current document  
  
*/  
let revisions : RevisionCollection = this.container.documentEditor.revisions;  
/
```

- Accept specific changes

```
*/  
revisions.get(0).accept();  
/
```

- Reject specific changes

```
*/  
revisions.get(1).reject();  
`
```

[Navigate between the tracked changes](#)

The following example demonstrates how to navigate tracked revision programmatically.

```
`ts  
/  
  
    • Navigate to next tracked change from the current selection.  
  
*/  
this.container.documentEditor.selection.navigateNextRevision();
```

/

- Navigate to previous tracked change from the current selection.

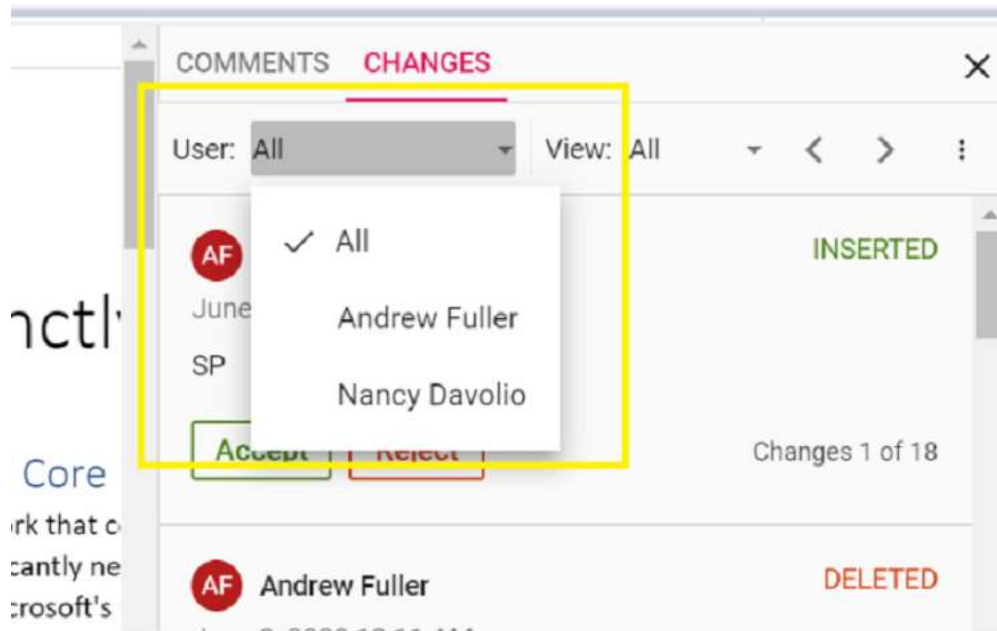
*/

```
this.container.documentEditor.selection.navigatePreviousRevision();
```

,

Filtering changes based on user

In DocumentEditor, we have built-in review panel in which we have provided support for filtering changes based on the user.



Protect the document in track changes only mode

Document Editor provides support for protecting the document with **RevisionsOnly** protection. In this protection, all the users are allowed to view the document and do their corrections, but they cannot accept or reject any tracked changes in the document. Later, the author can view their corrections and accept or reject the changes.

Document editor provides an option to protect and unprotect document using [enforceProtection](#) and [stopProtection](#) API.

The following example code illustrates how to enforce and stop protection in Document editor container.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import {
```

```
  DocumentEditorContainerComponent,
```

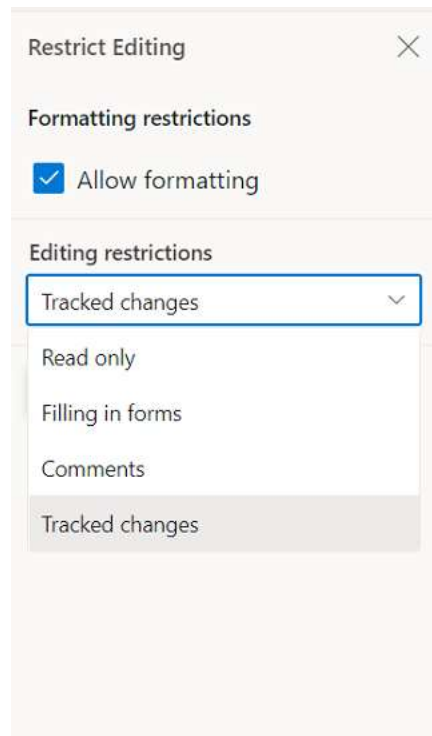
```
Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    EnforceProtection();
    StopProtection();
  }, []);
  function EnforceProtection() {
    //enforce protection
    container.documentEditor.editor.enforceProtection('123', 'CommentsOnly');
  }
  function StopProtection() {
    //stop the document protection
    container.documentEditor.editor.stopProtection('123');
  }
  return (
    <div>
      <button onClick={EnforceProtection}>enforceProtection</button>
      <button onClick={StopProtection}>stopProtection</button>
      <DocumentEditorContainerComponent
        id="container"
        ref={(scope) => {
          container = scope;
        }}
        height={'590px'}
        serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
        enableToolbar={true}
      />
    </div>
  );
}
```

```
export default App;
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

```
,
```

Tracked changes only protection can be enabled in UI by using [Restrict Editing pane](#)



Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are `NoProtection` | `ReadOnly` | `FormFieldsOnly` | `CommentsOnly` | `RevisionsOnly`. In stop protection method, parameter denotes the password.

Event

You can restrict the accept and reject changes based on the author name. The following example demonstrates how to restrict an author from accept/reject changes.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
const App = () => {
  // Event gets triggered before accepting/rejecting changes
  const beforeAcceptRejectChanges = (args) => {
```

```
// Check the author of the revision
if (args.author !== 'Hary') {
  // Cancel the accept/reject action
  args.cancel = true;
}
};

return (
  <div>
    <div>
      <div>
        <DocumentEditorContainerComponent
          ref={{(scope) => { container = scope; }}}
          style={{ display: 'block' }}
          height={'590px'}
          beforeAcceptRejectChanges={beforeAcceptRejectChanges}
          enableToolbar={true}
        />
      </div>
    </div>
  </div>
);
};

export default App;
`
```

Fields in React Document editor component

Document Editor has preservation support for all types of fields in an existing word document without any data loss.

Adding Fields

You can add a field to the document by using [insertField](#) method in [Editor](#) module.

The following example code illustrates how to insert merge field programmatically by providing the field code and field result.

```
`ts
let fieldCode: string = 'MERGEFIELD First Name \* MERGEFORMAT ';
let fieldResult: string = '«First Name»';
```

```
documenteditor.editor.insertField(fieldCode, fieldResult);
```

```
,
```

Note: Document editor does not validate/process the field code/field result. it simply inserts the field with specified field information.

Update fields

Document Editor provides support for updating bookmark cross reference field. The following example code illustrates how to update bookmark cross reference field.

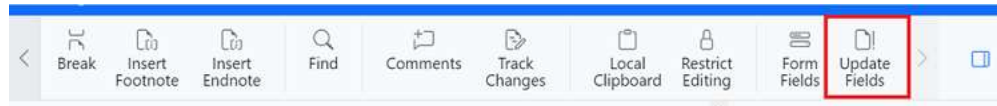
```
`ts
```

```
//Update all the bookmark cross reference field in the document.
```

```
documenteditor.updateFields();
```

```
,
```

Bookmark cross reference fields can be updated through UI by using update fields option in **Toolbar**.



The following type of fields are automatically updated in Document Editor.

- Numpages
- Section
- Page

Get field info

You can get field code and field result of the current selected field by using [getFieldInfo](#) method in the [Selection](#) module.

```
`ts
```

```
//Gets the field information of the selected field.
```

```
let fieldInfo: FieldInfo = documenteditor.selection.getFieldInfo();
```

```
,
```

Note: For nested fields, this method returns combined field code and result.

Set field info

You can modify the field code and field result of the current selected field by using [setFieldInfo](#) method in the [Editor](#) module.

```
`ts
```

```
//Gets the field information for the selected field.
```

```
let fieldInfo: FieldInfo = documenteditor.selection.getFieldInfo();
```

```
//Modify field code
```

```
fieldInfo.code = 'MERGEFIELD First Name \* MERGEFORMAT ';
```



```
//Modify field result
fieldInfo.result = '«First Name»';

//Modify field code and result of the current selected field.
documenteditor.editor.setFieldInfo(fieldInfo);
`
```

Note: For nested field, entire field gets replaced completely with the specified field information.

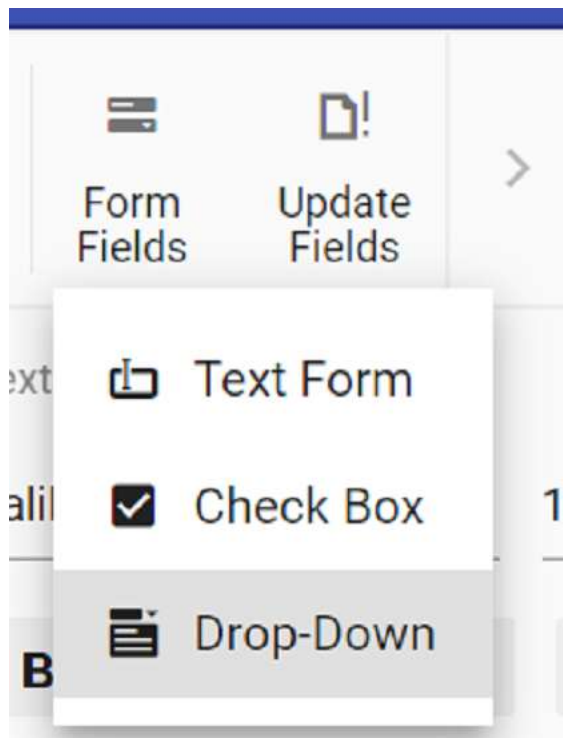
See Also

[Mail merge using Word library \(DocIO\)](#)

[Mail merge demo](#)

Form fields in React Document editor component

Document Editor Container component provide support for inserting Text, CheckBox, DropDown form fields through in-built toolbar.



Insert form field

Form fields can be inserted using [insertFormField](#) method in editor module.

```
`ts
```

```
//Insert Text form field
```

```
documentEditor.editor.insertFormField('Text');
```

```
//Insert Checkbox form field
```

```
documentEditor.editor.insertFormField('CheckBox');
```

```
//Insert Drop down form field
```

```
documentEditor.editor.insertFormField('Dropdown');
```

```
,
```

Get form field names

All the form fields names form current document can be retrieved using [getFormFieldNames\(\)](#).

```
`ts
```

```
let formFieldsNames: string[] = documentEditor.getFormFieldNames();
```

```
,
```

Get form field properties

Form field properties can be retrieved using [getFormFieldInfo\(\)](#).

```
`ts
```

```
//Text form field
```

```
let textfieldInfo: TextFormFieldInfo = documentEditor.getFormFieldInfo('Text1') as TextFormFieldInfo;
```

```
//Checkbox form field
```

```
let checkboxfieldInfo: CheckBoxFormFieldInfo = documentEditor.getFormFieldInfo('Check1') as  
CheckBoxFormFieldInfo;
```

```
//Dropdown form field
```

```
let dropdownfieldInfo: DropDownFormFieldInfo = documentEditor.getFormFieldInfo('Drop1') as  
DropDownFormFieldInfo;
```

```
,
```

Set form field properties

Form field properties can be modified using [setFormFieldInfo](#).

```
`ts
```

```
// Set text form field properties
```

```
let textfieldInfo: TextFormFieldInfo = documentEditor.getFormFieldInfo('Text1') as TextFormFieldInfo;
```

```
textfieldInfo.defaultValue = "Hello";
```

```
textfieldInfo.format = "Uppercase";
```

```
textfieldInfo.type = "Text";
```

```
textfieldInfo.name = "Text2";
```

```
documentEditor.setFormFieldInfo('Text1',textfieldInfo);
```

```
// Set checkbox form field properties
```

```
let checkboxfieldInfo: CheckBoxFormFieldInfo = documentEditor.getFormFieldInfo('Check1') as  
CheckBoxFormFieldInfo;
```

```
checkboxboxfieldInfo.defaultValue = true;
```

```
checkboxboxfieldInfo.name = "Check2";
```

```
documentEditor.setFormFieldInfo('Check1',checkboxboxfieldInfo);
```

```
// Set checkbox form field properties
```

```
let dropdownfieldInfo: DropDownFormFieldInfo = documentEditor.getFormFieldInfo('Drop1') as DropDownFormFieldInfo;
```

```
dropdownfieldInfo.dropDownItems = ['One','Two', 'Three'];
```

```
dropdownfieldInfo.name = "Drop2";
```

```
documentEditor.setFormFieldInfo('Drop1',dropdownfieldInfo);
```

```
,
```

Note:If a form field already exists in the document with the new name specified, the old form field name property will be cleared and it will not be accessible. Ensure the new name is unique.

Export form field data

Data of the all the Form fields in the document can be exported using [exportFormData](#).

```
`ts
```

```
let formFieldDate: FormFieldData[] = documentEditor.exportFormData();
```

```
,
```

Import form field data

Form fields can be prefilled with data using [importFormData](#).

```
`ts
```

```
let textformField: FormFieldData = {fieldName: 'Text1', value: 'Hello World'};
```

```
let checkformField: FormFieldData = {fieldName: 'Check1', value: true};
```

```
let dropdownformField: FormFieldData = {fieldName: 'Drop1', value: 1};
```

```
//Import form field data
```

```
documentEditor.importFormData([textformField,checkformField,dropdownformField]);
```

```
,
```

Reset form fields

Reset all the form fields in current document to default value using [resetFormFields](#).

```
`ts
```

```
documentEditor.resetFormFields();
```

```
,
```

Protect the document in form filling mode

Document Editor provides support for protecting the document with **FormFieldsOnly** protection. In this protection, user can only fill form fields in the document.

Document editor provides an option to protect and unprotect document using [enforceProtection](#) and [stopProtection](#) API.

The following example code illustrates how to enforce and stop protection in Document editor container.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    EnforceProtection();
    StopProtection();
  }, []);
  function EnforceProtection(){
    //enforce protection
    container.documentEditor.editor.enforceProtection('123','CommentsOnly');
  }
  function StopProtection(){
    //stop the document protection
    container.documentEditor.editor.stopProtection('123');
  }
  return (
    <div>
      <button onClick={EnforceProtection}>enforceProtection</button>
      <button onClick={StopProtection}>stopProtection</button>
      <DocumentEditorContainerComponent
        id="container"
        ref={(scope) => {
          container = scope;
        }}
        height={'590px'}
        serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
        enableToolbar={true}>
```

```

/>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Note: In enforce Protection method, first parameter denotes password and second parameter denotes protection type. Possible values of protection type are `NoProtection` | `ReadOnly` | `FormFieldsOnly` | `CommentsOnly`. In stop protection method, parameter denotes the password.

Clipboard in React Document editor component

Document Editor takes advantage of system clipboard and allows you to copy or move a portion of the document into it in HTML format, so that it can be pasted in any application that supports clipboard.

Copy

Copy a portion of document to system clipboard using built-in context menu of document editor. You can also do it programmatically using the following sample code.

```

`ts
documentEditor.selection.copy();
`

```

Cut

Cut a portion of document to system clipboard using built-in context menu of document editor. You can also do it programmatically using the following sample code.

```

`ts
documentEditor.editor.cut();
`

```

Paste

Due to limitations, you can paste contents from system clipboard in document editor only using the 'CTRL + V' keyboard shortcut.

Note: Due to browser limitation of getting content from system clipboard, paste using API and context menu option doesn't work.

Local paste (copy/paste within control)

Document Editor expose API to enable local paste within the control. On enabling this, the following is performed:

- Selected contents will be stored to an internal clipboard as SFDT in addition to system clipboard.
- Clipboard paste will be overridden, and internally stored data (SFDT data) that has formatted text will be pasted using `paste()` API in Document editor.

Refer to the following sample code.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfddExport, Selection, Editor } from '@syncfusion/ej2-react-
documenteditor';

DocumentEditorComponent.Inject(SfddExport, Selection, Editor);

function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function componentDidMount() {
    //Enable document editor local paste option.
    documenteditor.enableLocalPaste = true;
  }
  return (
    <DocumentEditorComponent id="container" height={'330px'} ref={(scope) => { documenteditor = scope;
    }} isReadOnly={false} enableSelection={true} enableEditor={true} />
  );
}

export default App

ReactDOM.render(<App />, document.getElementById('sample'));
```

By default, **enableLocalPaste** is false.

When local paste is enabled for a document editor instance, you can paste contents programmatically if the internal clipboard has stored data during last copy operation. Refer to the following sample code.

```
`ts
documentEditor.editor.paste();
```

Paste options in context menu

In Document editor, paste options in context menu will be in disabled state if you were try to copy/paste content from outside of Document editor. It gets enabled when **enableLocalPaste** is true and trying to copy/paste content inside Document editor.

Note: Due to browser limitation of getting content from system clipboard, paste using API and context menu option doesn't work. Hence, the paste option is disabled in context menu.

Alternatively, you can use the keyboard shortcuts,

- Cut: Ctrl + X
- Copy: Ctrl + C
- Paste: Ctrl + V

EnableLocalPaste behaviour

|EnableLocalPaste |Paste behavior details|

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

|True |Allows to paste content that is copied from the same Document Editor component alone and prevents pasting content from system clipboard. Hence the content copied from outside Document Editor component can't be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will be resolved. So, you can copy and paste content within the Document Editor component using API and context menu options too. |

|False|Allows to paste content from system clipboard. Hence the content copied from both the Document Editor component and outside can be pasted.
Browser limitation of pasting from system clipboard using API and context menu options, will remain as a limitation. |

Note:

- Keyboard shortcut for pasting will work properly in both cases.
- Copying content from Document Editor component and pasting outside will work properly in both cases.

Paste with formatting

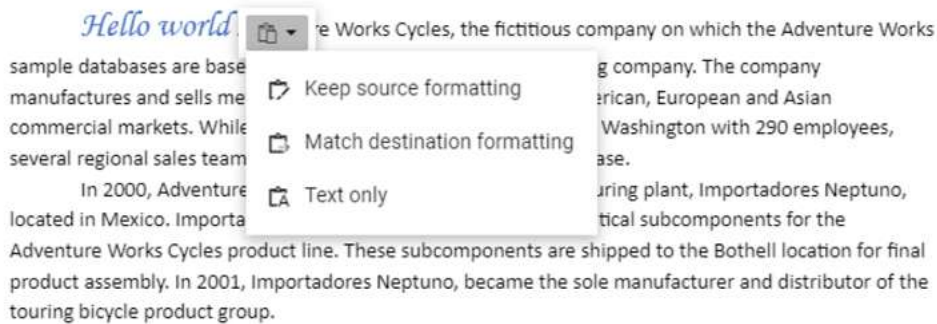
Document Editor provides support to paste the system clipboard data with formatting. To enable clipboard paste with formatting options, set the `enableLocalPaste` property in Document Editor to false and use this .NET Standard library [SynCFusion.EJ2.WordEditor.AspNet.Core](#) for Core by the web API service implementation. This library helps you to paste the system clipboard data with formatting.

You can paste your system clipboard data in the following ways:

- **Keep Source Formatting** This option retains the character styles and direct formatting applied to the copied text. Direct formatting includes characteristics such as font size, italics, or other formatting that is not included in the paragraph style.
- **Match Destination Formatting** This option discards most of the formatting applied directly to the copied text, but it retains the formatting applied for emphasis, such as bold and italic when it is applied to only a portion of the selection. The text takes on the style characteristics of the paragraph where it is pasted. The text also takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted.
- **Text Only** This option discards all formatting and non-text elements such as pictures or tables. The text takes on the style characteristics of the paragraph where it is pasted and takes on any direct formatting or character style properties of text that immediately precedes the cursor when the text is pasted. Graphical elements are discarded and tables are converted to a series of paragraphs.

This paste option appears as follows.

Adventure Works Cycles



See Also

- [Feature modules](#)
- [Keyboard shortcuts](#)

History in React Document editor component

Document Editor tracks the history of all editing actions done in the document, which allows undo and redo functionality.

Enable or disable history

Inject the 'EditorHistory' module in your application to provide history preservation functionality for 'DocumentEditor'. Refer to the following code example.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor, EditorHistory } from
'@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, EditorHistory);

function App() {
  let documenteditor: DocumentEditorComponent;

  React.useEffect(() => {
    ComponentDidMount()
  }, []);

  function ComponentDidMount() {
    //Enable history module.
    documenteditor.enableEditorHistory = true;
  }
}
```



```
return (  
<DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor = scope;  
}} isReadOnly={false} enableSelection={true} enableEditor={true} />  
);  
}  
  
export default App;  
  
ReactDOM.render(<App />, document.getElementById('sample'));  
`
```

You can enable or disable history preservation for a document editor instance any time using the 'enableEditorHistory' property. Refer to the following sample code.

```
`ts  
editor.enableEditorHistory = false;  
`
```

Undo and redo

You can perform undo and redo by 'CTRL+Z' and 'CTRL+Y' keyboard shortcuts. Document Editor exposes API to do it programmatically.

To undo the last editing operation in document editor, refer to the following sample code.

```
`ts  
editor.editorHistory.undo();  
`
```

To redo the last undone action, refer to the following code example.

```
`ts  
editor.editorHistory.redo();  
`
```

Stack size

History of editing actions will be maintained in stack, so that the last item will be reverted first. By default, document editor limits the size of undo and redo stacks to 500 each respectively. However, you can customize this limit. Refer to the following sample code.

```
`ts  
editor.editorHistory.undoLimit = 400;  
editor.editorHistory.redoLimit = 400;  
`
```

See Also

- [Feature modules](#)
- [Keyboard shortcuts](#)

Find and replace in React Document editor component

The document editor component searches a portion of text in the document through a built-in interface called **OptionsPane** or rich APIs. When used in combination with selection performs various operations on the search results like replacing it with some other text, highlighting it, making it bolder, and more.

Options pane

This provides the options to search for a portion of text in the document. After search operation is completed, the search results will be displayed in a list and options to navigate between them. The current occurrence of matched text or all occurrences with another text can be replaced by switching to **Replace** tab. This pane is opened using the keyboard shortcut **CTRL+F**. You can also open it programmatically using the following sample code.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, SfdtExport, Selection, Editor, BordersAndShadingDialog } from
 '@syncfusion/ej2-react-documenteditor';

//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, BordersAndShadingDialog);

function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    ComponentDidMount()
  }, []);
  function ComponentDidMount() {
    let sfdt: string = `{
      "sections": [
        {
          "blocks": [
            {
              "inlines": [
                {
                  "characterFormat": {
                    "bold": true,
                    "italic": true
                  },
                  "text": "Adventure Works Cycles, the fictitious company on which the AdventureWorks sample
databases are based, is a large, multinational manufacturing company. The company manufactures and
```

sells metal and composite bicycles to North American, European and Asian commercial markets. While its base operation is located in Bothell, Washington with 290 employees, several regional sales teams are located throughout their market base."

```

}
]
}
]
}
]
}';
//Open the document in Document Editor.
documenteditor.open(sfedt);
}
function ShowHideOptionsPane() {
//Open options pane.
documenteditor.showOptionsPane();
}
return (
<div>
<button onClick={ShowHideOptionsPane}>OptionsPane</button>
<DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableSearch={true}
enableOptionsPane={true} />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

You can close the options pane by pressing **Esc** key.

Search

The [Search](#) module of Document Editor exposes the following APIs:

API Name	Type	Description
---	---	---
findAll()	Method	Searches for specified text in the whole document and highlights it with yellow.

[|searchResults](#) | Property | This is an instance of [SearchResults](#). |

[|find\(\)](#) | Method | Find immediate occurrence of specified text from cursor position in the document and highlights it with yellow. |

Find the immediate occurrence in the document

Using [find\(\)](#) method, you can find the immediate occurrence of specified text from current cursor position in the document.

The following example code illustrates how to use find in Document editor.

```
`ts
this.documenteditor.search.find('Some text', 'None');
`
```

Note: Second parameter is optional parameter and it denotes find Options. Possible values of find options are 'None' | 'WholeWord' | 'CaseSensitive' | 'CaseSensitiveWholeWord'.

Find all the occurrences in the document

Using [findAll\(\)](#) method, you can find all the occurrences of specified text in the whole document and highlight it with yellow.

The following example code illustrates how to find All the text in the document.

```
`ts
this.documenteditor.search.findAll('Some text', 'None');
`
```

Note: Second parameter is optional parameter and it denotes find Options. Possible values of find options are 'None' | 'WholeWord' | 'CaseSensitive' | 'CaseSensitiveWholeWord'.

Search results

The [SearchResults](#) class provides information about the search results after a search operation is completed that can be identified using the [searchResultsChange](#) event. This will expose the following APIs:

|API Name|Type |Description|

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

|[length](#)|Property|Returns the total number of results found on the search. |

|[index](#)|Property|Returns the index of selected search result. You can change the value for this property to move the selection. |

|[replaceAll\(\)](#)|Method|Replaces all the occurrences with specified text. |

|[clear\(\)](#)|Method|Clears the search result. |

Replace all the occurrences

Using [replaceAll](#), you can replace all the occurrences with specified text.

The following example code illustrates how to use replace All in Document editor.

```
`ts
this.documentEditor.search.findAll ('Some text');
```

```
// Replace all the searched text with word 'Mike'
this.documentEditor.search.searchResults.replaceAll("Mike");
`
```

Replace

Using [insertText](#), you can replace the current searched text with specified text and it replace single occurrence.

Note: This [insertText](#) API accepts following control characters

- * New line characters ("r", "r\n", "n") - Inserts a new paragraph and appends the remaining text to the new paragraph.
- * Line break character ("v") - Moves the remaining text to start in new line.
- * Tab character ("t") - Allocates a tab space and continue the next character.

The following example code illustrates how to find a text in the document and replace each occurrence of the text one by one programmatically.

```
`ts
this.container.documentEditor.search.findAll('works');
let searchLength: number = container.documentEditor.search.searchResults.length;
for (let i = 0; i < searchLength; i++) {
// It will move selection to specific searched index,move to each occurrence one by one
this.container.documentEditor.search.searchResults.index = i;
// Replace it with some text
this.container.documentEditor.editor.insertText('Hello');
}
container.documentEditor.search.searchResults.clear();
`
```

Customize find and replace

Using the exposed APIs, you can customize the find and replace functionality in your application. Refer to the following sample code.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor, Search } from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, Search);
function App() {
```

```
let documenteditor: DocumentEditorComponent;

React.useEffect(() => {
  ComponentDidMount()
}, []);

function ComponentDidMount() {
  let sfdt: string = `{
    "sections": [
      {
        "blocks": [
          {
            "inlines": [
              {
                "characterFormat": {
                  "bold": true,
                  "italic": true
                },
                "text": "Adventure Works Cycles, the fictitious company on which the AdventureWorks sample
databases are based, is a large, multinational manufacturing company. The company manufactures and
sells metal and composite bicycles to North American, European and Asian commercial markets. While
its base operation is located in Bothell, Washington with 290 employees, several regional sales teams
are located throughout their market base."
              }
            ]
          }
        ]
      }
    ]
  }`;

  //Open the document in Document Editor.
  documenteditor.open(sfdt);
}

function replaceAll() {
  let textToFind: string = (document.getElementById('find_text') as HTMLInputElement).value;
  let textToReplace: string = (document.getElementById('replace_text') as HTMLInputElement).value;
```

```

if (textToFind !== '') {
  // Find all the occurrences of given text
  documenteditor.searchModule.findAll(textToFind);
  if (documenteditor.searchModule.searchResults.length > 0) {
    // Replace all the occurrences of given text
    documenteditor.searchModule.searchResults.replaceAll(textToReplace);
  }
}
}

return (
  <div>
    <button onClick={replaceAll}>Replace All</button>
    <DocumentEditorComponent id="container" height={'590px'} ref={{scope} => { documenteditor = scope;
    }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSearch={true} />
  </div>
);
}

export default App

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

See Also

- [Options pane](#)
- [Feature modules](#)

Keyboard shortcut in React Document editor component

Text formatting

The following table lists the default keyboard shortcuts in document editor for formatting text:

Key combination	Description
----- -----	
Ctrl + B	Toggles the bold property of selected text.
Ctrl + I	Toggles the italic property of selected text.
Ctrl + U	Toggles the underline property of selected text.
Ctrl + +	Toggles the subscript formatting of selected text.
Ctrl + Shift + +	Toggles the superscript formatting of selected contents.

| Ctrl + } | Increases the actual font size of selected text by one point. |

| Ctrl + { | Decreases the actual font size of selected text by one point. |

Paragraph formatting

The following table lists the default keyboard shortcuts for formatting the paragraph:

Key combination	Description
-----------------	-------------

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

Ctrl + E	Selected paragraphs are center aligned.
----------	---

Ctrl + J	Selected paragraphs are justified.
----------	------------------------------------

Ctrl + L	Selected paragraphs are left aligned.
----------	---------------------------------------

Ctrl + R	Selected paragraphs are right aligned.
----------	--

Ctrl + 1	Single line spacing is applied for selected paragraphs.
----------	---

Ctrl + 5	1.5 line spacing is applied for selected paragraphs.
----------	--

Ctrl + 2	Double spacing is applied for selected paragraphs.
----------	--

Ctrl + 0	No spacing is applied before the selected paragraphs.
----------	---

Ctrl + M	Increases the left indent of selected paragraphs by a factor of 36 points.
----------	--

Ctrl + Shift + M	Decreases the left indent of selected paragraphs by a factor of 36 points.
------------------	--

Ctrl + *	Show/Hide the hidden characters like spaces, tab, paragraph marks, and breaks.
----------	--

Clipboard

Key Combination	Description
-----------------	-------------

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

Ctrl + C	Copies selected contents to the clipboard.
----------	--

Ctrl + V	Pastes plain text content from the clipboard.
----------	---

Ctrl + X	Moves selected content to the clipboard.
----------	--

Keyboard shortcut to navigate around the document

Key Combination	Description
-----------------	-------------

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

Left arrow	Moves the cursor position one character to the left.
------------	--

Right arrow	Moves the cursor position one character to the right.
-------------	---

Down arrow	Moves the cursor position down one line.
------------	--

Up arrow	Moves the cursor position up one line.
----------	--

Ctrl + Left arrow	Moves the cursor position one word to the left.
-------------------	---

Ctrl + Right arrow	Moves the cursor position one word to the right.
--------------------	--

Ctrl + Up arrow	Moves the cursor position one paragraph up.
-----------------	---

Ctrl + Down arrow	Moves the cursor position one paragraph down.
-------------------	---

Tab (in table)	Moves the cursor position one cell to the right.
Shift + Tab (in table)	Moves the cursor position one cell to the left.
Home	Moves the cursor position to the start of a line.
End	Moves the cursor position to the end of a line.
Page up	Moves the cursor position one screen up.
Page down	Moves the cursor position one screen down.
Ctrl + Home	Moves the cursor position to the start of a document.
Ctrl + End	Moves the cursor position to the end of a document.

Keyboard shortcut to extend selection

Key Combination	Description
----- -----	
Shift + Left arrow	Extends selection one character to the left.
Shift + Right arrow	Extends selection one character to the right.
Shift + Down arrow	Extends selection one line downward.
Shift + Up arrow	Extends selection one line upward.
Shift + Home	Extends selection to the start of a line.
Shift + End	Extends Selection to the end of a line.
Ctrl + A	Extends selection to the entire document.
Ctrl + Shift + Left arrow	Extends selection one word to the left.
Ctrl + Shift + Right arrow	Extends selection one word to the right.
Ctrl + Shift + Down arrow	Extends selection to the end of a paragraph.
Ctrl + Shift + Up arrow	Extends selection to the start of a paragraph.
Ctrl + Shift + Home	Extends selection to the start of a document.
Ctrl + Shift + End	Extends selection to the end of a document.

Find and Replace

Key Combination	Description
----- -----	
Ctrl + F	Opens options pane.
Ctrl + H	Opens replace tab in options pane.

Create, Save and Print document

Key Combination	Description
----- -----	
Ctrl + N	Opens empty document.
Ctrl + S	Saves the document in SFDT format.

|Ctrl + P| Prints the document.|

Edit Operation

Key Combination	Description
-----	-----
Backspace	Deletes one character to the left.
Delete	Deletes one character to the right.
Ctrl + Z	Undo last performed action.
Ctrl + Y	Redo last undo action.

Insert special characters

Key Combination	Description
-----	-----
Ctrl + Enter	Inserts page break.
Shift + Enter	Inserts line break.

Dialog

Key Combination	Description
-----	-----
Ctrl + F	Opens options pane.
Ctrl + D	Opens font dialog.
Ctrl + K	Opens hyperlink dialog.

See Also

- [How to override the keyboard shortcuts](#)

Scrolling zooming in React Document editor component

The Document Editor renders the document as page by page. You can scroll through the pages by mouse wheel or touch interactions. You can also scroll through the page by using 'scrollToPage()' method of document editor instance. Refer to the following code example.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-documenteditor';
function App() {
  let documenteditor;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function onLoadDefault() {
    let defaultDocument = {
      sections: [
```

```

        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'First page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'Second page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
      ],
      characterFormat: {},
      paragraphFormat: {},
      background: {
        color: '#FFFFFF',
      },
      styles: [
        {
          type: 'Paragraph',
          name: 'Normal',
          next: 'Normal',
        },
        {
          type: 'Character',
          name: 'Default Paragraph Font',
        },
      ],
    ],
  };
  documenteditor.open(JSON.stringify(defaultDocument));
  documenteditor.focusIn();
}
function componentDidMount() {
  setTimeout(() => {
    //Load default document.
    onLoadDefault();
    //Scroll to specified page.
    documenteditor.scrollToPage(2);
  });
}

```

```

    });
  }
  return (<DocumentEditorComponent id="DocumentEditor" height={'330px'}
  ref={(scope) => { documenteditor = scope; }} isReadOnly={false}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-
documenteditor';
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onLoadDefault() {
    let defaultDocument: object = {
      sections: [
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'First page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'Second page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
      ],
      characterFormat: {},
    };
  }
}

```

```

        paragraphFormat: {},
        background: {
            color: '#FFFFFF',
        },
        styles: [
            {
                type: 'Paragraph',
                name: 'Normal',
                next: 'Normal',
            },
            {
                type: 'Character',
                name: 'Default Paragraph Font',
            },
        ],
    },
    };
    documenteditor.open(JSON.stringify(defaultDocument));
    documenteditor.focusIn();
}
function componentDidMount() {
    setTimeout(() => {
        //Load default document.
        onLoadDefault();
        //Scroll to specified page.
        documenteditor.scrollToPage(2);
    });
}
return (
    <DocumentEditorComponent id="DocumentEditor" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false} />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Calling this method brings the specified page into view but doesn't move selection. Hence this method will work by default. That is, it works even if selection is not enabled.

In case, if you wish to move the selection to any page in document editor and bring it into view, you can use 'goToPage()' method of selection instance. Refer to the following code example.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, SfddtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog, } from '@syncfusion/ej2-react-
documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfddtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
function App() {
  let documenteditor;

```

```

React.useEffect(() => {
  componentDidMount();
}, []);
function onLoadDefaultDocument() {
  let defaultDocument = {
    sections: [
      {
        blocks: [
          {
            paragraphFormat: {
              styleName: 'Normal',
            },
            inlines: [
              {
                text: 'First page',
              },
            ],
          },
        ],
        headersFooters: {},
      },
      {
        blocks: [
          {
            paragraphFormat: {
              styleName: 'Normal',
            },
            inlines: [
              {
                text: 'Second page',
              },
            ],
          },
        ],
        headersFooters: {},
      },
      {
        blocks: [
          {
            paragraphFormat: {
              styleName: 'Normal',
            },
            inlines: [
              {
                text: 'Third page',
              },
            ],
          },
        ],
        headersFooters: {},
      },
    ],
    characterFormat: {},
    paragraphFormat: {},
    background: {
      color: '#FFFFFFFF',
    },
  },

```

```

        styles: [
            {
                type: 'Paragraph',
                name: 'Normal',
                next: 'Normal',
            },
            {
                type: 'Character',
                name: 'Default Paragraph Font',
            },
        ],
    };
    documenteditor.open(JSON.stringify(defaultDocument));
    documenteditor.focusIn();
}
function componentDidMount() {
    setTimeout(() => {
        //Load default document.
        onLoadDefaultDocument();
        //Navigate to specified page.
        documenteditor.viewer.scrollToPage(3);
    });
}
return (<DocumentEditorComponent id="container" height={'330px'}
ref={ (scope) => {
    documenteditor = scope;
    }} isReadOnly={false} enablePrint={true} enableSelection={true}
enableEditor={true} enableEditorHistory={true} enableContextMenu={true}
enableSearch={true} enableOptionsPane={true} enableBookmarkDialog={true}
enableBordersAndShadingDialog={true} enableFontDialog={true}
enableTableDialog={true} enableParagraphDialog={true}
enableHyperlinkDialog={true} enableImageResizer={true}
enableListDialog={true} enablePageSetupDialog={true} enableSfdtExport={true}
enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
enableTextExport={true} enableWordExport={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog } from '@syncfusion/ej2-react-
documenteditor';
//Inject require modules.

```



```

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onLoadDefaultDocument(): void {
    let defaultDocument: object = {
      sections: [
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'First page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'Second page',
                },
              ],
            },
          ],
          headersFooters: {},
        },
        {
          blocks: [
            {
              paragraphFormat: {
                styleName: 'Normal',
              },
              inlines: [
                {
                  text: 'Third page',
                },
              ],
            },
          ],
        },
      ],
    };
  }
}

```

```

        ],
        headersFooters: {},
    },
    ],
    characterFormat: {},
    paragraphFormat: {},
    background: {
        color: '#FFFFFF',
    },
    styles: [
        {
            type: 'Paragraph',
            name: 'Normal',
            next: 'Normal',
        },
        {
            type: 'Character',
            name: 'Default Paragraph Font',
        },
    ],
    ],
};
documenteditor.open(JSON.stringify(defaultDocument));
documenteditor.focusIn();
}
function componentDidMount() {
    setTimeout(() => {
        //Load default document.
        onLoadDefaultDocument();
        //Navigate to specified page.
        documenteditor.viewer.scrollToPage(3);
    });
}
return (
    <DocumentEditorComponent
        id="container"
        height={'330px'}
        ref={(scope) => {
            documenteditor = scope;
        }}
        isReadOnly={false}
        enablePrint={true}
        enableSelection={true}
        enableEditor={true}
        enableEditorHistory={true}
        enableContextMenu={true}
        enableSearch={true}
        enableOptionsPane={true}
        enableBookmarkDialog={true}
        enableBordersAndShadingDialog={true}
        enableFontDialog={true}
        enableTableDialog={true}
        enableParagraphDialog={true}
        enableHyperlinkDialog={true}
        enableImageResizer={true}
        enableListDialog={true}
        enablePageSetupDialog={true}
        enableSfddExport={true}
    />

```

```

        enableStyleDialog={true}
        enableTableOfContentsDialog={true}
        enableTableOptionsDialog={true}
        enableTablePropertiesDialog={true}
        enableTextExport={true}
        enableWordExport={true}
    />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdn.jsdelivr.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>

```

```
</body>
</html>
```

Zooming

You can scale the contents in document editor ranging from 10% to 500% of the actual size. You can achieve this using mouse or touch interactions. You can also use 'zoomFactor' property of document editor instance. The value can be specified in a range from 0.1 to 5. Refer to the following code example.

```
{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
  Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu, OptionsPane,
  HyperlinkDialog, TableDialog, BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog,
  ListDialog, ParagraphDialog, BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog,
  BordersAndShadingDialog, TableOptionsDialog, CellOptionsDialog, StylesDialog,
} from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);

function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function componentDidMount() {
    //Set zoom factor.
    documenteditor.zoomFactor = 3;
  }
  return (
    <DocumentEditorComponent id="container" ref={(scope) => { documenteditor = scope; }}
    height={'330px'} style={{
    width: '100%'
```

```

    isReadOnly={false} enablePrint={true} enableSelection={true} enableEditor={true}
    enableEditorHistory={true} enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
    enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
    enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
    enableImageResizer={true} enableListDialog={true} enablePageSetupDialog={true}
    enableSfdtExport={true} enableStyleDialog={true} enableTableOfContentsDialog={true}
    enableTableOptionsDialog={true} enableTablePropertiesDialog={true} enableTextExport={true}
    enableWordExport={true} />
  );
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

{% endraw %}
{% raw %}
`ts

import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport, Selection, Search,
Editor, ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog, } from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);

function App() {
  let documenteditor;

  React.useEffect(() => {
    componentDidMount();
  }, []);

  function componentDidMount() {
    //Set zoom factor.
    documenteditor.zoomFactor = 3;
  }
}

```

```

return (<DocumentEditorComponent id="container" ref={(scope) => { documenteditor = scope; }}
height={'330px'} style={{
width: '100%'
}} isReadOnly={false} enablePrint={true} enableSelection={true} enableEditor={true}
enableEditorHistory={true} enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
enableImageResizer={true} enableListDialog={true} enablePageSetupDialog={true}
enableSfdtExport={true} enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true} enableTablePropertiesDialog={true} enableTextExport={true}
enableWordExport={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

```
{% endraw %}
```

Page Fit Type

Apart from specifying the zoom factor as value, the Document Editor provides option to specify page fit options such as fit to full page or fit to page width. You can set this option using 'fitPage' method of document editor instance. Refer to the following code example.

```

{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
DocumentEditorComponent, DocumentEditor, Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu, OptionsPane,
HyperlinkDialog, TableDialog, BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog,
ListDialog, ParagraphDialog, BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog,
BordersAndShadingDialog, TableOptionsDialog, CellOptionsDialog, StylesDialog,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);
function App() {
let documenteditor: DocumentEditorComponent;

```

```

React.useEffect(() => {
  componentDidMount()
}, []);
function componentDidMount(): void {
  documenteditor.fitPage('FitPageWidth');
}
return (
  <DocumentEditorComponent id="container" ref={{scope} => { documenteditor = scope; }}
  height={'330px'} style={{ width: '100%' }} isReadOnly={false} enablePrint={true} enableSelection={true}
  enableEditor={true} enableEditorHistory={true} enableContextMenu={true} enableSearch={true}
  enableOptionsPane={true} enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
  enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
  enableHyperlinkDialog={true} enableImageResizer={true} enableListDialog={true}
  enablePageSetupDialog={true} enableSfdtExport={true} enableStyleDialog={true}
  enableTableOfContentsDialog={true} enableTableOptionsDialog={true}
  enableTablePropertiesDialog={true} enableTextExport={true} enableWordExport={true} />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
{% endraw %}
{% raw %}
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport, Selection, Search,
Editor, ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog, } from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);

function App() {
  let documenteditor;

```

```

React.useEffect(() => {
  componentDidMount();
}, []);
function componentDidMount() {
  documenteditor.fitPage('FitPageWidth');
}

return (<DocumentEditorComponent id="container" ref={{(scope) => { documenteditor = scope; }}
  height={'330px'} style={{ width: '100%' }} isReadOnly={false} enablePrint={true} enableSelection={true}
  enableEditor={true} enableEditorHistory={true} enableContextMenu={true} enableSearch={true}
  enableOptionsPane={true} enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
  enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
  enableHyperlinkDialog={true} enableImageResizer={true} enableListDialog={true}
  enablePageSetupDialog={true} enableSfdtExport={true} enableStyleDialog={true}
  enableTableOfContentsDialog={true} enableTableOptionsDialog={true}
  enableTablePropertiesDialog={true} enableTextExport={true} enableWordExport={true}/>);
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));

```

```
{% endraw %}
```

Zoom option using UI

The following code example shows how to provide zoom options in document editor.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent,
  Print,
  SfdtExport,
  WordExport,
  TextExport,
  Selection,
  Search,
  Editor,
  ImageResizer,
  EditorHistory,
  ContextMenu,
  OptionsPane,
  HyperlinkDialog,
  TableDialog,
  BookmarkDialog,
  TableOfContentsDialog,
  PageSetupDialog,
  StyleDialog,

```



```

ListDialog,
ParagraphDialog,
BulletsAndNumberingDialog,
FontDialog,
TablePropertiesDialog,
BordersAndShadingDialog,
TableOptionsDialog,
CellOptionsDialog,
StylesDialog,
} from '@syncfusion/ej2-react-documenteditor';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
function App() {
  let documenteditor: DocumentEditorComponent;
  let pageCount: any;
  let editablePageNumber: any;
  let editorPageCount: any;
  let pageNumberLabel: any;
  let startPage = 1;
  let zoom: any;
  let items: ItemModel[] = [
    { text: '200%' },
    { text: '175%' },
    { text: '150%' },
    { text: '125%' },
    { text: '100%' },
    { text: '75%' },
    { text: '50%' },
    { text: '25%' },
    { separator: true },
    { text: 'Fit one page' },
    { text: 'Fit page width' },
  ];
  React.useEffect(() => {
    let instance: any = this;
    instance.pageCount = document.getElementById('documenteditor_pagecount');
    instance.editablePageNumber =
document.getElementById('editablePageNumber');
    instance.pageNumberLabel =
document.getElementById('documenteditor_page_no');
    updatePageCount();
    updatePageNumber();
    documenteditor.viewChange = function (e) {
      updatePageNumberOnViewChange(e);
    };
    documenteditor.contentChange = function () {
      updatePageCount();
    };
    instance.editablePageNumber.onclick = function () {
      updateDocumentEditorPageNumber();
    };
    instance.editablePageNumber.onkeydown = function (e) {
      onKeyDown(e);
    };
    instance.editablePageNumber.onblur = function () {
      onBlur();
    };
  });
};

```

```

}, []);
function updatePageNumberOnViewChange(args: any) {
  if (
    documenteditor.selection &&
    documenteditor.selection.startPage >= args.startPage &&
    documenteditor.selection.startPage <= args.endPage
  ) {
    startPage = documenteditor.selection.startPage;
  } else {
    startPage = args.startPage;
  }
  updatePageNumber();
}
function onBlur() {
  if (
    editablePageNumber.textContent === '' ||
    parseInt(editablePageNumber.textContent, 0) > editorPageCount
  ) {
    updatePageNumber();
  }
  editablePageNumber.contentEditable = 'false';
}
function onKeyDown(e: any) {
  if (e.which === 13) {
    e.preventDefault();
    var pageNumber = parseInt(editablePageNumber.textContent, 0);
    if (pageNumber > editorPageCount) {
      updatePageNumber();
    } else {
      if (documenteditor.selection) {
documenteditor.selection.goToPage(parseInt(editablePageNumber.textContent,
0));
      } else {
documenteditor.scrollToPage(parseInt(editablePageNumber.textContent, 0));
      }
    }
    editablePageNumber.contentEditable = 'false';
    if (editablePageNumber.textContent === '') {
      updatePageNumber();
    }
  }
  if (e.which > 64) {
    e.preventDefault();
  }
}
function onZoom(args: any) {
  setZoomValue(args.item.text);
  updateZoomContent();
}
function setZoomValue(text: string) {
  if (text.match('Fit one page')) {
    documenteditor.fitPage('FitOnePage');
  } else if (text.match('Fit page width')) {
    documenteditor.fitPage('FitPageWidth');
  } else {

```

```

        documenteditor.zoomFactor = parseInt(text, 0) / 100;
    }
}
function updateZoomContent() {
    zoom.content = Math.round(documenteditor.zoomFactor * 100) + '%';
}
function updatePageNumber() {
    pageNumberLabel.textContent = startPage.toString();
}
function updatePageCount() {
    editorPageCount = documenteditor.pageCount;
    pageCount.textContent = editorPageCount.toString();
}
function updateDocumentEditorPageNumber() {
    let editPageNumber = document.getElementById('editablePageNumber');
    editPageNumber.contentEditable = 'true';
    editPageNumber.focus();
    window.getSelection().selectAllChildren(editPageNumber);
}
return (
    <div>
        <DocumentEditorComponent
            id="container"
            height={'330px'}
            ref={(scope) => {
                documenteditor = scope;
            }}
            isReadOnly={false}
            enablePrint={true}
            enableSelection={true}
            enableEditor={true}
            enableEditorHistory={true}
            enableContextMenu={true}
            enableSearch={true}
            enableOptionsPane={true}
            enableBookmarkDialog={true}
            enableBordersAndShadingDialog={true}
            enableFontDialog={true}
            enableTableDialog={true}
            enableParagraphDialog={true}
            enableHyperlinkDialog={true}
            enableImageResizer={true}
            enableListDialog={true}
            enablePageSetupDialog={true}
            enableSfdtExport={true}
            enableStyleDialog={true}
            enableTableOfContentsDialog={true}
            enableTableOptionsDialog={true}
            enableTablePropertiesDialog={true}
            enableTextExport={true}
            enableWordExport={true}
        />
        <div id="page-fit-type-div">
            <label id="page"> Page </label>
            <div id="editablePageNumber">
                <label id="documenteditor_page_no" />
            </div>
        </div>
    </div>
)

```

```

    <label id="of">of</label>
    <label id="documenteditor_pagecount" />
    <DropDownButtonComponent
      ref={(scope) => {
        zoom = scope;
      }}
      items={items}
      cssClass="e-de-statusbar-zoom"
      select={onZoom}
    >
      100%
    </DropDownButtonComponent>
  </div>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent,
  Print,
  SfdtExport,
  WordExport,
  TextExport,
  Selection,
  Search,
  Editor,
  ImageResizer,
  EditorHistory,
  ContextMenu,
  OptionsPane,
  HyperlinkDialog,
  TableDialog,
  BookmarkDialog,
  TableOfContentsDialog,
  PageSetupDialog,
  StyleDialog,
  ListDialog,
  ParagraphDialog,
  BulletsAndNumberingDialog,
  FontDialog,
  TablePropertiesDialog,
  BordersAndShadingDialog,
  TableOptionsDialog,
  CellOptionsDialog,
  StylesDialog,
} from '@syncfusion/ej2-react-documenteditor';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';

```

```

function App() {
  let documenteditor: DocumentEditorComponent;
  let pageCount: any;
  let editablePageNumber: any;
  let editorPageCount: any;
  let pageNumberLabel: any;
  let startPage = 1;
  let zoom: any;
  let items: ItemModel[] = [
    { text: '200%' },
    { text: '175%' },
    { text: '150%' },
    { text: '125%' },
    { text: '100%' },
    { text: '75%' },
    { text: '50%' },
    { text: '25%' },
    { separator: true },
    { text: 'Fit one page' },
    { text: 'Fit page width' },
  ];
  React.useEffect(() => {
    let instance: any = this;
    instance.pageCount = document.getElementById('documenteditor_pagecount');
    instance.editablePageNumber =
    document.getElementById('editablePageNumber');
    instance.pageNumberLabel =
    document.getElementById('documenteditor_page_no');
    updatePageCount();
    updatePageNumber();
    documenteditor.viewChange = function (e) {
      updatePageNumberOnViewChange(e);
    };
    documenteditor.contentChange = function () {
      updatePageCount();
    };
    instance.editablePageNumber.onclick = function () {
      updateDocumentEditorPageNumber();
    };
    instance.editablePageNumber.onkeydown = function (e) {
      onKeyDown(e);
    };
    instance.editablePageNumber.onblur = function () {
      onBlur();
    };
  }, []);
  function updatePageNumberOnViewChange(args: any) {
    if (
      documenteditor.selection &&
      documenteditor.selection.startPage >= args.startPage &&
      documenteditor.selection.startPage <= args.endPage
    ) {
      startPage = documenteditor.selection.startPage;
    } else {
      startPage = args.startPage;
    }
    updatePageNumber();
  }
}

```

```

}
function onBlur() {
  if (
    editablePageNumber.textContent === '' ||
    parseInt(editablePageNumber.textContent, 0) > editorPageCount
  ) {
    updatePageNumber();
  }
  editablePageNumber.contentEditable = 'false';
}
function onKeyDown(e: any) {
  if (e.which === 13) {
    e.preventDefault();
    var pageNumber = parseInt(editablePageNumber.textContent, 0);
    if (pageNumber > editorPageCount) {
      updatePageNumber();
    } else {
      if (documenteditor.selection) {
documenteditor.selection.goToPage(parseInt(editablePageNumber.textContent,
0));
      } else {
documenteditor.scrollToPage(parseInt(editablePageNumber.textContent, 0));
      }
    }
    editablePageNumber.contentEditable = 'false';
    if (editablePageNumber.textContent === '') {
      updatePageNumber();
    }
  }
  if (e.which > 64) {
    e.preventDefault();
  }
}
function onZoom(args: any) {
  setZoomValue(args.item.text);
  updateZoomContent();
}
function setZoomValue(text: string) {
  if (text.match('Fit one page')) {
    documenteditor.fitPage('FitOnePage');
  } else if (text.match('Fit page width')) {
    documenteditor.fitPage('FitPageWidth');
  } else {
    documenteditor.zoomFactor = parseInt(text, 0) / 100;
  }
}
function updateZoomContent() {
  zoom.content = Math.round(documenteditor.zoomFactor * 100) + '%';
}
function updatePageNumber() {
  pageNumberLabel.textContent = startPage.toString();
}
function updatePageCount() {
  editorPageCount = documenteditor.pageCount;
  pageCount.textContent = editorPageCount.toString();
}

```

```

}
function updateDocumentEditorPageNumber() {
  let editPageNumber = document.getElementById('editablePageNumber');
  editPageNumber.contentEditable = 'true';
  editPageNumber.focus();
  window.getSelection().selectAllChildren(editPageNumber);
}
return (
  <div>
    <DocumentEditorComponent
      id="container"
      height={ '330px' }
      ref={ (scope) => {
        documenteditor = scope;
      }}
      isReadOnly={false}
      enablePrint={true}
      enableSelection={true}
      enableEditor={true}
      enableEditorHistory={true}
      enableContextMenu={true}
      enableSearch={true}
      enableOptionsPane={true}
      enableBookmarkDialog={true}
      enableBordersAndShadingDialog={true}
      enableFontDialog={true}
      enableTableDialog={true}
      enableParagraphDialog={true}
      enableHyperlinkDialog={true}
      enableImageResizer={true}
      enableListDialog={true}
      enablePageSetupDialog={true}
      enableSfdtExport={true}
      enableStyleDialog={true}
      enableTableOfContentsDialog={true}
      enableTableOptionsDialog={true}
      enableTablePropertiesDialog={true}
      enableTextExport={true}
      enableWordExport={true}
    />
    <div id="page-fit-type-div">
      <label id="page"> Page </label>
      <div id="editablePageNumber">
        <label id="documenteditor_page_no" />
      </div>
      <label id="of">of</label>
      <label id="documenteditor_pagecount" />
      <DropDownButtonComponent
        ref={ (scope) => {
          zoom = scope;
        }}
        items={items}
        cssClass="e-de-statusbar-zoom"
        select={onZoom}
      >
        100%
      </DropDownButtonComponent>
    </div>
  </div>
)

```

```

    </div>
  </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```


Print in React Document editor component

To print the document, use the [print](#) method from document editor instance.

Refer to the following example for showing a document and print it.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print);
function App() {
  let documenteditor;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function onPrint() {
    //Print the document content.
    documenteditor.print();
  }
  function componentDidMount() {
    let sfdt = `{
      "sections": [
        {
          "blocks": [
            {
              "inlines": [
                {
                  "characterFormat": {
                    "bold": true,
                    "italic": true
                  },
                  "text": "Hello World"
                }
              ]
            }
          ],
          "headersFooters": {
            }
          }
        ]
      }`;
    setTimeout(() => {
      //Open the document in Document Editor.
      documenteditor.open(sfdt);
    });
  }
  return (<div>
    <button onClick={onPrint}>Print</button>
    <DocumentEditorComponent id="container" ref={scope => {
      documenteditor = scope;
    }} enablePrint={true} height={'330px'}/>
  </div>);
}
```

```
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onPrint() {
    //Print the document content.
    documenteditor.print();
  }
  function componentDidMount() {
    let sfdt: string = `{
      "sections": [
        {
          "blocks": [
            {
              "inlines": [
                {
                  "characterFormat": {
                    "bold": true,
                    "italic": true
                  },
                  "text": "Hello World"
                }
              ]
            }
          ],
          "headersFooters": {
            }
          }
        ]
      }`;
    setTimeout(() => {
      //Open the document in Document Editor.
      documenteditor.open(sfdt);
    });
  }
  return (
    <div>
      <button onClick={onPrint}>Print</button>
      <DocumentEditorComponent
        id="container"
        ref={scope => {
          documenteditor = scope;
        }}
      />
    </div>
  );
}
```

```

    }}
    enablePrint={true}
    height={'330px'}
  />
</div>
);
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdn.jsdelivr.net/npm/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Refer to the following example for creating a document and print it.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, Editor, Selection, EditorHistory,
SfdtExport
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Print, Editor, Selection, SfdtExport,
EditorHistory);
let documenteditor: DocumentEditorComponent;
function App() {
  return (
    <div>
      <button onClick={onPrint}>Print</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={ (scope)
=> { documenteditor = scope; }} enablePrint={true} isReadOnly={false}
enableSelection={true} enableSfdtExport={true} enableEditor={true}
enableEditorHistory={true}
      />
    </div>
  );
  function onPrint() {
    //Print the document content.
    documenteditor.print();
  }
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, Editor, Selection, EditorHistory,
SfdtExport
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Print, Editor, Selection, SfdtExport,
EditorHistory);
let documenteditor: DocumentEditorComponent;
function App() {
  return (
    <div>
      <button onClick={onPrint}>Print</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={ (scope)
=> { documenteditor = scope; }} enablePrint={true} isReadOnly={false}
enableSelection={true} enableSfdtExport={true} enableEditor={true}
enableEditorHistory={true}
      />
    </div>
  );
};
```

```
function onPrint() {
  //Print the document content.
  documenteditor.print();
}
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

DocumentEditor features are segregated into individual feature-wise modules. To use print inject `Print` module using the `DocumentEditor.Inject(Print)`.

To enable print for a document editor instance, set `enablePrint` as `true`.

Improve print quality

Document editor provides an option to improve the print quality using `printDevicePixelRatio` in Document editor settings. Document editor using canvas approach to render content. Then, canvas are converted to image and it process for print. Using `printDevicePixelRatio` API, you can increase the image quality based on your requirement.

The following example code illustrates how to improve the print quality in Document editor container.

```
`ts
import * as ReactDOM from 'react-dom';

import {
  DocumentEditorContainerComponent, Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);

function App() {
  let settings={printDevicePixelRatio :2};
  return (
    <DocumentEditorContainerComponent id="container" height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
    enableToolbar={true} documentEditorSettings= {settings}/>
  )
}

export default App

ReactDOM.render(<App />, document.getElementById('root'));
```

Note: By default, `printDevicePixelRatio` value is 1

Print using window object

You can print the document in document editor by passing the window instance. This is useful to implement print in third party frameworks such as electron, where the window instance will not be available. Refer to the following example.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent,
  DocumentEditor,
```

```

Print,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditor.Inject(Print);
function App() {
let documenteditor: DocumentEditorComponent;
React.useEffect(() => {
componentDidMount()
}, []);
function componentDidMount() {
//Print the document content.
documenteditor.print(window);
}
return (
<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
enablePrint={true}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, DocumentEditor, Print, } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditor.Inject(Print);
function App() {

```

```

let documenteditor;
React.useEffect(() => {
  componentDidMount();
}, []);
function componentDidMount() {
  //Print the document content.
  documenteditor.print(window);
}
return (<DocumentEditorComponent id="container" height={'330px'} ref={scope => {
  documenteditor = scope;
}} enablePrint={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Page setup

Some of the print options cannot be configured using JavaScript. Refer to the following links to learn more about the browser page setup:

- [Chrome](#)
- [Firefox](#)

However, you can customize margins, paper, and layout options by modifying the section format properties using page setup dialog

`ts

```

import * as ReactDOM from 'react-dom';

import { DocumentEditorComponent, DocumentEditor, Print, Editor, Selection, EditorHistory,
PageSetupDialog, SfdtExport } from '@syncfusion/ej2-react-documenteditor';

//Inject require modules.
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory, SfdtExport, PageSetupDialog);

function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  React.useEffect(() => {
    componentDidMount()
  }, []);
  return (

```



```

<DocumentEditorComponent
id="container"
height={'330px'}
ref={scope => {
documenteditor = scope;
}}
enablePrint={true}
isReadOnly={false}
enableSelection={true}
enableSfdtExport={true}
enableEditor={true}
enableEditorHistory={true}
enablePageSetupDialog={true}
/>
);
function ComponentDidMount() {
//Open page setup dialog.
documenteditor.showPageSetupDialog();
}
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`ts
import * as ReactDOM from 'react-dom';
import { DocumentEditorComponent, DocumentEditor, Print, Editor, Selection, EditorHistory,
PageSetupDialog, SfdtExport } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory, SfdtExport, PageSetupDialog);
function App() {
let documenteditor = new DocumentEditorComponent(undefined);
React.useEffect(() => {
componentDidMount();

```

```

}, []);

return (<DocumentEditorComponent id="container" height={'330px'} ref={scope => {
  documenteditor = scope;
}} enablePrint={true} isReadOnly={false} enableSelection={true} enableSfdtExport={true}
enableEditor={true} enableEditorHistory={true} enablePageSetupDialog={true}/>);

function ComponentDidMount() {
  //Open page setup dialog.
  documenteditor.showPageSetupDialog();
}
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

By customizing margins, papers, and layouts, the layout of the document will be changed in document editor. To modify these options during print operation, serialize the document as SFDT using the [serialize](#) method in document editor instance and open the SFDT data in another instance of document editor in separate window.

The following example shows how to customize layout options only for printing.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, DocumentEditor, Print, Editor, Selection,
EditorHistory, SfdtExport, } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory, SfdtExport);
function App() {
  let documenteditor1 = new DocumentEditorComponent(undefined);
  let documenteditor2 = new DocumentEditorComponent(undefined);
  function onPrint() {
    //Serialize the document content.
    let sfdtData = documenteditor1.serialize();
    //Open the serialized content in Document Editor.
    documenteditor2.open(sfdtData);
    //Set A5 paper size
    documenteditor2.selection.sectionFormat.pageWidth = 419.55;
    documenteditor2.selection.sectionFormat.pageHeight = 595.3;
    //Print the document content.
    documenteditor2.print();
  }
  return (<div>
    <button onClick={onPrint}>Print</button>
    <DocumentEditorComponent id="DocumentEditor" height={'330px'}
    ref={scope => {
      documenteditor1 = scope;

```

```

    }} enablePrint={true} isReadOnly={false} enableSelection={true}
enableSfdtExport={true} enableEditor={true} enableEditorHistory={true}/>
    <DocumentEditorComponent id="DocumentEditor2" height={'330px'}
ref={scope => {
        documenteditor2 = scope;
    }} enablePrint={true} isReadOnly={false} enableSelection={true}
enableSfdtExport={true} enableEditor={true} enableEditorHistory={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, DocumentEditor, Print, Editor, Selection,
EditorHistory, SfdtExport, } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditor.Inject(Print, Editor, Selection, EditorHistory, SfdtExport);
function App() {
    let documenteditor1: DocumentEditorComponent = new
DocumentEditorComponent(undefined);
    let documenteditor2: DocumentEditorComponent = new
DocumentEditorComponent(undefined);
    function onPrint() {
        //Serialize the document content.
        let sfdtData = documenteditor1.serialize();
        //Open the serialized content in Document Editor.
        documenteditor2.open(sfdtData);
        //Set A5 paper size
        documenteditor2.selection.sectionFormat.pageWidth = 419.55;
        documenteditor2.selection.sectionFormat.pageHeight = 595.3;
        //Print the document content.
        documenteditor2.print();
    }
    return (
        <div>
            <button onClick={onPrint}>Print</button>
            <DocumentEditorComponent
                id="DocumentEditor"
                height={'330px'}
                ref={scope => {
                    documenteditor1 = scope;
                }}
                enablePrint={true}
                isReadOnly={false}
                enableSelection={true}
                enableSfdtExport={true}
                enableEditor={true}
                enableEditorHistory={true}
            />
            <DocumentEditorComponent
                id="DocumentEditor2"

```

```

        height={'330px'}
        ref={scope => {
            documenteditor2 = scope;
        }}
        enablePrint={true}
        isReadOnly={false}
        enableSelection={true}
        enableSfdtExport={true}
        enableEditor={true}
        enableEditorHistory={true}
    />
</div>
);
} export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>

```

```
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

See Also

- [Feature modules](#)
- [Page Setup dialog](#)

Dialog in React Document editor component

Document Editor provides dialog support to major operations such as insert or edit hyperlink, formatting text, paragraph, style, list and table properties.

Font Dialog

Font dialog allows you to modify all text properties for selected contents at once such as bold, italic, underline, font size, font color, strikethrough, subscript and superscript.

Document Editor features are segregated into individual feature-wise modules. To use font Dialog, inject 'FontDialog' module using the 'DocumentEditor.Inject(Selection, SfdtExport, Editor, FontDialog)'.

To enable font dialog for a document editor instance, set 'enableFontDialog' to true.

Refer to the following example.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfdtExport, Selection, Editor, FontDialog }
from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, FontDialog);
let documenteditor;
function App() {
  return (
    <div>
      <button onClick={showFontDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={(scope)
=> { documenteditor = scope; }} isReadOnly={false} enableSelection={true}
enableEditor={true} enableSfdtExport={true} enableFontDialog={true}/>
    </div>
  );
  function showFontDialog() {
    //Open font dialog.
    documenteditor.showDialog('Font');
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, SfddExport, Selection, Editor, FontDialog }
from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfddExport, Selection, Editor, FontDialog);
let documenteditor: DocumentEditorComponent;
function App() {
  return (
    <div>
      <button onClick={showFontDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={(scope)
=> { documenteditor = scope; }} isReadOnly={false} enableSelection={true}
enableEditor={true} enableSfddExport={true} enableFontDialog={true} />
    </div>
  );
  function showFontDialog() {
    //Open font dialog.
    documenteditor.showDialog('Font');
  }
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />

```

```

    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Paragraph dialog

This dialog allows modifying the paragraph formatting for selection at once such as text alignment, indentation, and spacing.

To open this dialog, refer to the following example.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import { DocumentEditorComponent, SfdtExport, Selection, Editor, ParagraphDialog } from
 '@syncfusion/ej2-react-documenteditor';
```

```
//Inject require modules.
```

```
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, ParagraphDialog);
```

```
function App() {
```

```
let documenteditor: DocumentEditorComponent;
```

```
return (
```

```
<div>
```

```
<button onClick={showParagraphDialog}>Dialog</button>
```

```
<DocumentEditorComponent id="container" height={'330px'} ref={(scope) => {documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableParagraphDialog={true} />
```

```
</div>
```

```
);
```

```
function showParagraphDialog(){
```

```
//Open paragraph dialog.
```

```
documenteditor.showDialog('Paragraph');
```

```
}
```

```

}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Table dialog

This dialog allows creating and inserting a table at cursor position by specifying the required number of rows and columns.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, TableDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, TableDialog);
function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  function ShowTableDialog() {
    //Open table dialog.
    documenteditor.showDialog('Table');
  }
  return (
    <div>
      <button onClick={ShowTableDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={(scope) => { documenteditor = scope;
      }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
      enableTableDialog={true} />
    </div>
  );
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Bookmark dialog

This dialog allows you to perform the following operations:

- View all bookmarks.
- Navigate to a bookmark.
- Create a bookmark at current selection.
- Delete an existing bookmark.

To open this dialog, refer to the following example.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, BookmarkDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, BookmarkDialog);
function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  function ShowBookmarkDialog() {
    //Open bookmark dialog.
    documenteditor.showDialog('Bookmark');
  }
  return (
    <div>
      <button onClick={ShowBookmarkDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
        }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
        enableBookmarkDialog={true} />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Hyperlink dialog

This dialog allows editing or inserting a hyperlink at cursor position.

To open this dialog, refer to the following example.

```
`ts
import * as ReactDOM from 'react-dom';
```

```

import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, HyperlinkDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, HyperlinkDialog);
function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  function ShowHyperlinkDialog() {
    //Open hyperlink dialog;
    documenteditor.showDialog('Hyperlink');
  }
  return (
    <div>
      <button onClick={ShowHyperlinkDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
      }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
      enableHyperlinkDialog={true} />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Table of contents dialog](#)

This dialog allows creating and inserting table of contents at cursor position. If the table of contents already exists at cursor position, you can customize its properties.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, TableOfContentsDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, TableOfContentsDialog);

```

```
function App() {
  let documenteditor: DocumentEditorComponent = new DocumentEditorComponent(undefined);
  function ShowTableOfContentsDialog() {
    //Open table of contents dialog.
    documenteditor.showDialog('TableOfContents');
  }
  return (
    <div>
      <button onClick={ShowTableOfContentsDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor = scope;
        }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
        enableTableOfContentsDialog={true} />
    </div>
  );
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Styles Dialog

This dialog allows managing the styles in a document. It will display all the styles in the document with options to modify the properties of the existing style or create new style with the help of 'Style dialog'. Refer to the following example.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, StyleDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, StyleDialog, StylesDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  function ShowStylesDialog() {
    //Open styles dialog.
    documenteditor.showDialog('Styles');
  }
}
```

```

}
return (
<div>
<button onClick={ShowStylesDialog}>Dialog</button>

<DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableStyleDialog={true} />
</div>
);
}

export default App

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Style dialog

You can directly use this dialog for modifying any existing style or add new style by providing the style name.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, StyleDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, StyleDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  function ShowStyleDialog() {
    //Open style dialog.
    documenteditor.showDialog('Style');
  }
  return (
<div>
<button onClick={ShowStyleDialog}>Dialog</button>

```

```

<DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableStyleDialog={true} />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

List dialog

This dialog allows creating a new list or modifying existing lists in the document.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, ListDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, ListDialog);
export class Default extends React.Component<{}, {}> {
  public documenteditor: DocumentEditorComponent;
  showListDialog(): void {
    //Open list dialog.
    this.documenteditor.showDialog('List');
  }
  render() {
    return (
      <div>
        <button onClick={this.showListDialog.bind(this)}>Dialog</button>
        <DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { this.documenteditor =
scope; }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableListDialog={true} />
      </div>
    );
  }
}

```

```

}
}
ReactDOM.render(<Default />, document.getElementById('sample'));
`

```

Borders and shading dialog

This dialog allows customizing the border style, border width, and background color of the table or selected cells.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, BordersAndShadingDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, BordersAndShadingDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    ComponentDidMount();
  }, []);
  function ComponentDidMount() {
    //Insert table
    documenteditor.editor.insertTable(2, 2);
  }
  function ShowBordersAndShadingDialog() {
    //Open borders and shading dialog.
    documenteditor.showDialog('BordersAndShading');
  }
  return (
    <div>
      <button onClick={ShowBordersAndShadingDialog}>Dialog</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor = scope;
        }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
        enableBordersAndShadingDialog={true} />
    </div>
  );
}

```

```

</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Table options dialog

This dialog allows customizing the default cell margins and spacing between each cells of the selected table.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, TableOptionsDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, TableOptionsDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    ComponentDidMount();
  }, []);
  function ComponentDidMount() {
    //Insert table.
    documenteditor.editor.insertTable(2, 2);
  }
  function ShowTableOptionsDialog() {
    //Open table options dialog.
    documenteditor.showDialog('TableOptions');
  }
  return (
    <div>
    <button onClick={ShowTableOptionsDialog}>Dialog</button>

```

```

<DocumentEditorComponent id="container" height={'330px'} ref={(scope) => { documenteditor = scope;
}} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
enableTableOptionsDialog={true} />
</div>
);
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Table properties dialog

This dialog allows customizing the table, row, and cell properties of the selected table.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, TableOptionsDialog, TablePropertiesDialog,
  BordersAndShadingDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, BordersAndShadingDialog,
TableOptionsDialog, TablePropertiesDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  React.useEffect(() => {
    ComponentDidMount();
  }, []);
  function ComponentDidMount() {
    //Insert table.
    documenteditor.editor.insertTable(2, 2);
  }
  function ShowTablePropertiesDialog() {
    //Open table properties dialog.
    documenteditor.showDialog('TableProperties');
  }
  return (

```



```

<div>
  <button onClick={ShowTablePropertiesDialog}>Dialog</button>

  <DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
  }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
  enableBordersAndShadingDialog={true} enableTableOptionsDialog={true}
  enableTablePropertiesDialog={true} />
</div>
);
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Page setup dialog

This dialog allows customizing margins, size, and layout options for pages of the section.

To open this dialog, refer to the following example.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, SfdtExport, Selection, Editor, PageSetupDialog
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SfdtExport, Selection, Editor, PageSetupDialog);
function App() {
  let documenteditor: DocumentEditorComponent;
  function ShowPageSetupDialog() {
    //Open page setup dialog.
    documenteditor.showDialog('PageSetup');
  }
  return (
    <div>
      <button onClick={ShowPageSetupDialog}>Dialog</button>

      <DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor = scope;
      }} isReadOnly={false} enableSelection={true} enableEditor={true} enableSfdtExport={true}
      enablePageSetupDialog={true} />
    </div>

```

```
);
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`
```

See Also

- [Feature modules](#)

Right to left in React Document editor component

Document Editor provides RTL (right-to-left) support. This can be enabled using the “enableRtl” property.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, SfddtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog } from '@syncfusion/ej2-react-
documenteditor';
import { L10n } from '@syncfusion/ej2-base';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfddtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
//Load locale constant for Document Editor.
L10n.load({
  'ar-AE': {
    'documenteditor': {
      'Table': 'الجدول',
      'Row': 'الصف',
      'Cell': 'الخلية',
      'Ok': 'موافق',
      'Cancel': 'إلغاء الأمر',
      'Size': 'حجم',
      'Preferred Width': 'العرض المفضل',
      'Points': 'نقاط',
      'Percent': 'المائه',
      'Measure in': 'القياس في',
      'Alignment': 'محاذاة',
      'Left': 'ليسار',
      'Center': 'مركز',
    }
  }
});
```

```

    'Right': 'الحق',
    'Justify': 'تبرير',
    'Indent from left': 'مسافة بادئه من اليسار',
    'Borders and Shading': 'الحدود والتظليل',
    'Options': 'خيارات',
    'Specify height': 'تحديد الارتفاع',
    'At least': 'الاقل',
    'Exactly': 'تماما',
    'Row height is': 'ارتفاع الصف هو',
    'Allow row to break across pages': 'السماح بفصل الصف عبر',
    'الصفحات',
    'Repeat as header row at the top of each page': 'تكرار كصف راس',
    'في اعلي كل صفحه',
    'Vertical alignment': 'محاذاة عمودي',
    'Top': 'أعلى',
    'Bottom': 'اسفل',
    'Default cell margins': 'هوامش الخلية الافتراضية',
    'Default cell spacing': 'تباعد الخلايا الافتراضي',
    'Allow spacing between cells': 'السماح بالتباعد بين الخلايا',
    'Cell margins': 'هوامش الخلية',
    'Same as the whole table': 'نفس الجدول بأكمله',
    'Borders': 'الحدود',
    'None': 'اي',
    'Single': 'واحد',
    'Dot': 'نقطه',
    'DashSmallGap': 'داشسمجاب',
    'DashLargeGap': 'الاتحاد الخاص',
    'DashDot': 'داشدوت',
    'DashDotDot': 'ددهدودوت',
    'Double': 'انقر نقرا مزدوجا',
    'Triple': 'الثلاثي',
    'ThinThickSmallGap': 'فجوه صغيره سميكه رقيق',
    'ThickThinSmallGap': 'الفجوة الصغيره رقيقه سميكه',
    'ThinThickThinSmallGap': 'صغيره سميكه رقيقه الفجوة الصغيره',
    'ThinThickMediumGap': 'فجوه متوسطه سميك',
    'ThickThinMediumGap': 'سميكه الفجوة متوسطه رقيقه',
    'ThinThickThinMediumGap': 'رقيقه سميكه متوسطه الفجوة',
    'ThinThickLargeGap': 'الفجوة الكبيره رقيقه سميكه',
    'ThickThinLargeGap': 'فجوه كبيره رقيقه سميك',
    'ThinThickThinLargeGap': 'رقيقه سميكه الفجوة الكبيره',
    'SingleWavy': 'واحد مائج',
    'DoubleWavy': 'مزدوج مائج',
    'DashDotStroked': 'اندفاعه نقطه القوية',
    'Emboss3D': 'مزخرفD3',
    'Engrave3D': 'نقشD3',
    'Outset': 'البدايه',
    'Inset': 'الداخلي',
    'Thick': 'سميكه',
    'Style': 'نمط',
    'Width': 'عرض',
    'Height': 'ارتفاع',
    'Letter': 'رساله',
    'Tabloid': 'التابلويد',
    'Legal': 'القانونيه',
    'Statement': 'بيان',
    'Executive': 'التنفيذي',
    'A3': 'A3',

```

```

'A4': 'A4',
'A5': 'A5',
'B4': 'B4',
'B5': 'B5',
'Custom Size': 'حجم مخصص',
'Different odd and even': 'مختلفه غريبه وحتى',
'Different first page': 'الصفحة الاولى مختلفه',
'From edge': 'من الحافة',
'Header': 'رأس',
'Footer': 'تذييل الصفحة',
'Margin': 'الهوامش',
'Paper': 'الورق',
'Layout': 'تخطيط',
'Orientation': 'التوجه',
'Landscape': 'المناظر الطبيعيه',
'Portrait': 'صوره',
'Table Of Contents': 'جدول المحتويات',
'Show page numbers': 'إظهار أرقام الصفحات',
'Right align page numbers': 'محاذاة أرقام الصفحات إلى اليمين',
'Nothing': 'شيء',
'Tab leader': 'قائد علامة التبويب',
'Show levels': 'إظهار المستويات',
'Use hyperlinks instead of page numbers': 'استخدام الارتباطات',
'التشعبية بدلا من أرقام الصفحات',
'Build table of contents from': 'بناء جدول محتويات من',
'Styles': 'انماط',
'Available styles': 'الأنماط المتوفرة',
'TOC level': 'مستوي جدول المحتويات',
'Heading': 'عنوان',
'Heading 1': 'عنوان 1',
'Heading 2': 'عنوان 2',
'Heading 3': 'عنوان 3',
'Heading 4': 'عنوان 4',
'Heading 5': 'عنوان 5',
'Heading 6': 'عنوان 6',
'List Paragraph': 'فقرة القائمة',
'Normal': 'العادي',
'Outline levels': 'مستويات المخطط التفصيلي',
'Table entry fields': 'حقول إدخال الجدول',
'Modify': 'تعديل',
'Color': 'لون',
'Setting': 'اعداد',
'Box': 'مربع',
'All': 'جميع',
'Custom': 'المخصصه',
'Preview': 'معاينه',
'Shading': 'التظليل',
'Fill': 'ملء',
'Apply To': 'تنطبق علي',
'Table Properties': 'خصائص الجدول',
'Cell Options': 'خيارات الخلية',
'Table Options': 'خيارات الجدول',
'Insert Table': 'ادراج جدول',
'Number of columns': 'عدد الاعمده',
'Number of rows': 'عدد الصفوف',
'Text to display': 'النص الذي سيتم عرضه',
'Address': 'عنوان',

```

```

'Insert Hyperlink': 'إدراج ارتباط تشعبي',
'Edit Hyperlink': 'تحرير ارتباط تشعبي',
'Insert': 'إدراج',
'General': 'العامه',
'Indentation': 'المسافه البادئه',
'Before text': 'قبل النص',
'Special': 'الخاصه',
'First line': 'السطر الأول',
'Hanging': 'معلقه',
'After text': 'بعد النص',
'By': 'من',
'Before': 'قبل',
'Line Spacing': 'تباعد الأسطر',
'After': 'بعد',
'At': 'في',
'Multiple': 'متعدده',
'Spacing': 'تباعد',
'Define new Multilevel list': 'تحديد قائمه متعددة الاصعده جديده',
'List level': 'مستوي القائمة',
'Choose level to modify': 'اختر المستوي الذي تريد تعديله',
'Level': 'مستوي',
'Number format': 'تنسيق الأرقام',
'Number style for this level': 'نمط الرقم لهذا المستوي',
'Enter formatting for number': 'إدخال تنسيق لرقم',
'Start at': 'بداية من',
'Restart list after': 'أعاده تشغيل قائمه بعد',
'Position': 'موقف',
'Text indent at': 'المسافة البادئة للنص في',
'Aligned at': 'محاذاة في',
'Follow number with': 'اتبع الرقم مع',
'Tab character': 'حرف علامة التبويب',
'Space': 'الفضاء',
'Arabic': 'العربية',
'UpRoman': 'حتى الروماني',
'LowRoman': 'الرومانية منخفضه',
'UpLetter': '',
'LowLetter': '',
'Number': 'عدد',
'Leading zero': 'يؤدي صفر',
'Bullet': 'رماسه',
'Ordinal': 'الترتيبيه',
'Ordinal Text': 'النص الترتيبي',
'For East': 'للشرق',
'No Restart': 'لا أعاده تشغيل',
'Font': 'الخط',
'Font style': 'نمط الخط',
'Underline style': 'نمط التسطير',
'Font color': 'لون الخط',
'Effects': 'الاثار',
'Strikethrough': 'يتوسطه',
'Superscript': 'مرتفع',
'Subscript': 'منخفض',
'Double strikethrough': 'خط مزدوج يتوسطه خط',
'Regular': 'العادي',
'Bold': 'جريئه',
'Italic': 'مائل',
'Cut': 'قطع',

```

```

'Copy': 'نسخ',
'Paste': 'لصق',
'Hyperlink': 'الارتباط التشعبي',
'Open Hyperlink': 'فتح ارتباط تشعبي',
'Copy Hyperlink': 'نسخ ارتباط تشعبي',
'Remove Hyperlink': 'أزاله ارتباط تشعبي',
'Paragraph': 'الفقره',
'Linked(Paragraph and Character)': 'مرتبط (فقره وحرف)',
'Character': 'حرف',
'Merge Cells': 'دمج الخلايا',
'Insert Above': 'ادراج أعلاه',
'Insert Below': 'ادراج أدناه',
'Insert Left': 'ادراج إلى اليسار',
'Insert Right': 'ادراج اليمين',
>Delete': 'حذف',
>Delete Table': 'حذف جدول',
>Delete Row': 'حذف صف',
>Delete Column': 'حذف عمود',
'File Name': 'اسم الملف',
'Format Type': 'نوع التنسيق',
'Save': 'حفظ',
'Navigation': 'التنقل',
'Results': 'نتائج',
'Replace': 'استبدال',
'Replace All': 'استبدال الكل',
'We replaced all': 'استبدلنا جميع',
'Find': 'العثور',
'No matches': 'لا توجد تطابقات',
'All Done': 'كل القيام به',
'Result': 'نتيجه',
'of': 'من',
'instances': 'الحالات',
'with': 'مع',
'Click to follow link': 'انقر لمتابعه الارتباط',
'Continue Numbering': 'متابعه الترقيم',
'Bookmark name': 'اسم الإشارة المرجعية',
'Close': 'اغلاق',
'Restart At': 'أعاده التشغيل عند',
'Properties': 'خصائص',
'Name': 'اسم',
'Style type': 'نوع النمط',
'Style based on': 'نمط استنادا إلى',
'Style for following paragraph': 'نمط للفقره التالية',
'Formatting': 'التنسيق',
'Numbering and Bullets': 'الترقيم والتعداد النقطي',
'Numbering': 'ترقيم',
'Update Field': 'تحديث الحقل',
'Edit Field': 'تحرير الحقل',
'Bookmark': 'الإشارة المرجعية',
'Page Setup': 'اعداد الصفحة',
'No bookmarks found': 'لم يتم العثور على إشارات مرجعيه',
'Number format tooltip information': 'تنسيق رقم أحادي المستوي' +
'</br>' + '[' + 'باده' + '%[مستوي الاعداد] للاحقه]' + '
// tslint:disable-next-line:max-line-length
+ 'علي سبيل المثال ، "الفصل 1". سيتم عرض الترقيم مثل' +
'</br>' + 'الفصل الثاني- البند' + '</br>' + 'الفصل الأول- البند' +
'</br>' + 'الفصل نون- البند' + '</br>'

```

```

// tslint:disable-next-line:max-line-length
+ '</br>' + 'تنسيق رقم متعدد الإعدادات:' + '</br>' +
'[مستوي المستوي]' + '% [بأدته' + '</br>' + ']' + '% [لاحقه' + '</br>' +
'[المستوي]' + '% [لاحقه' + '</br>' + 'سيتم عرض الترقيم' + '% 1 2.' +
'مثال ،' + '</br>' + 'البند 1.1' + '</br>' + 'البند 1.2' + '</br>...' + '</br>' +
'نون-البند 1.',
'Format': 'تنسيق',
'Create New Style': 'إنشاء نمط جديد',
'Modify Style': 'تعديل النمط',
'New': 'الجديد',
'Bullets': 'الرصاص',
'Use bookmarks': 'استخدام الإشارات المرجعية',
'Table of Contents': 'جدول المحتويات',
'AutoFit': 'الاحتواء',
'AutoFit to Contents': 'احتواء تلقائي للمحتويات',
'AutoFit to Window': 'احتواء تلقائي للإطار',
'Fixed Column Width': 'عرض العمود الثابت',
'Reset': 'إعادة تعيين',
'Match case': 'حالة المباراة',
'Whole words': 'كلمات كامل',
'Add': 'إضافه',
'Go To': 'الانتقال إلى',
'Search for': 'البحث عن',
'Replace with': 'استبدال',
'TOC 1': 'جدول المحتويات 1',
'TOC 2': 'جدول المحتويات 2',
'TOC 3': 'جدول المحتويات 3',
'TOC 4': 'جدول المحتويات 4',
'TOC 5': 'جدول المحتويات 5',
'TOC 6': 'جدول المحتويات 6',
'TOC 7': 'جدول المحتويات 7',
'TOC 8': 'جدول المحتويات 8',
'TOC 9': 'جدول المحتويات 9',
'Right-to-left': 'من اليمين إلى اليسار',
'Left-to-right': 'من اليسار إلى اليمين',
'Direction': 'الاتجاه',
'Table direction': 'اتجاه الجدول',
'Indent from right': 'مسافة بادئه من اليمين',
'Page': 'صفحه',
'Fit one page': 'احتواء صفحه واحد',
'Fit page width': 'احتواء عرض الصفحة',
// tslint:disable-next-line:max-line-length
'The current page number in the document. Click or tap to
navigate specific page.': 'رقم الصفحة الحالية في المستند. انقر أو اضغط
للتنقل في صفحه معينه'
},
'colorpicker': {
'Apply': 'تطبيق',
'Cancel': 'إلغاء الأمر',
'ModeSwitcher': 'مفتاح كهربائي الوضع'
}
}
});
function App() {
let documenteditor;
React.useEffect(() => {

```

```

        componentDidMount();
    }, []);
    function componentDidMount() {
        let sfdt = `{
            "sections": [
                {
                    "blocks": [
                        {
                            "characterFormat": {
                                "fontSize": 18.0,
                                "fontFamily": "Calibri",
                                "fontFamilyBidi": "Calibri"
                            },
                            "paragraphFormat": {
                                "beforeSpacing": 18.0,
                                "afterSpacing": 30.0,
                                "styleName": "Heading 1",
                                "bidi": true
                            },
                            "inlines": [
                                {
                                    "text": "اعمال المغامرة دورات",
                                    "characterFormat": {
                                        "fontSize": 18.0,
                                        "bidi": true,
                                        "fontSizeBidi": 18.0
                                    }
                                }
                            ]
                        }
                    ]
                }
            ]
        }`;
        setTimeout(() => {
            //Open the document in Document Editor.
            documenteditor.open(sfdt);
        });
    }
    return (<DocumentEditorComponent id="container" height={'330px'}
    ref={(scope) => { documenteditor = scope; }} isReadOnly={false}
    enablePrint={true} enableSelection={true} enableEditor={true}
    enableEditorHistory={true} enableContextMenu={true} enableSearch={true}
    enableOptionsPane={true} enableBookmarkDialog={true}
    enableBordersAndShadingDialog={true} enableFontDialog={true}
    enableTableDialog={true} enableParagraphDialog={true}
    enableHyperlinkDialog={true} enableImageResizer={true}
    enableListDialog={true} enablePageSetupDialog={true} enableSfdtExport={true}
    enableStyleDialog={true} enableTableOfContentsDialog={true}
    enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
    enableTextExport={true} enableWordExport={true} enableRtl={true} locale={'ar-AE'} />);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```


INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
  Selection, Search, Editor, ImageResizer, EditorHistory,
  ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
  TableOfContentsDialog,
  PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
  BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
  CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
import { L10n } from '@syncfusion/ej2-base';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
  Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
  OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
  TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
  ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
  CellOptionsDialog, StylesDialog);
//Load locale constant for Document Editor.
L10n.load({
  'ar-AE': {
    'documenteditor': {
      'Table': 'الجدول',
      'Row': 'الصف',
      'Cell': 'الخلية',
      'Ok': 'موافق',
      'Cancel': 'إلغاء الأمر',
      'Size': 'حجم',
      'Preferred Width': 'العرض المفضل',
      'Points': 'نقاط',
      'Percent': 'المائه',
      'Measure in': 'القياس في',
      'Alignment': 'محاذاه',
      'Left': 'ليسار',
      'Center': 'مركز',
      'Right': 'الحق',
      'Justify': 'تبرير',
      'Indent from left': 'مسافة بادئه من اليسار',
      'Borders and Shading': 'الحدود والتظليل',
      'Options': 'خيارات',
      'Specify height': 'تحديد الارتفاع',
      'At least': 'الاقل',
      'Exactly': 'تماما',
      'Row height is': 'ارتفاع الصف هو',
      'Allow row to break across pages': 'السماح بفصل الصف عبر',
      'Repeat as header row at the top of each page': 'تكرار كصف راس',
      'في اعلي كل صفحه',
      'Vertical alignment': 'محاذاة عمودي',
    },
  },
});

```

```

'Top': 'أعلى',
'Bottom': 'اسفل',
'Default cell margins': 'هوامش الخلية الافتراضية',
'Default cell spacing': 'تباعد الخلايا الافتراضي',
'Allow spacing between cells': 'السماح بالتباعد بين الخلايا',
'Cell margins': 'هوامش الخلية',
'Same as the whole table': 'نفس الجدول بأكمله',
'Borders': 'الحدود',
'None': 'اي',
'Single': 'واحد',
'Dot': 'نقطه',
'DashSmallGap': 'داشسمجاب',
'DashLargeGap': 'الاتحاد الخاص',
'DashDot': 'داشدوت',
'DashDotDot': 'ددهدودوت',
'Double': 'انقر نقرا مزدوجا',
'Triple': 'الثلاثي',
'ThinThickSmallGap': 'فجوه صغيره سميكه رقيق',
'ThickThinSmallGap': 'الفجوة الصغيرة رقيقه سميكه',
'ThinThickThinSmallGap': 'فجوه صغيره سميكه رقيقه الفجوة الصغيرة',
'ThinThickMediumGap': 'فجوه متوسطه سميك',
'ThickThinMediumGap': 'سميكه الفجوة متوسطه رقيقه',
'ThinThickThinMediumGap': 'رقيقه سميكه متوسطه الفجوة',
'ThinThickLargeGap': 'الفجوة الكبيرة رقيقه سميكه',
'ThickThinLargeGap': 'فجوه كبيره رقيقه سميك',
'ThinThickThinLargeGap': 'رقيقه سميكه الفجوة الكبيرة',
'SingleWavy': 'واحد مائج',
'DoubleWavy': 'مزدوج مائج',
'DashDotStroked': 'اندفاعه نقطه القوية',
'Emboss3D': 'D3مزخرف',
'Engrave3D': 'D3نقش',
'Outset': 'البدايه',
'Inset': 'الداخلي',
'Thick': 'سميكه',
'Style': 'نمط',
'Width': 'عرض',
'Height': 'ارتفاع',
'Letter': 'رساله',
'Tabloid': 'التابلويد',
'Legal': 'القانونيه',
'Statement': 'بيان',
'Executive': 'التنفيذي',
'A3': 'A3',
'A4': 'A4',
'A5': 'A5',
'B4': 'B4',
'B5': 'B5',
'Custom Size': 'حجم مخصص',
'Different odd and even': 'مختلفه غريبه وحتى',
'Different first page': 'الصفحة الاولى مختلفه',
'From edge': 'من الحافة',
'Header': 'راس',
'Footer': 'تذييل الصفحه',
'Margin': 'الهوامش',
'Paper': 'الورق',
'Layout': 'تخطيط',
'Orientation': 'التوجه',

```

```

'Landscape': 'المناظر الطبيعيه',
'Portrait': 'صوره',
'Table Of Contents': 'جدول المحتويات',
'Show page numbers': 'إظهار أرقام الصفحات',
'Right align page numbers': 'محاذاة أرقام الصفحات إلى اليمين',
'Nothing': 'شيء',
'Tab leader': 'قائد علامة التبويب',
'Show levels': 'إظهار المستويات',
'Use hyperlinks instead of page numbers': 'استخدام الارتباطات',
'التشعبية بدلا من أرقام الصفحات',
'Build table of contents from': 'بناء جدول محتويات من',
'Styles': 'انماط',
'Available styles': 'الأنماط المتوفرة',
'TOC level': 'مستوي جدول المحتويات',
'Heading': 'عنوان',
'Heading 1': 'عنوان 1',
'Heading 2': 'عنوان 2',
'Heading 3': 'عنوان 3',
'Heading 4': 'عنوان 4',
'Heading 5': 'عنوان 5',
'Heading 6': 'عنوان 6',
'List Paragraph': 'فقره القائمة',
'Normal': 'العادي',
'Outline levels': 'مستويات المخطط التفصيلي',
'Table entry fields': 'حقول إدخال الجدول',
'Modify': 'تعديل',
'Color': 'لون',
'Setting': 'اعداد',
'Box': 'مربع',
'All': 'جميع',
'Custom': 'المخصصه',
'Preview': 'معاينه',
'Shading': 'التظليل',
'Fill': 'ملء',
'Apply To': 'تنطبق على',
'Table Properties': 'خصائص الجدول',
'Cell Options': 'خيارات الخلية',
'Table Options': 'خيارات الجدول',
'Insert Table': 'ادراج جدول',
'Number of columns': 'عدد الاعمده',
'Number of rows': 'عدد الصفوف',
'Text to display': 'النص الذي سيتم عرضه',
'Address': 'عنوان',
'Insert Hyperlink': 'ادراج ارتباط تشعبي',
'Edit Hyperlink': 'تحرير ارتباط تشعبي',
'Insert': 'ادراج',
'General': 'العامة',
'Indentation': 'المسافه البادئه',
'Before text': 'قبل النص',
'Special': 'الخاصه',
'First line': 'السطر الأول',
'Hanging': 'معلقه',
'After text': 'بعد النص',
'By': 'من',
'Before': 'قبل',
'Line Spacing': 'تباعد الأسطر',
'After': 'بعد',

```

```

'At': 'في',
'Multiple': 'متعدده',
'Spacing': 'تباعد',
'Define new Multilevel list': 'تحديد قائمه متعددة الاصعده جديده',
'List level': 'مستوي القائمة',
'Choose level to modify': 'اختر المستوى الذي تريد تعديله',
'Level': 'مستوي',
'Number format': 'تنسيق الأرقام',
'Number style for this level': 'نمط الرقم لهذا المستوى',
'Enter formatting for number': 'إدخال تنسيق لرقم',
'Start at': 'بداية من',
'Restart list after': 'أعاده تشغيل قائمه بعد',
'Position': 'موقف',
'Text indent at': 'المسافة البادئة للنص في',
'Aligned at': 'محاذاة في',
'Follow number with': 'اتبع الرقم مع',
'Tab character': 'حرف علامة التبويب',
'Space': 'الفضاء',
'Arabic': 'العربية',
'UpRoman': 'حتى الروماني',
'LowRoman': 'الرومانية منخفضه',
'UpLetter': '',
'LowLetter': '',
'Number': 'عدد',
'Leading zero': 'يؤدي صفر',
'Bullet': 'رصاصه',
'Ordinal': 'الترتيبيه',
'Ordinal Text': 'النص الترتيبي',
'For East': 'للشرق',
'No Restart': 'لا أعاده تشغيل',
'Font': 'الخط',
'Font style': 'نمط الخط',
'Underline style': 'نمط التسطير',
'Font color': 'لون الخط',
'Effects': 'الاثار',
'Strikethrough': 'يتوسطه',
'Superscript': 'مرتفع',
'Subscript': 'منخفض',
'Double strikethrough': 'خط مزدوج يتوسطه خط',
'Regular': 'العاديه',
'Bold': 'جريئه',
'Italic': 'مائل',
'Cut': 'قطع',
'Copy': 'نسخ',
'Paste': 'لصق',
'Hyperlink': 'الارتباط التشعبي',
'Open Hyperlink': 'فتح ارتباط تشعبي',
'Copy Hyperlink': 'نسخ ارتباط تشعبي',
'Remove Hyperlink': 'أزاله ارتباط تشعبي',
'Paragraph': 'الفقره',
'Linked(Paragraph and Character)': 'مرتبط (فقره وحرف)',
'Character': 'حرف',
'Merge Cells': 'دمج الخلايا',
'Insert Above': 'ادراج أعلاه',
'Insert Below': 'ادراج أدناه',
'Insert Left': 'ادراج إلى اليسار',
'Insert Right': 'ادراج اليمين',

```

```

Delete': 'حذف',
Delete Table': 'حذف جدول',
Delete Row': 'حذف صف',
Delete Column': 'حذف عمود',
File Name': 'اسم الملف',
Format Type': 'نوع التنسيق',
Save': 'حفظ',
Navigation': 'التنقل',
Results': 'نتائج',
Replace': 'استبدال',
Replace All': 'استبدال الكل',
We replaced all': 'استبدلنا جميع',
Find': 'العثور',
No matches': 'لا توجد تطابقات',
All Done': 'كل القيام به',
Result': 'نتيجته',
of': 'من',
instances': 'الحالات',
with': 'مع',
Click to follow link': 'انقر لمتابعه الارتباط',
Continue Numbering': 'متابعه الترقيم',
Bookmark name': 'اسم الإشارة المرجعية',
Close': 'اغلق',
Restart At': 'أعاده التشغيل عند',
Properties': 'خصائص',
Name': 'اسم',
Style type': 'نوع النمط',
Style based on': 'نمط استنادا إلى',
Style for following paragraph': 'نمط للفقرة التالية',
Formatting': 'التنسيق',
Numbering and Bullets': 'الترقيم والتعداد النقطي',
Numbering': 'ترقيم',
Update Field': 'تحديث الحقل',
Edit Field': 'تحرير الحقل',
Bookmark': 'الإشارة المرجعية',
Page Setup': 'اعداد الصفحة',
No bookmarks found': 'لم يتم العثور علي إشارات مرجعيه',
Number format tooltip information': 'تنسيق رقم أحادي المستوي' +
'</br>' + ']' + 'بأدئه' [%مستوي الاعداد] [لاحقه]
// tslint:disable-next-line:max-line-length
+ 'علي سبيل المثال ، "الفصل 1". سيتم عرض الترقيم مثل' +
'</br>' + 'الفصل الثاني- البند' + '</br>' + 'الفصل الأول- البند' +
'</br>' + 'الفصل نون-البند' + '</br>'
// tslint:disable-next-line:max-line-length
+ '</br>' + 'تنسيق رقم متعدد الإعدادات' + '</br>' +
'[%لاحقه] + 'بأدئه' + '</br>' + ']' + 'بأدئه' [%مستوي المستوي]
+ 'علي سبيل المثال ، "1.2". سيتم عرض الترقيم' + '</br>' +
'</br>' + '1.1 البند' + '</br>' + '1.2 البند' + '</br>' + '1.1 البند' +
'</br>' + 'نون-البند' +
Format': 'تنسيق',
Create New Style': 'إنشاء نمط جديد',
Modify Style': 'تعديل النمط',
New': 'الجديد',
Bullets': 'الرماس',
Use bookmarks': 'استخدام الإشارات المرجعية',
Table of Contents': 'جدول المحتويات',

```

```

        'AutoFit': 'الاحتواء',
        'AutoFit to Contents': 'احتواء تلقائي للمحتويات',
        'AutoFit to Window': 'احتواء تلقائي للإطار',
        'Fixed Column Width': 'عرض العمود الثابت',
        'Reset': 'إعادة تعيين',
        'Match case': 'حالة المباراة',
        'Whole words': 'كلمات كامل',
        'Add': 'إضافه',
        'Go To': 'الانتقال إلى',
        'Search for': 'البحث عن',
        'Replace with': 'استبدال',
        'TOC 1': 'جدول المحتويات 1',
        'TOC 2': 'جدول المحتويات 2',
        'TOC 3': 'جدول المحتويات 3',
        'TOC 4': 'جدول المحتويات 4',
        'TOC 5': 'جدول المحتويات 5',
        'TOC 6': 'جدول المحتويات 6',
        'TOC 7': 'جدول المحتويات 7',
        'TOC 8': 'جدول المحتويات 8',
        'TOC 9': 'جدول المحتويات 9',
        'Right-to-left': 'من اليمين إلى اليسار',
        'Left-to-right': 'من اليسار إلى اليمين',
        'Direction': 'الاتجاه',
        'Table direction': 'اتجاه الجدول',
        'Indent from right': 'مسافة بادئه من اليمين',
        'Page': 'صفحه',
        'Fit one page': 'احتواء صفحه واحد',
        'Fit page width': 'احتواء عرض الصفحة',
        // tslint:disable-next-line:max-line-length
        'The current page number in the document. Click or tap to
        navigate specific page.': 'رقم الصفحة الحالية في المستند. انقر أو اضغط
        للتنقل في صفحه معينه'
    },
    'colorpicker': {
        'Apply': 'تطبيق',
        'Cancel': 'إلغاء الأمر',
        'ModeSwitcher': 'مفتاح كهربائي الوضع'
    }
}
});
function App() {
    let documenteditor: DocumentEditorComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function componentDidMount() {
        let sfdt: string = `{
            "sections": [
                {
                    "blocks": [
                        {
                            "characterFormat": {
                                "fontSize": 18.0,
                                "fontFamily": "Calibri",
                                "fontFamilyBidi": "Calibri"
                            },
                            "paragraphFormat": {

```

```

        "beforeSpacing": 18.0,
        "afterSpacing": 30.0,
        "styleName": "Heading 1",
        "bidi": true
    },
    "inlines": [
        {
            "text": "اعمال المغامرة دورات",
            "characterFormat": {
                "fontSize": 18.0,
                "bidi": true,
                "fontSizeBidi": 18.0
            }
        }
    ]
}
]
}
]
}';
setTimeout(() => {
    //Open the document in Document Editor.
    documenteditor.open(sfddt);
});
}
return (
    <DocumentEditorComponent id="container" height={'330px'} ref={(scope)
=> { documenteditor = scope; }}
        readOnly={false} enablePrint={true}
        enableSelection={true} enableEditor={true}
enableEditorHistory={true}
        enableContextMenu={true} enableSearch={true}
enableOptionsPane={true}
        enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
enableHyperlinkDialog={true} enableImageResizer={true}
enableListDialog={true}
        enablePageSetupDialog={true} enableSfddtExport={true}
        enableStyleDialog={true} enableTableOfContentsDialog={true}
        enableTableOptionsDialog={true}
enableTablePropertiesDialog={true}
        enableTextExport={true} enableWordExport={true} enableRtl={true}
locale={'ar-AE'} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Chart in React Document editor component

Document Editor provides chart preservation support. Using Document Editor, you can see the chart reports from your Word document.

The following example shows chart preservation in Document Editor.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-
documenteditor';
function App() {
    let documenteditor = new DocumentEditorComponent(undefined);
    React.useEffect(() => {
        componentDidMount();
    });
}

```



```

    }, []);
    function componentDidMount() {
      let sfdt =
        `{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"textAlignment":"Center","afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"bold":true,"fontSize":12,"fontFamily":"Verdana","fontSizeBidi":12,"fontFamilyBidi":"Verdana"},"inlines":[{"characterFormat":{"bold":true,"fontSize":14,"fontFamily":"Verdana","fontColor":"#17365DFF","styleName":"a","fontSizeBidi":14,"fontFamilyBidi":"Verdana"},"text":"Northwind Management Report"}]}],{"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"inlines":[]},{"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"This management report provides information obtained through data analysis, regarding the "},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"performance of Northwind Traders. This report will pay particular"},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":" "},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":" attention to the "},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"best-selling products, of our company. "},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"The best-selling products of Northwind Traders "},{ "characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Company as follows: "]}],{"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[]},{"rows":[{"cells":[{"blocks":[{"paragraphFormat":{"rightIndent":26.850000381469727,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"S.No"}]}]}],"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"para

```

```

graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inl
ines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi
":10,"fontFamilyBidi":"Times New Roman"},"text":"Product
Name"]]},"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":
false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"
color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5
,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,
"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"col
or":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false
,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"co
lor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow"
: false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineS
tyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#00
0000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"s
pace":0}},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty",
"textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWi
dthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"s
tyleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"charact
erFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamily
Bidi":"Times New Roman"},"text":"Sum of Sales(in
$)"}]},"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":fa
lse,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"c
olor":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"
shadow":false,"space":0},"right":{"color":"#4472C4FF","hasNoneStyle":false,"l
ineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color
":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shad
ow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"
lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"colo
r":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":f
alse,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineSty
le":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#0000
00","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"spa
ce":0}},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty","t
extureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWid
thType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"
verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.39999961
8530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,
"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Singl
e","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","has
NoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":
0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","li
neWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lin
eWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneS
tyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horiz
ontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWi
dth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneSt
yle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"g
ridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,
"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells":[{"blocks":[{"para
graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inl
ines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi
":10,"fontFamilyBidi":"Times New
Roman"},"text":"1"}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","h

```

```

asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},
"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Côte de Blaye"}]}],
"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},
"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":1},
{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"141.396"}]}],
"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},
"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},
"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":2}],
"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s

```

```

shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"2"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Thüringer Rostbratwurst"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"80.368"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}}

```

```

:0,"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single",
lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","h
asNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}
,"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lin
eWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneS
tyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"verti
cal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0
,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregro
undColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220
703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnS
pan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}},"rowFormat":{"
"height":14.3999999618530273,"allowBreakAcrossPages":true,"heightType":"Exactl
y","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":fals
e,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lin
eStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":
"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow
":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"li
neStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color"
:"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":fal
se,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineSty
le":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#
8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":
false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"P
oint"},"gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"},"cells
":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},{"char
acterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Ver
dana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"3"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":
false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundCo
lor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},"blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{}},{"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"Raclette
Courdavault"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWid
th":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle"
: false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom
":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":
0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle
":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalU
p":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"

```

```

shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false
,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"colo
r":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":f
alse,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"
empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"pref
erredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowS
pan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{"paragraphForm
at":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"
characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fon
tFamilyBidi":"Times New
Roman"},"text":["71.155"]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EAADBFF
","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"
space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Singl
e","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","ha
sNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space"
:0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","
lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","h
asNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}
,"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lin
ewidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneS
tyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"verti
cal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0
,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregro
undColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220
703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnS
pan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{
"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactl
y","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":fals
e,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lin
eStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":
"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow
":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"li
neStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color"
:"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":fal
se,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineSty
le":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#
8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":
false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"P
oint","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells
": [{"blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"char
acterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Ve
rdana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":["4"]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle"
: false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundCo

```



```

lor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Tarte au sucre"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"47.234"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells": [{"blocks": [{"para

```

```

graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"5"}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Camembert Pierrot"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"46.825"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnS

```



```

pan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2]],"rowFormat":{
"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactl
y","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":fals
e,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lin
eStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":
"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow
":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"li
neStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color"
:"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":fal
se,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineSty
le":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#
8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":
false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"P
oint","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{
"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":
"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"6"}]]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle"
: false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":
"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{}},
"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"Gnocchi di nonna
Alice"}]]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle"
: false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":
{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false
,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"co
lor":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":fals
e,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow
": false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"line
Style":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#0
00000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"
space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty"
,"textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredW
idthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1
,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"
styleName":"Normal","listFormat":{}},
"characterFormat":{},"inlines":[{"charac
terFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamili
yBidi":"Times New

```

```
Roman"},"text":"42.593"]]]],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"7"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0}},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Manjimup Dried Apples"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0}},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Manjimup Dried Apples"}]}]}
```

```

e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"c
olor": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "
shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": fal
se, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"
color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shado
w": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lin
eStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#
000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false,
"space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty
", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferred
WidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan":
1, "verticalAlignment": "Top", "columnIndex": 1}, {"blocks": [{"paragraphFormat": {"
styleName": "Normal", "listFormat": {}}, "characterFormat": {"fontFamily": "Times New
Roman", "fontSize": 10, "fontFamilyBidi": "Times New
Roman", "text": "41.819"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF
", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "
space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Singl
e", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "ha
sNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "
lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "h
asNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}
, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lin
eWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneS
tyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "verti
cal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0
, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregro
undColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220
703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnS
pan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2}], "rowFormat": {"
height": 14.399999618530273, "allowBreakAcrossPages": true, "heightType": "Exactl
y", "isHeader": false, "borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": fals
e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"col
or": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "sh
adow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lin
eStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color":
"#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow
": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "li
neStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color"
: "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": fal
se, "space": 0}, "horizontal": {"color": "#8EADBFF", "hasNoneStyle": false, "lineSty
le": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "vertical": {"color": "#
8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow":
false, "space": 0}}, "gridBefore": 0, "gridBeforeWidth": 0, "gridBeforeWidthType": "P
oint", "gridAfter": 0, "gridAfterWidth": 0, "gridAfterWidthType": "Point"}}, {"cells
": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "char
acterFormat": {"fontFamily": "Times New Roman", "fontSize": 10, "fontFamilyBidi": "Ve
rdana", "fontFamilyBidi": "Times New
Roman", "text": "8"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space
": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "l
ineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNone
Style": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "
bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineW
idth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNon
eStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "dia

```

```
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Alice Mutton"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"32.698"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0}
```

```

le":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "9"}]}]}, {"cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"columnIndex": 0}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Carnarvon Tigers"}]}]}, {"cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"columnIndex": 1}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "29.171"}]}]}, {"cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}

```

```
, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregro
undColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 37.720001220
703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnS
pan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, "columnIndex": 2}], "rowFormat": {
"height": 14.399999618530273, "allowBreakAcrossPages": true, "heightType": "Exactl
y", "isHeader": false, "borders": {"top": {"color": "#8EAADBFF", "hasNoneStyle": fals
e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"col
or": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "sh
adow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle": false, "lin
eStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color":
"#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow
": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "li
neStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color"
: "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": fal
se, "space": 0}, "horizontal": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineSty
le": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "vertical": {"color": "#
8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow":
false, "space": 0}}, "gridBefore": 0, "gridBeforeWidth": 0, "gridBeforeWidthType": "P
oint", "gridAfter": 0, "gridAfterWidth": 0, "gridAfterWidthType": "Point"}}, {"cells
": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "char
acterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Ve
rdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New
Roman"}, "text": "10"}]}], "cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "
hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "spac
e": 0}, "left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "
lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNon
eStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0},
"bottom": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "line
Width": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNo
neStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "di
agonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWid
th": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle
": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical"
: {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "sh
adow": false, "space": 0}}, "shading": {"backgroundColor": "#FFFFFFF", "foregroundC
olor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.4200000762939
45, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan":
1, "rowSpan": 1, "verticalAlignment": "Top"}, "columnIndex": 0}, {"blocks": [{"paragr
aphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlin
es": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi":
10, "fontFamilyBidi": "Times New Roman"}, "text": "Rössle
Sauerkraut."}]}], "cellFormat": {"borders": {"top": {"color": "#8EAADBFF", "hasNone
Style": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "
left": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWid
th": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EAADBFF", "hasNoneStyle"
: false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom
": {"color": "#8EAADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth":
0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle
": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalU
p": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "
shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false
, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"colo
r": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": f
alse, "space": 0}}, "shading": {"backgroundColor": "#FFFFFFF", "foregroundColor": "
empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "pref
erredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowS
pan": 1, "verticalAlignment": "Top"}, "columnIndex": 1}, {"blocks": [{"paragraphForm
```



```

at":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"25.696"}]}], "cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}}, "shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}}, "gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}}, "grid":[64.71214527422465,292.87942351880633,117.95841899993776], "tableFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}}, "shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"cellSpacing":0,"leftIndent":0,"tableAlignment":"Left","topMargin":0,"rightMargin":0.5,"leftMargin":0.5,"bottomMargin":0,"preferredWidth":475.54998779296875,"preferredWidthType":"Point","bidi":false,"allowAutoFit":true},"description":null,"title":null}, {"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{"fontFamily":"Calibri","fontColor":"#000000FF","fontFamilyBidi":"Calibri"},"inlines":[]}, {"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"The best-selling

```

```

product of the company is Cote de Blaye, being part of the Beverages
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "category. The contribution of this product to the sum of our sales is $ 141.396."}], {"paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}}, "characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}}, {"inlines": []}, {"paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}}, "characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}}, {"inlines": []}, {"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {}, "chartLegend": {"position": "Right", "chartTitleArea": {"fontName": "+mn-1t", "fontSize": 9, "layout": {"layoutX": 0, "layoutY": 0}, "dataFormat": {"fill": {"foreColor": "000000", "rgb": "#000000"}, "line": {"color": "808080", "rgb": "#808080"}}, "chartTitleArea": {"fontName": "+mn-1t", "fontSize": 14, "layout": {"layoutX": 0, "layoutY": 0}, "dataFormat": {"fill": {"foreColor": "000000", "rgb": "#000000"}, "line": {"color": "000000", "rgb": "#000000"}}, "chartArea": {"foreColor": "#FFFFFF", "plotArea": {"foreColor": "#000000FF"}}, "chartCategory": [{"chartData": [{"yValue": 141.396}], "categoryXName": "Côte de Blaye"}, {"chartData": [{"yValue": 80.368}], "categoryXName": "Thüringer Rostbratwurst"}, {"chartData": [{"yValue": 71.155}], "categoryXName": "Raclette Courdavault"}, {"chartData": [{"yValue": 47.234}], "categoryXName": "Tarte au sucre"}, {"chartData": [{"yValue": 46.825}], "categoryXName": "Camembert Pierrot"}, {"chartData": [{"yValue": 42.593}], "categoryXName": "Gnocchi di nonna Alice"}, {"chartData": [{"yValue": 41.819}], "categoryXName": "Manjimup Dried Apples"}, {"chartData": [{"yValue": 32.698}], "categoryXName": "Alice Mutton"}, {"chartData": [{"yValue": 29.171}], "categoryXName": "Carnarvon Tigers"}, {"chartData": [{"yValue": 25.696}], "categoryXName": "Rössle Sauerkraut"}], "chartSeries": [{"dataPoints": [{"fill": {"foreColor": "4472c4", "rgb": "#4472c4"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "ed7d31", "rgb": "#ed7d31"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "a5a5a5", "rgb": "#a5a5a5"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "ffc000", "rgb": "#ffc000"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "5b9bd5", "rgb": "#5b9bd5"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "70ad47", "rgb": "#70ad47"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "264379", "rgb": "#264379"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "9f480e", "rgb": "#9f480e"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "636363", "rgb": "#636363"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"fill": {"foreColor": "9a7200", "rgb": "#9a7200"}, "line": {"color": "ffffff", "rgb": "#ffffff"}}, {"seriesName": "Sales"}], "chartPrimaryCategoryAxis": {"chartTitle": null, "chartTitleArea": {"layout": {}, "dataFormat": {"fill": {}, "line": {}}, "categoryType": "Automatic", "fontSize": 11, "fontName": "Calibri", "numberFormat": "General", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "hasMajorGridLines": false, "hasMinorGridLines": false, "majorTickMark": "TickMark_Outside", "minorTickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosition_NextToAxis"}, "chartPrimaryValueAxis": {"chartTitle": null, "chartTitleArea": {"layout": {}, "dataFormat": {"fill": {}, "line": {}}, "fontSize": 11, "fontName": "Calibri", "maximumValue": 0, "minimumValue": 0, "majorUnit": 0, "hasMajorGridLines": false, "hasMinorGridLines": false, "majorTickMark": "TickMark_Outside", "minorTickMark": "TickMark_None", "tickLabelPosition": "TickLabelPosition_NextToAxis"}, "chartTitle": "Best Selling Products"}, {"chartType": "Pie", "gapWidth": 0, "overlap": 0, "height": 225, "width": 432}], {"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterForma

```



```

t": {}, "inlines": [], { "paragraphFormat": { "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "styleName": "Normal", "listFormat": {} }, "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "inlines": [ { "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "text": "According to the above chart, the total count of the selling products is 24 and the average" }, { "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "text": "sales attributed to this product is $ 5.891 with highest sale $ 15.810 in the month of May in" }, { "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "text": "2014. In the same year, in the month of March the same product reached the amount of $" }, { "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "text": "15.019. These were the highest sales of the product among the other products for the year" }, { "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana" }, "text": "2014." } ] ], "headersFooters": {} }, "characterFormat": { "bold": false, "italic": false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignment": "Normal", "highlightColor": "NoColor", "fontColor": "#000000", "fontSizeBidi": 11, "fontFamilyBidi": "Calibri" }, "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing": 8, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listFormat": {}, "bidi": false, "defaultTabWidth": 36, "enforcement": false, "hashValue": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection", "styles": [ { "name": "Normal", "type": "Paragraph", "paragraphFormat": { "listFormat": {} }, "characterFormat": {}, "next": "Normal" }, { "name": "Heading 1", "type": "Paragraph", "paragraphFormat": { "beforeSpacing": 12, "afterSpacing": 3, "lineSpacing": 1, "lineSpacingType": "Multiple", "outlineLevel": "Level1", "listFormat": {} }, "characterFormat": { "bold": true, "fontSize": 16, "fontFamily": "Arial", "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial" }, "basedOn": "Normal", "link": "Heading 1 Char", "next": "Normal" }, { "name": "Heading 1 Char", "type": "Character", "characterFormat": { "bold": true, "fontSize": 16, "fontFamily": "Arial", "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial" }, "basedOn": "Default Paragraph Font" }, { "name": "Default Paragraph Font", "type": "Character", "characterFormat": {} }, { "name": "Balloon Text", "type": "Paragraph", "paragraphFormat": { "afterSpacing": 0, "lineSpacing": 1, "lineSpacingType": "Multiple", "listFormat": {} }, "characterFormat": { "fontSize": 9, "fontFamily": "Segoe UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe UI" }, "basedOn": "Normal", "link": "Balloon Text Char" }, { "name": "Balloon Text Char", "type": "Character", "characterFormat": { "fontSize": 9, "fontFamily": "Segoe UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe UI" }, "basedOn": "Default Paragraph Font" }, { "name": "a", "type": "Character", "characterFormat": {}, "basedOn": "Default Paragraph Font" }, { "name": "Heading 2", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level2", "listFormat": {} }, "characterFormat": { "fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Normal", "link": "Heading 2 Char", "next": "Normal" }, { "name": "Heading 2 Char", "type": "Character", "characterFormat": { "fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496" }, "basedOn": "Default Paragraph Font" }, { "name": "Heading 3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin

```

```
eSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"},{"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"},{"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4 Char","next":"Normal"},{"name":"Heading 4 Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level5","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5 Char","next":"Normal"},{"name":"Heading 5 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level6","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 6 Char","next":"Normal"},{"name":"Heading 6 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"}],"lists":[],"abstractLists":[]}`;
    setTimeout(() => {
        //Open the default document in Document Editor.
        documenteditor.open(sfdd);
    });
}
return (<DocumentEditorComponent id="container" height={'330px'}
ref={(scope) => { documenteditor = scope; }}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent } from '@syncfusion/ej2-react-documenteditor';
function App() {
    let documenteditor: DocumentEditorComponent= new
    DocumentEditorComponent(undefined);
```

```

    React.useEffect(() => {
      componentDidMount();
    }, []);
    function componentDidMount() {
      let sfdt: string =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"textAlignment":"Center","afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"bold":true,"fontSize":12,"fontFamily":"Verdana","fontSizeBidi":12,"fontFamilyBidi":"Verdana"},"inlines":[{"characterFormat":{"bold":true,"fontSize":14,"fontFamily":"Verdana","fontColor":"#17365DFF","styleName":"a","fontSizeBidi":14,"fontFamilyBidi":"Verdana"},"text":"Northwind Management Report"]}],{"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"inlines":[]},{"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"This management report provides information obtained through data analysis, regarding the "}],{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"performance of Northwind Traders. This report will pay particular"},{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":" "}],{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":" attention to the "}],{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","styleName":"a","fontSizeBidi":10,"fontFamilyBidi":"Verdana"},"text":"best-selling products, of our company."}],{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"The best-selling products of Northwind Traders "}],{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Company as follows: "]}],{"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[]},{"rows":[{"cells":[{"blocks":[{"paragraphFormat":{"rightIndent":26.850000381469727,"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"S.No"}]}]}],"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629

```

```

3945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":
{"1","rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{{"blocks":[{"para
graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inl
ines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi
":10,"fontFamilyBidi":"Times New Roman"},"text":"Product
Name"}]}]},"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":
false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"
color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5
,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,
"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"col
or":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false
,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"col
or":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow"
:false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineS
tyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#00
0000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"s
pace":0},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty",
"textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWi
dthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":1},{{"blocks":[{"paragraphFormat":{"s
tyleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"charact
erFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamily
Bidi":"Times New Roman"},"text":"Sum of Sales(in
$)"}]}]},"cellFormat":{"borders":{"top":{"color":"#4472C4FF","hasNoneStyle":fa
lse,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"c
olor":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"
shadow":false,"space":0},"right":{"color":"#4472C4FF","hasNoneStyle":false,"l
ineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color
":"#4472C4FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shad
ow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"
lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"colo
r":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":f
alse,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineSty
le":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#0000
00","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"spa
ce":0},"shading":{"backgroundColor":"#4472C4FF","foregroundColor":"empty","t
extureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWi
dthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,
"verticalAlignment":"Top"},"columnIndex":2}],"rowFormat":{"height":14.39999961
8530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,
"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Singl
e","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","has
NoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":
0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","li
neWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lin
eWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneS
tyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horiz
ontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWi
dth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneSt
yle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"g
ridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,
"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{{"cells":[{"blocks":[{"para
graphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inl
ines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi

```

```

":10,"fontFamilyBidi":"Times New
Roman"},"text":"1"]]]],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle"
: false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical"
: {"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":
"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"Côte de
Blaye"}]]]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle"
: false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":
{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false
,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"co
lor":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":fals
e,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow
": false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"line
Style":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#0
00000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"
space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty"
,"textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredW
idthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1
,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"
styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"charac
terFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamili
yBidi":"Times New
Roman"},"text":"141.396"}]]]}],"cellFormat":{"borders":{"top":{"color":"#8EADB
FF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,
"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Sing
le","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single",
"lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","
hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0
},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","li
neWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNone
Style":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vert
ical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":
0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregr
oundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.72000122
0703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"column
Span":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}], "rowFormat":
{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exact
ly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":fal

```

```

se,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"co
lor":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"li
neStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color"
:"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shado
w":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"l
ineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color
":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":fa
lse,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineSt
yle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"
#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow"
: false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":
"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point",{
"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},
{"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":
"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"2"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},
"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle"
: false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor
": "empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{
"blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{}},{"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"Thüringer
Rostbratwurst"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNo
neStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}
,"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyl
e":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bott
om":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth
":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyl
e":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagona
lUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0
,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":fal
se,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"co
lor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow"
: false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor
": "empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"pr
eferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"ro
wSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{}},{"characterFormat":{},"inlines":
[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"f
ontFamilyBidi":"Times New
Roman"},"text":"80.368"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF",
"hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"
space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Singl

```



```
e","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"3"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Raclette Courdavault"}]}]},{"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Raclette Courdavault"}]}]}
```

```

":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{
"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"
shadow":false,"space":0},"horizontal":{ "color":"#000000","hasNoneStyle":false
,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{ "colo
r":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":f
alse,"space":0}}, "shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"
empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"pref
erredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rows
pan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{ "paragraphForm
at": { "styleName": "Normal", "listFormat": {} }, "characterFormat": { }, "inlines": [ { "
characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fon
tFamilyBidi": "Times New
Roman", "text": "71.155" } ] } ], "cellFormat": { "borders": { "top": { "color": "#8EADBFF
F", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "
space": 0 }, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Singl
e", "lineWidth": 0.5, "shadow": false, "space": 0 }, "right": { "color": "#8EADBFF", "ha
sNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0 }, "bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "
lineWidth": 0.5, "shadow": false, "space": 0 }, "diagonalDown": { "color": "#000000", "h
asNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0 }
,"diagonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lin
eWidth": 0, "shadow": false, "space": 0 }, "horizontal": { "color": "#000000", "hasNoneS
tyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0 }, "verti
cal": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0
,"shadow": false, "space": 0 } }, "shading": { "backgroundColor": "#D9E2F3FF", "foregro
undColor": "empty", "textureStyle": "TextureNone", "preferredWidth": 37.720001220
703125, "preferredWidthType": "Percent", "cellWidth": 117.95841899993776, "columnS
pan": 1, "rowSpan": 1, "verticalAlignment": "Top", "columnIndex": 2 } ], "rowFormat": {
"height": 14.399999618530273, "allowBreakAcrossPages": true, "heightType": "Exactl
y", "isHeader": false, "borders": { "top": { "color": "#8EADBFF", "hasNoneStyle": fals
e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0 }, "left": { "col
or": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "sh
adow": false, "space": 0 }, "right": { "color": "#8EADBFF", "hasNoneStyle": false, "lin
eStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0 }, "bottom": { "color":
"#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow
": false, "space": 0 }, "diagonalDown": { "color": "#000000", "hasNoneStyle": false, "li
neStyle": "None", "lineWidth": 0, "shadow": false, "space": 0 }, "diagonalUp": { "color"
: "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": fal
se, "space": 0 }, "horizontal": { "color": "#8EADBFF", "hasNoneStyle": false, "lineSty
le": "Single", "lineWidth": 0.5, "shadow": false, "space": 0 }, "vertical": { "color": "#
8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow":
false, "space": 0 } }, "gridBefore": 0, "gridBeforeWidth": 0, "gridBeforeWidthType": "P
oint", "gridAfter": 0, "gridAfterWidth": 0, "gridAfterWidthType": "Point" } }, { "cells
": [ { "blocks": [ { "paragraphFormat": { "styleName": "Normal", "listFormat": {} }, "char
acterFormat": { }, "inlines": [ { "characterFormat": { "fontSize": 10, "fontFamily": "Ve
rdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New
Roman", "text": "4" } ] } ], "cellFormat": { "borders": { "top": { "color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space
": 0 }, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "l
ineWidth": 0.5, "shadow": false, "space": 0 }, "right": { "color": "#8EADBFF", "hasNone
Style": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0 }, "
bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineW
idth": 0.5, "shadow": false, "space": 0 }, "diagonalDown": { "color": "#000000", "hasNon
eStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0 }, "dia
gonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidt
h": 0, "shadow": false, "space": 0 }, "horizontal": { "color": "#000000", "hasNoneStyle"
: false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0 }, "vertical":

```



```

{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Tarte au sucre"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"47.234"}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"g

```

```

ridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,
"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells": [{ "blocks": [{ "para
graphFormat": { "styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inl
ines": [{ "characterFormat": {}, "bookmarkType": 0, "name": "_GoBack"}, { "characterFo
rmat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi
": "Times New
Roman"}, "text": "5"} ]}], "cellFormat": { "borders": { "top": { "color": "#8EADBFF", "h
asNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space
": 0}, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "l
ineWidth": 0.5, "shadow": false, "space": 0}, "right": { "color": "#8EADBFF", "hasNone
Style": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "
bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineW
idth": 0.5, "shadow": false, "space": 0}, "diagonalDown": { "color": "#000000", "hasNon
eStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "dia
gonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidt
h": 0, "shadow": false, "space": 0}, "horizontal": { "color": "#000000", "hasNoneStyle"
: false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical":
{ "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "sha
dow": false, "space": 0}, "shading": { "backgroundColor": "#D9E2F3FF", "foregroundCo
lor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.42000007629394
5, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1
, "rowSpan": 1, "verticalAlignment": "Top"}, "columnIndex": 0}, { "blocks": [{ "paragra
phFormat": { "styleName": "Normal", "listFormat": {}, "characterFormat": {}, "inline
s": [{ "characterFormat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 1
0, "fontFamilyBidi": "Times New Roman"}, "text": "Camembert Pierrot
"} ]}], "cellFormat": { "borders": { "top": { "color": "#8EADBFF", "hasNoneStyle": fals
e, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": { "col
or": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "sh
adow": false, "space": 0}, "right": { "color": "#8EADBFF", "hasNoneStyle": false, "lin
eStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": { "color":
"#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow
": false, "space": 0}, "diagonalDown": { "color": "#000000", "hasNoneStyle": false, "li
neStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": { "color"
: "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": fal
se, "space": 0}, "horizontal": { "color": "#000000", "hasNoneStyle": false, "lineStyle
": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": { "color": "#000000
", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space
": 0}, "shading": { "backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "tex
tureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthT
ype": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "ver
ticalAlignment": "Top"}, "columnIndex": 1}, { "blocks": [{ "paragraphFormat": { "style
Name": "Normal", "listFormat": {}, "characterFormat": {}, "inlines": [{ "characterFo
rmat": { "fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi
": "Times New
Roman"}, "text": "46.825"} ]}], "cellFormat": { "borders": { "top": { "color": "#8EADBFF
", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "
space": 0}, "left": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Singl
e", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": { "color": "#8EADBFF", "ha
sNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space"
: 0}, "bottom": { "color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "
lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": { "color": "#000000", "h
asNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}
, "diagonalUp": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lin
ewidth": 0, "shadow": false, "space": 0}, "horizontal": { "color": "#000000", "hasNoneS
tyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "verti
cal": { "color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0
, "shadow": false, "space": 0}, "shading": { "backgroundColor": "#D9E2F3FF", "foregro

```

```

undColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220
703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnS
pan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}},"rowFormat":{
"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactl
y","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":fals
e,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"col
or":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"sh
adow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lin
eStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":
"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow
":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"li
neStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color"
:"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":fal
se,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineSty
le":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#
8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":
false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"P
oint","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},"cells
":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"char
acterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Ve
rdana","fontSizeBidi":10,"fontFamilyBidi":"Times New
Roman"},"text":"6"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","h
asNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space
":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","l
ineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNone
Style":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"
bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW
idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNon
eStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"dia
gonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidt
h":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle"
:false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":
{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"sha
dow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundCo
lor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.42000007629394
5,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1
,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},"blocks":[{"paragra
phFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inline
s":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":1
0,"fontFamilyBidi":"Times New Roman"},"text":"Gnocchi di nonna
Alice"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle"
:false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":
{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false
,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"co
lor":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":fals
e,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"c
olor":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow
":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"line
Style":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#0
00000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"
space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty"
,"textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredW
idthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1
,"verticalAlignment":"Top"},"columnIndex":1},"blocks":[{"paragraphFormat":{"
styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"charac

```

```

terFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"42.593"]]],"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}},"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"},"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{,"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"7"}]]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{}},"characterFormat":{,"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Manjimup Dried Apples"}]]},"cellFormat":{"borders":{"top":{"color":"#8EAADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left"

```

```

:{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top","columnIndex":1},{ "blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"41.819"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},{"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top","columnIndex":2}], "rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},{"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}]}, {"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"8"}]}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineW

```

```

idth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{ "blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Alice Mutton"]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"preferredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{ "blocks": [{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"32.698"]}]}, {"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}]},"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":

```



```

:{"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{ "cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "9"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 13.420000076293945, "preferredWidthType": "Percent", "cellWidth": 64.71214527422465, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "Carnarvon Tigers"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}, {"blocks": [{"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi": "Times New Roman"}, "text": "29.171"}]}], "cellFormat": {"borders": {"top": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "left": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "right": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "bottom": {"color": "#8EADBFF", "hasNoneStyle": false, "lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0}, "diagonalDown": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "diagonalUp": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "horizontal": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}, "vertical": {"color": "#000000", "hasNoneStyle": false, "lineStyle": "None", "lineWidth": 0, "shadow": false, "space": 0}}, "shading": {"backgroundColor": "#D9E2F3FF", "foregroundColor": "empty", "textureStyle": "TextureNone"}, "preferredWidth": 48.86000061035156, "preferredWidthType": "Percent", "cellWidth": 292.87942351880633, "columnSpan": 1, "rowSpan": 1, "verticalAlignment": "Top"}]}]}]

```

```

type":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"shading":{"backgroundColor":"#D9E2F3FF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}},{cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"10"}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":13.420000076293945,"preferredWidthType":"Percent","cellWidth":64.71214527422465,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":0},{blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"Rössle Sauerkraut."]}]}]}],"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":48.86000061035156,"pref

```



```

erredWidthType":"Percent","cellWidth":292.87942351880633,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":1},{
"blocks":[{"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"25.696"}]}]},"cellFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"vertical":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"preferredWidth":37.720001220703125,"preferredWidthType":"Percent","cellWidth":117.95841899993776,"columnSpan":1,"rowSpan":1,"verticalAlignment":"Top"},"columnIndex":2}],
"rowFormat":{"height":14.399999618530273,"allowBreakAcrossPages":true,"heightType":"Exactly","isHeader":false,"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"gridBefore":0,"gridBeforeWidth":0,"gridBeforeWidthType":"Point","gridAfter":0,"gridAfterWidth":0,"gridAfterWidthType":"Point"}]},
"grid":[64.71214527422465,292.87942351880633,117.95841899993776],
"tableFormat":{"borders":{"top":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"left":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"right":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"bottom":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"diagonalDown":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"diagonalUp":{"color":"#000000","hasNoneStyle":false,"lineStyle":"None","lineWidth":0,"shadow":false,"space":0},"horizontal":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0},"vertical":{"color":"#8EADBFF","hasNoneStyle":false,"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0}},"shading":{"backgroundColor":"#FFFFFF","foregroundColor":"empty","textureStyle":"TextureNone"},"cellSpacing":0,"leftIndent":0,"tableAlignment":"Left","topMargin":0,"rightMargin":0.5,"leftMargin":0.5,"bottomMargin":0,"preferredWidth":475.54998779296875,"preferredWidthType":"Point","bidi":false,"allowAutoFit":true},
"description":null,"title":null},{
"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFormat":{"fontFamily":"Calibri","fontColor":"#000000FF","fontFamilyBidi":"Calibri"},"inlines":[]},{
"paragraphFormat":{"afterSpacing":0,"styleName":"Normal","listFormat":{},"characterFor

```

```

mat":{},"inlines":[{"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"The best-selling product of the company is Cote de Blaye, being part of the Beverages"}, {"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"text":"category. The contribution of this product to the sum of our sales is $141.396."}],{"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]}, {"paragraphFormat":{"afterSpacing":0,"lineSpacing":1,"lineSpacingType":"Multiple","styleName":"Normal","listFormat":{},"characterFormat":{"fontSize":10,"fontFamily":"Verdana","fontSizeBidi":10,"fontFamilyBidi":"Times New Roman"},"inlines":[]}, {"paragraphFormat":{"styleName":"Normal","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"chartLegend":{"position":"Right","chartTitleArea":{"fontName":"+mn-1t","fontSize":9,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"808080","rgb":"#808080"}}}}, {"chartTitleArea":{"fontName":"+mn-1t","fontSize":14,"layout":{"layoutX":0,"layoutY":0},"dataFormat":{"fill":{"foreColor":"000000","rgb":"#000000"},"line":{"color":"000000","rgb":"#000000"}}}}, {"chartArea":{"foreColor":"#FFFFFFF","plotArea":{"foreColor":"#000000FF"},"chartCategory":[{"chartData":[{"yValue":141.396}], "categoryXName":"Côte de Blaye"}, {"chartData":[{"yValue":80.368}], "categoryXName":"Thüringer Rostbratwurst"}, {"chartData":[{"yValue":71.155}], "categoryXName":"Raclette Courdavault"}, {"chartData":[{"yValue":47.234}], "categoryXName":"Tarte au sucre"}, {"chartData":[{"yValue":46.825}], "categoryXName":"Camembert Pierrot"}, {"chartData":[{"yValue":42.593}], "categoryXName":"Gnocchi di nonna Alice"}, {"chartData":[{"yValue":41.819}], "categoryXName":"Manjimup Dried Apples"}, {"chartData":[{"yValue":32.698}], "categoryXName":"Alice Mutton"}, {"chartData":[{"yValue":29.171}], "categoryXName":"Carnarvon Tigers"}, {"chartData":[{"yValue":25.696}], "categoryXName":"Rössle Sauerkraut"}], "chartSeries":[{"dataPoints":[{"fill":{"foreColor":"4472c4","rgb":"#4472c4"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"ed7d31","rgb":"#ed7d31"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"a5a5a5","rgb":"#a5a5a5"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"ffc000","rgb":"#ffc000"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"5b9bd5","rgb":"#5b9bd5"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"70ad47","rgb":"#70ad47"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"264379","rgb":"#264379"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"9f480e","rgb":"#9f480e"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"636363","rgb":"#636363"},"line":{"color":"ffffff","rgb":"#ffffff"}}, {"fill":{"foreColor":"9a7200","rgb":"#9a7200"},"line":{"color":"ffffff","rgb":"#ffffff"}}], "seriesName":"Sales"}], "chartPrimaryCategoryAxis":{"chartTitle":null, "chartTitleArea":{"layout":{},"dataFormat":{"fill":{},"line":{}}, "categoryType":"Automatic", "fontSize":11, "fontName":"Calibri", "numberFormat":"General", "maximumValue":0, "minimumValue":0, "majorUnit":0, "hasMajorGridLines":false, "hasMinorGridLines":false, "majorTickMark":"TickMark_Outside", "minorTickMark":"TickMark_None", "tickLabelPosition":"TickLabelPosition_NextToAxis"}, "chartPrimaryValueAxis":{"chartTitle":null, "chartTitleArea":{"layout":{},"dataFormat":{"fill":{},"line":{}}, "fontSize":11, "fontName":"Calibri", "maximumValue":0, "minimumValue":0, "majorUnit":0, "hasMajorGridLines":false, "hasMinorGridLines":false, "majorTickMark":"TickMark_Outside", "minorTickMark":"TickMark_None", "tickLabelPosition":"TickLabelPosition_NextToAxis"}, "chartTitle":"Best Selling

```

```

Products", "chartType": "Pie", "gapWidth": 0, "overlap": 0, "height": 225, "width": 432
]]], {"paragraphFormat": {"styleName": "Normal", "listFormat": {}}, "characterForma
t": {}, "inlines": []}, {"paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1, "lin
eSpacingType": "Multiple", "styleName": "Normal", "listFormat": {}}, "characterForm
at": {"fontSize": 10, "fontFamily": "Verdana", "fontSizeBidi": 10, "fontFamilyBidi":
"Verdana"}, "inlines": [{"characterFormat": {"fontSize": 10, "fontFamily": "Verdana
", "styleName": "a", "fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "Accor
ding to the above chart, the total count of the selling products is 24 and
the average
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "
fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "sales attributed to this
product is $ 5.891 with highest sale $ 15.810 in the month of May in
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "
fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "2014. In the same year,
in the month of March the same product reached the amount of $
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "
fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "15.019. These were the
highest sales of the product among the other products for the year
"}, {"characterFormat": {"fontSize": 10, "fontFamily": "Verdana", "styleName": "a", "
fontSizeBidi": 10, "fontFamilyBidi": "Verdana"}, "text": "2014."}]]], "headersFoote
rs": {}], "characterFormat": {"bold": false, "italic": false, "fontSize": 11, "fontFa
mily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignment"
: "Normal", "highlightColor": "NoColor", "fontColor": "#000000", "fontSizeBidi": 11,
"fontFamilyBidi": "Calibri"}, "paragraphFormat": {"leftIndent": 0, "rightIndent": 0
, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing":
8, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listFormat":
{}}, {"bidi": false}, {"defaultTabWidth": 36, "enforcement": false, "hashValue": "", "sal
tValue": "", "formatting": false, "protectionType": "NoProtection", "styles": [{"nam
e": "Normal", "type": "Paragraph", "paragraphFormat": {"listFormat": {}}, "character
Format": {}, "next": "Normal"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 3,
"lineSpacing": 1, "lineSpacingType": "Multiple", "outlineLevel": "Level1", "listFor
mat": {}}, "characterFormat": {"bold": true, "fontSize": 16, "fontFamily": "Arial", "b
oldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial"}, {"basedOn": "Normal",
"link": "Heading 1 Char", "next": "Normal"}, {"name": "Heading 1
Char", "type": "Character", "characterFormat": {"bold": true, "fontSize": 16, "fontFa
mily": "Arial", "boldBidi": true, "fontSizeBidi": 16, "fontFamilyBidi": "Arial"}, {"ba
sedOn": "Default Paragraph Font", "next": "Normal"}, {"name": "Default Paragraph
Font", "type": "Character", "characterFormat": {}}, {"name": "Balloon
Text", "type": "Paragraph", "paragraphFormat": {"afterSpacing": 0, "lineSpacing": 1,
"lineSpacingType": "Multiple", "listFormat": {}}, "characterFormat": {"fontSize": 9
, "fontFamily": "Segoe UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe
UI"}, {"basedOn": "Normal", "link": "Balloon Text Char"}, {"name": "Balloon Text
Char", "type": "Character", "characterFormat": {"fontSize": 9, "fontFamily": "Segoe
UI", "fontSizeBidi": 9, "fontFamilyBidi": "Segoe UI"}, {"basedOn": "Default
Paragraph
Font"}, {"name": "a", "type": "Character", "characterFormat": {}, {"basedOn": "Default
Paragraph Font"}, {"name": "Heading
2", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el2", "listFormat": {}}, "characterFormat": {"fontSize": 13, "fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, {"basedOn": "Normal", "link": "Heading 2
Char", "next": "Normal"}, {"name": "Heading 2
Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, {"basedOn": "Default Paragraph
Font"}, {"name": "Heading

```

```

3", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {}, "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal" }, { "name": "Heading 3 Char", "type": "Character", "characterFormat": { "fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" }, { "name": "Heading 4", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}, "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal" }, { "name": "Heading 4 Char", "type": "Character", "characterFormat": { "italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 5", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal" }, { "name": "Heading 5 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#2F5496", "basedOn": "Default Paragraph Font" }, { "name": "Heading 6", "type": "Paragraph", "paragraphFormat": { "leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}, "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal" }, { "name": "Heading 6 Char", "type": "Character", "characterFormat": { "fontFamily": "Calibri Light", "fontColor": "#1F3763", "basedOn": "Default Paragraph Font" } }], "lists": [], "abstractLists": []}`;
    setTimeout(() => {
        //Open the default document in Document Editor.
        documenteditor.open(sfdd);
    });
}

return (
    <DocumentEditorComponent id="container" height={ '330px' }
    ref={ (scope) => { documenteditor = scope; } } />
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>SynCFusion React Button</title>

```

```

<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Supported Chart Types

The following chart types are supported in document editor

- Scatter_Markers
- Bubble
- Area
- Area_Stacked
- AreaStacked100
- Bar_Clustered
- Bar_Stacked
- BarStacked100
- Column_Clustered
- Column_Stacked

- *ColumnStacked100*
- *Pie*
- *Doughnut*
- *Line*
- *Line_Markers*
- *LineMarkersStacked*
- *LineMarkersStacked_100*
- *Line_Stacked*
- *LineStacked100*

Restrict editing in React Document editor component

Document Editor provides support to restrict editing. When the protected document includes range permission, then unique user or user group only authorized to edit separate text area.

Set current user

You can use the `currentUser` property to authorize the current document user by name, email, or user group name.

The following code shows how to set `currentUser`

```
`ts
documentEditor.currentUser = 'engineer@mycompany.com';
`
```

Highlighting the text area

You can highlight the editable region of the current user using the `userColor` property.

The following code shows how to set `userColor`.

```
`ts
documentEditor.userColor = '#fff000';
`
```

You can toggle the highlight the editable region value using the "highlightEditableRanges" property.

The following code shows how to toggle the highlight editable region value.

```
`typescript
container.documentEditor.documentEditorSettings.highlightEditableRanges = true;
`
```

Restrict Editing Pane

Restrict Editing Pane provides the following options to manage the document:

- To apply formatting restrictions to the current document, select the allow formatting check box.
- To apply editing restrictions to the current document, select the read only check box.
- To add users to the current document, select more users option and add user from the popup dialog.
- To include range permission to the current document, select parts of the document and choose users who are allowed to freely edit them from the listed check box.

- To apply the chosen editing restrictions, click the **YES, START ENFORCING PROTECTION** button. A dialog box displays asking for a password to protect.
- To stop protection, select **STOP PROTECTION** button. A dialog box displays asking for a password to stop protection.

The following code shows Restrict Editing Pane. To unprotect the document, use password '123'.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorContainerComponent.Inject(Toolbar);
export class Default extends React.Component {
  container;
  componentDidMount() {
    let sfdt =
`{"sections":[{"blocks":[{"characterFormat":{"fontSize":14.0,"fontSizeBidi":14.0},"paragraphFormat":{"lineSpacing":32.0,"lineSpacingType":"Exactly","styleName":"Normal"},"inlines":[{"text":"Name","characterFormat":{"bold":true,"fontSize":14.0,"boldBidi":true,"fontSizeBidi":14.0}},{text":":","characterFormat":{"fontSize":14.0,"fontSizeBidi":14.0}}]},{rows":[{"rowFormat":{"allowBreakAcrossPages":true,"isHeader":false,"height":20.0,"heightType":"AtLeast","borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}]},"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal"},"inlines":[{"editRangeId":"1348272392","columnFirst":0,"columnLast":0,"user":"engineer@mycompany.com"}],"text":"Enter name"}],"editRangeId":"1348272392","editableRangeStart":{"editRangeId":"1348272392","columnFirst":0,"columnLast":0,"user":"engineer@mycompany.com"}}]}],"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredWidthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}]},"title":null,"description":null,"tableFormat":{"allowAutoFit":true,"leftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto","borders":{"left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneS
```



```

type":false}, "right":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}}, {"bidi":false}}, {"characterFormat":{"bold":true, "fontSize":14.0, "boldBidi":true, "fontSizeBidi":14.0}, "paragraphFormat":{"lineSpacing":32.0, "lineSpacingType":"Exactly", "styleName":"Normal"}, "inlines":[{"text":"Designation:", "characterFormat":{"bold":true, "fontSize":14.0, "boldBidi":true, "fontSizeBidi":14.0}}]}, {"rows":[{"rowFormat":{"allowBreakAcrossPages":true, "isHeader":false, "height":20.0, "heightType":"AtLeast", "borders":{"left":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "right":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}}}}, {"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal"}, "inlines":[{"editRangeId":"808933422", "columnFirst":0, "columnLast":0, "user":"engineer@mycompany.com"}, {"text":"Enter designation"}, {"editRangeId":"808933422", "editableRangeStart":{"editRangeId":"808933422", "columnFirst":0, "columnLast":0, "user":"engineer@mycompany.com"}}]}], "cellFormat":{"columnSpan":1, "rowSpan":1, "preferredWidth":467.5, "preferredWidthType":"Point", "verticalAlignment":"Center", "isSamePaddingAsTable":true, "borders":{"left":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "right":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}}}}}], "title":null, "description":null, "tableFormat":{"allowAutoFit":true, "leftIndent":0.0, "tableAlignment":"Left", "preferredWidthType":"Auto", "borders":{"left":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "right":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "horizontal":{"lineStyle":"Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDown":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp":{"lineStyle":"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}}, {"bidi":false}}, {"characterFormat":{"bold":true, "fontSize":14.0, "boldBidi":true, "fontSizeBidi":14.0}, "paragraphFormat":{"lineSpacing":32.0, "lineSpacingType":"Exactly", "styleName":"Normal"}, "inlines":[{"text":"Email Address:", "characterFormat":{"bold":true, "fontSize":14.0, "boldBidi":true, "fon

```



```

tSizeBidi":14.0}}, {"name": "_GoBack", "bookmarkType":0}, {"name": "_GoBack", "book
markType":1}}], {"rows": [{"rowFormat": {"allowBreakAcrossPages":true, "isHeader"
:false, "height":20.0, "heightType": "AtLeast", "borders": {"left": {"lineStyle": "N
one", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "right"
: {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle
":false}, "top": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0
, "hasNoneStyle":false}, "bottom": {"lineStyle": "None", "lineWidth":0.0, "shadow":
false, "space":0.0, "hasNoneStyle":false}, "vertical": {"lineStyle": "None", "lineW
idth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "horizontal": {"lin
eStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":fals
e}, "diagonalDown": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":
0.0, "hasNoneStyle":false}, "diagonalUp": {"lineStyle": "None", "lineWidth":0.0, "s
hadow":false, "space":0.0, "hasNoneStyle":false}}, "cells": [{"blocks": [{"paragr
aphFormat": {"styleName": "Normal"}, "inlines": [{"editRangeId": "810441411", "colu
mnFirst":0, "columnLast":0, "user": "engineer@mycompany.com"}], {"text": "Enter
email
address"}, {"editRangeId": "810441411", "editableRangeStart": {"editRangeId": "810
441411", "columnFirst":0, "columnLast":0, "user": "engineer@mycompany.com"}}]]], "
cellFormat": {"columnSpan":1, "rowSpan":1, "preferredWidth":467.5, "preferredWidt
hType": "Point", "verticalAlignment": "Center", "isSamePaddingAsTable":true, "bord
ers": {"left": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "
hasNoneStyle":false}, "right": {"lineStyle": "None", "lineWidth":0.0, "shadow":fal
se, "space":0.0, "hasNoneStyle":false}, "top": {"lineStyle": "None", "lineWidth":0.
0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom": {"lineStyle": "Non
e", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical
": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyl
e":false}, "horizontal": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "sp
ace":0.0, "hasNoneStyle":false}, "diagonalDown": {"lineStyle": "None", "lineWidth"
:0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp": {"lineStyl
e": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}}]
}], {"title": null, "description": null, "tableFormat": {"allowAutoFit": true, "leftI
ndent":0.0, "tableAlignment": "Left", "preferredWidthType": "Auto", "borders": {"le
ft": {"lineStyle": "Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNone
Style":false}, "right": {"lineStyle": "Single", "lineWidth":0.5, "shadow":false, "s
pace":0.0, "hasNoneStyle":false}, "top": {"lineStyle": "Single", "lineWidth":0.5, "
shadow":false, "space":0.0, "hasNoneStyle":false}, "bottom": {"lineStyle": "Single
", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyle":false}, "vertical"
: {"lineStyle": "Single", "lineWidth":0.5, "shadow":false, "space":0.0, "hasNoneStyl
e":false}, "horizontal": {"lineStyle": "Single", "lineWidth":0.5, "shadow":false,
"space":0.0, "hasNoneStyle":false}, "diagonalDown": {"lineStyle": "None", "lineWid
th":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalUp": {"lines
tyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}
}, {"bidi":false}}, {"characterFormat": {"bold":true, "fontSize":14.0, "boldBidi":t
rue, "fontSizeBidi":14.0}, "paragraphFormat": {"lineSpacing":32.0, "lineSpacingTy
pe": "Exactly", "styleName": "Normal"}, "inlines": [{"text": "Feedbacks:", "characte
rFormat": {"bold":true, "fontSize":14.0, "boldBidi":true, "fontSizeBidi":14.0}}]]
}, {"rows": [{"rowFormat": {"allowBreakAcrossPages":true, "isHeader":false, "height
":20.0, "heightType": "AtLeast", "borders": {"left": {"lineStyle": "None", "lineWidt
h":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "right": {"lineStyle":
"None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "top"
: {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle
":false}, "bottom": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":
0.0, "hasNoneStyle":false}, "vertical": {"lineStyle": "None", "lineWidth":0.0, "sha
dow":false, "space":0.0, "hasNoneStyle":false}, "horizontal": {"lineStyle": "None"
, "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneStyle":false}, "diagonalDo
wn": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "space":0.0, "hasNoneSt
yle":false}, "diagonalUp": {"lineStyle": "None", "lineWidth":0.0, "shadow":false, "

```

```
space":0.0,"hasNoneStyle":false}}},{"cells":[{"blocks":[{"paragraphFormat":{"styleType":"Normal"},"inlines":[{"editRangeId":"1016946268","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}],"text":"Enter the feedbacks"},{"editRangeId":"1016946268","editableRangeStart":{"editRangeId":"1016946268","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}}]}],{"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredWidthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}}}}],"title":null,"description":null,"tableFormat":{"allowAutoFit":true,"leftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto","borders":{"left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},"bidi":false}},{"characterFormat":{"bold":true,"fontSize":14.0,"boldBidi":true,"fontSizeBidi":14.0},"paragraphFormat":{"lineSpacing":32.0,"lineSpacingType":"Exactly","styleName":"Normal"},"inlines":[{"text":"Review comments","characterFormat":{"bold":true,"fontSize":14.0,"boldBidi":true,"fontSizeBidi":14.0}}]},{"rows":[{"rowFormat":{"allowBreakAcrossPages":true,"isHeader":false,"height":20.0,"heightType":"AtLeast","borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},"cells":[{"blocks":[{"paragraphFormat":{"styleName":"Normal"},"inlines":[{"editRangeId":"1373703080","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}],"text":"Enter the comments"},{"editRangeId":"1373703080","editableRangeStart":{"editRangeId":"1373703080","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}}]}],{"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredWidthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}}}}]]
```

```

al":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}}}}},"title":null,"description":null,"tableFormat":{"allowAutoFit":true,"leftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto","borders":{"left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},"bidi":false}},{"paragraphFormat":{"styleName":"Normal"},"inlines":[]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"styleName":"Header"},"inlines":[{"text":"Employee's Details"}]}]}},{"sectionFormat":{"headerDistance":36.0,"footerDistance":36.0,"pageWidth":612.0,"pageHeight":792.0,"leftMargin":72.0,"rightMargin":72.0,"topMargin":72.0,"bottomMargin":72.0,"differentFirstPage":false,"differentOddAndEvenPages":false,"bidi":false}},{"characterFormat":{"fontSize":11.0,"fontFamily":"Calibri","fontSizeBidi":11.0,"fontFamilyBidi":"Calibri"},"paragraphFormat":{"afterSpacing":8.0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple"},"background":{"color":"#FFFFFF"},"styles":[{"type":"Paragraph","name":"Normal","next":"Normal"},{"type":"Character","name":"Default Paragraph Font"},{"type":"Paragraph","name":"List Paragraph","basedOn":"Normal","paragraphFormat":{"leftIndent":36.0,"contextualSpacing":true}},{"type":"Paragraph","name":"Header","basedOn":"Normal","next":"Normal","link":"Header Char","paragraphFormat":{"afterSpacing":0.0,"lineSpacing":1.0,"lineSpacingType":"Multiple","tabs":[{"tabJustification":"Center","position":234.0,"tabLeader":"None","deletePosition":0.0},{"tabJustification":"Right","position":468.0,"tabLeader":"None","deletePosition":0.0}]}},{"type":"Character","name":"Header Char","basedOn":"Default Paragraph Font"},{"type":"Paragraph","name":"Footer","basedOn":"Normal","link":"Footer Char","paragraphFormat":{"afterSpacing":0.0,"lineSpacing":1.0,"lineSpacingType":"Multiple","tabs":[{"tabJustification":"Center","position":234.0,"tabLeader":"None","deletePosition":0.0},{"tabJustification":"Right","position":468.0,"tabLeader":"None","deletePosition":0.0}]}},{"type":"Character","name":"Footer Char","basedOn":"Default Paragraph Font"}]},"defaultTabWidth":36.0,"formatting":false,"protectionType":"ReadOnly","enforcement":true,"hashValue":"TQGuJuLceQCe234Ygx4q6NFgHpRMfilhjFTojyKzbQOkwk+ckEM9CjNidkiUhSR/e/7sfMxO4sbPcg/DBzztMg==","saltValue":"FXbkr1RtDIIIZfwlM7ldMg=="}`;

setTimeout(() => {
    //Open the document in Document Editor.
    this.container.documentEditor.open(sfdt);
});
this.container.serviceUrl =
'https://ej2services.syncfusion.com/production/web-services/api/documenteditor/';
}
render() {

```

```

    return (<DocumentEditorContainerComponent id="container"
height={'590px'} enableToolbar={true} ref={(scope) => { this.container =
scope; }}/>);
  }
}
ReactDOM.render(<Default />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-
react-documenteditor';
//Inject require modules.
DocumentEditorContainerComponent.Inject(Toolbar);
export class Default extends React.Component<{}, {}> {
  public container: DocumentEditorContainerComponent;
  public componentDidMount(): void {
    let sfdt: string =
`{"sections": [{"blocks": [{"characterFormat": {"fontSize":14.0,"fontSizeBidi":1
4.0},"paragraphFormat": {"lineSpacing":32.0,"lineSpacingType": "Exactly", "style
Name": "Normal"}, "inlines": [{"text": "Name", "characterFormat": {"bold": true, "fon
tSize":14.0,"boldBidi": true, "fontSizeBidi":14.0}}, {"text": ":", "characterForma
t": {"fontSize":14.0,"fontSizeBidi":14.0}}]}, {"rows": [{"rowFormat": {"allowBrea
kAcrossPages": true, "isHeader": false, "height":20.0,"heightType": "AtLeast", "bor
ders": {"left": {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "space":0.0,
"hasNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth":0.0,"shadow": fa
lse, "space":0.0,"hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth":0
.0,"shadow": false, "space":0.0,"hasNoneStyle": false}, "bottom": {"lineStyle": "No
ne", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}, "vertica
l": {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneSty
le": false}, "horizontal": {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "s
pace":0.0,"hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth
":0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}, "diagonalUp": {"lineSty
le": "None", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}}]
}, {"cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal"}, "inlines": [{"e
ditRangeId": "1348272392", "columnFirst":0, "columnLast":0, "user": "engineer@myco
mpany.com"}], {"text": "Enter
name"}], {"editRangeId": "1348272392", "editableRangeStart": {"editRangeId": "13482
72392", "columnFirst":0, "columnLast":0, "user": "engineer@mycompany.com"}}]}], "c
ellFormat": {"columnSpan":1, "rowSpan":1, "preferredWidth":467.5, "preferredWidth
Type": "Point", "verticalAlignment": "Center", "isSamePaddingAsTable": true, "borde
rs": {"left": {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "space":0.0, "h
asNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth":0.0,"shadow": fals
e, "space":0.0,"hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth":0.0
,"shadow": false, "space":0.0,"hasNoneStyle": false}, "bottom": {"lineStyle": "None
", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}, "vertical"
: {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneStyle
": false}, "horizontal": {"lineStyle": "None", "lineWidth":0.0,"shadow": false, "spa
ce":0.0,"hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth":
0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}, "diagonalUp": {"lineStyle
": "None", "lineWidth":0.0,"shadow": false, "space":0.0,"hasNoneStyle": false}}]}]
}], "title": null, "description": null, "tableFormat": {"allowAutoFit": true, "leftIn
dent":0.0, "tableAlignment": "Left", "preferredWidthType": "Auto", "borders": {"lef

```

```

t":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneS
tyle":false},"right":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"sp
ace":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single","lineWidth":0.5,"s
hadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"Single"
,"lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertical":
{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyl
e":false},"horizontal":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"
space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWidt
h":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lineSt
yle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}
,"bidi":false}},{"characterFormat":{"bold":true,"fontSize":14.0,"boldBidi":tr
ue,"fontSizeBidi":14.0},"paragraphFormat":{"lineSpacing":32.0,"lineSpacingTyp
e":"Exactly","styleName":"Normal"},"inlines":[{"text":"Designation:"},"charact
erFormat":{"bold":true,"fontSize":14.0,"boldBidi":true,"fontSizeBidi":14.0}}]
},{ "rows":[{"rowFormat":{"allowBreakAcrossPages":true,"isHeader":false,"heigh
t":20.0,"heightType":"AtLeast","borders":{"left":{"lineStyle":"None","lineWidt
h":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"right":{"lineStyle"
:"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"top
":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyl
e":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space"
:0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None","lineWidth":0.0,"sh
adow":false,"space":0.0,"hasNoneStyle":false},"horizontal":{"lineStyle":"None
","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalD
own":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneS
tyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,
"space":0.0,"hasNoneStyle":false}}},"cells":[{"blocks":[{"paragraphFormat":{"
styleName":"Normal"},"inlines":[{"editRangeId":"808933422","columnFirst":0,"c
olumnLast":0,"user":"engineer@mycompany.com"}],"text":"Enter
designation"}],"editRangeId":"808933422","editableRangeStart":{"editRangeId":
"808933422","columnFirst":0,"columnLast":0,"user":"engineer@mycompany.com"}}]
}], "cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferred
WidthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"
borders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0
.0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow"
:false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth
":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":
"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"vert
ical":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNone
Style":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false
,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWi
dth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"line
Style":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false
}}}}}], "title":null,"description":null,"tableFormat":{"allowAutoFit":true,"l
eftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto","borders":
{"left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"has
NoneStyle":false},"right":{"lineStyle":"Single","lineWidth":0.5,"shadow":fals
e,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single","lineWidth":0
.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"Si
ngle","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"verti
cal":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNon
eStyle":false},"horizontal":{"lineStyle":"Single","lineWidth":0.5,"shadow":fa
lse,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lin
eWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"l
ineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":fa
lse}}},"bidi":false}},{"characterFormat":{"bold":true,"fontSize":14.0,"boldBid
i":true,"fontSizeBidi":14.0},"paragraphFormat":{"lineSpacing":32.0,"lineSpaci
ngType":"Exactly","styleName":"Normal"},"inlines":[{"text":"Email

```

```
Address:", {"characterFormat": {"bold": true, "fontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0}}, {"name": "_GoBack", "bookmarkType": 0}, {"name": "_GoBack", "bookmarkType": 1}]]], {"rows": [{"rowFormat": {"allowBreakAcrossPages": true, "isHeader": false, "height": 20.0, "heightType": "AtLeast", "borders": {"left": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}}], "cells": [{"blocks": [{"paragraphFormat": {"styleName": "Normal"}, "inlines": [{"editRangeId": "810441411", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com"}, {"text": "Enter email address"}], {"editRangeId": "810441411", "editableRangeStart": {"editRangeId": "810441411", "columnFirst": 0, "columnLast": 0, "user": "engineer@mycompany.com"}}]}], "cellFormat": {"columnSpan": 1, "rowSpan": 1, "preferredWidth": 467.5, "preferredWidthType": "Point", "verticalAlignment": "Center", "isSamePaddingAsTable": true, "borders": {"left": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}}]}, {"title": null, "description": null, "tableFormat": {"allowAutoFit": true, "leftIndent": 0.0, "tableAlignment": "Left", "preferredWidthType": "Auto", "borders": {"left": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}, "bidi": false}}, {"characterFormat": {"bold": true, "fontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0}, "paragraphFormat": {"lineSpacing": 32.0, "lineSpacingType": "Exactly", "styleName": "Normal"}, "inlines": [{"text": "Feedbacks:", "characterFormat": {"bold": true, "fontSize": 14.0, "boldBidi": true, "fontSizeBidi": 14.0}}]}, {"rows": [{"rowFormat": {"allowBreakAcrossPages": true, "isHeader": false, "height": 20.0, "heightType": "AtLeast", "borders": {"left": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}
```



```

yle":false},"diagonalUp":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"
space":0.0,"hasNoneStyle":false}}},"cells":[{"blocks":[{"paragraphFormat":{"s
tyleName":"Normal"},"inlines":[{"editRangeId":"1016946268","columnFirst":0,"c
olumnLast":0,"user":"manager@mycompany.com"},{"text":"Enter the
feedbacks"},{"editRangeId":"1016946268","editableRangeStart":{"editRangeId":"
1016946268","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}}]
},"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredW
idthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"b
orders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.
0,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":
false,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth"
:0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"
None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"verti
cal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneS
tyle":false},"horizontal":{"lineStyle":"None","lineWidth":0.0,"shadow":false,
"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","lineWid
th":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"lines
tyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}
}}}}]},"title":null,"description":null,"tableFormat":{"allowAutoFit":true,"le
ftIndent":0.0,"tableAlignment":"Left","preferredWidthType":"Auto","borders":{"
left":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasN
oneStyle":false},"right":{"lineStyle":"Single","lineWidth":0.5,"shadow":false
,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"Single","lineWidth":0.
5,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"Sin
gle","lineWidth":0.5,"shadow":false,"space":0.0,"hasNoneStyle":false},"vertic
al":{"lineStyle":"Single","lineWidth":0.5,"shadow":false,"space":0.0,"hasNone
Style":false},"horizontal":{"lineStyle":"Single","lineWidth":0.5,"shadow":fal
se,"space":0.0,"hasNoneStyle":false},"diagonalDown":{"lineStyle":"None","line
Width":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"diagonalUp":{"li
neStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":fal
se},"bidi":false},"characterFormat":{"bold":true,"fontSize":14.0,"boldBidi
":true,"fontSizeBidi":14.0},"paragraphFormat":{"lineSpacing":32.0,"lineSpacin
gType":"Exactly","styleName":"Normal"},"inlines":[{"text":"Review
comments","characterFormat":{"bold":true,"fontSize":14.0,"boldBidi":true,"fo
ntSizeBidi":14.0}}]},"rows":[{"rowFormat":{"allowBreakAcrossPages":true,"isH
eader":false,"height":20.0,"heightType":"AtLeast","borders":{"left":{"lineSty
le":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"
right":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNon
eStyle":false},"top":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"spac
e":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"None","lineWidth":0.0,"sh
adow":false,"space":0.0,"hasNoneStyle":false},"vertical":{"lineStyle":"None",
"lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"horizontal"
:{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0,"hasNoneStyle
":false},"diagonalDown":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"s
pace":0.0,"hasNoneStyle":false},"diagonalUp":{"lineStyle":"None","lineWidth":
0.0,"shadow":false,"space":0.0,"hasNoneStyle":false}}},"cells":[{"blocks":[{"
paragraphFormat":{"styleName":"Normal"},"inlines":[{"editRangeId":"1373703080
","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"},{"text":"Ent
er the
comments"},{"editRangeId":"1373703080","editableRangeStart":{"editRangeId":"1
373703080","columnFirst":0,"columnLast":0,"user":"manager@mycompany.com"}}]
},"cellFormat":{"columnSpan":1,"rowSpan":1,"preferredWidth":467.5,"preferredW
idthType":"Point","verticalAlignment":"Center","isSamePaddingAsTable":true,"bo
rders":{"left":{"lineStyle":"None","lineWidth":0.0,"shadow":false,"space":0.0
,"hasNoneStyle":false},"right":{"lineStyle":"None","lineWidth":0.0,"shadow":f
alse,"space":0.0,"hasNoneStyle":false},"top":{"lineStyle":"None","lineWidth":
0.0,"shadow":false,"space":0.0,"hasNoneStyle":false},"bottom":{"lineStyle":"N

```

```

one", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}}}}, {"title": null, "description": null, "tableFormat": {"allowAutoFit": true, "leftIndent": 0.0, "tableAlignment": "Left", "preferredWidthType": "Auto", "borders": {"left": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "right": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "top": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "bottom": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "vertical": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "horizontal": {"lineStyle": "Single", "lineWidth": 0.5, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalDown": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}, "diagonalUp": {"lineStyle": "None", "lineWidth": 0.0, "shadow": false, "space": 0.0, "hasNoneStyle": false}}, "bidi": false}}, {"paragraphFormat": {"styleName": "Normal"}, "inlines": []}], "headersFooters": {"header": {"blocks": [{"paragraphFormat": {"styleName": "Header"}}, {"inlines": [{"text": "Employee's Details"}]}]}, {"sectionFormat": {"headerDistance": 36.0, "footerDistance": 36.0, "pageWidth": 612.0, "pageHeight": 792.0, "leftMargin": 72.0, "rightMargin": 72.0, "topMargin": 72.0, "bottomMargin": 72.0, "differentFirstPage": false, "differentOddAndEvenPages": false, "bidi": false}}, {"characterFormat": {"fontSize": 11.0, "fontFamily": "Calibri", "fontSizeBidi": 11.0, "fontFamilyBidi": "Calibri"}, "paragraphFormat": {"afterSpacing": 8.0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple"}, {"background": {"color": "#FFFFFF"}, "styles": [{"type": "Paragraph", "name": "Normal", "next": "Normal"}, {"type": "Character", "name": "Default Paragraph Font", "next": "Paragraph", "name": "List Paragraph", "basedOn": "Normal", "paragraphFormat": {"leftIndent": 36.0, "contextualSpacing": true}}, {"type": "Paragraph", "name": "Header", "basedOn": "Normal", "next": "Normal", "link": "Header Char", "paragraphFormat": {"afterSpacing": 0.0, "lineSpacing": 1.0, "lineSpacingType": "Multiple", "tabs": [{"tabJustification": "Center", "position": 234.0, "tabLeader": "None", "deletePosition": 0.0}, {"tabJustification": "Right", "position": 468.0, "tabLeader": "None", "deletePosition": 0.0}]}], {"type": "Character", "name": "Header Char", "basedOn": "Default Paragraph Font"}, {"type": "Paragraph", "name": "Footer", "basedOn": "Normal", "link": "Footer Char", "paragraphFormat": {"afterSpacing": 0.0, "lineSpacing": 1.0, "lineSpacingType": "Multiple", "tabs": [{"tabJustification": "Center", "position": 234.0, "tabLeader": "None", "deletePosition": 0.0}, {"tabJustification": "Right", "position": 468.0, "tabLeader": "None", "deletePosition": 0.0}]}], {"type": "Character", "name": "Footer Char", "basedOn": "Default Paragraph Font"}], "defaultTabWidth": 36.0, "formatting": false, "protectionType": "ReadOnly", "enforcement": true, "hashValue": "TQGuJuLceQCe234Ygx4q6NFgHpRMfilhjFTojyKzbQOkwk+ckEM9CjNIdkiUhSR/e/7sfMxO4sbPcg/DBzztMg==", "saltValue": "FXbkr1RtDIIIZfwlM7ldMg=="}`;

setTimeout(() => {
    //Open the document in Document Editor.
    this.container.documentEditor.open(sfdt);
});
this.container.serviceUrl =
'https://ej2services.syncfusion.com/production/web-services/api/documenteditor/';
}
render() {

```



```

        return (
            <DocumentEditorContainerComponent id="container" height={'590px'}
enableToolbar={true} ref={(scope) => { this.container = scope; }} />
        );
    }
}
ReactDOM.render(<Default />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

See Also

- [How to protect the document in form filling mode](#)
- [How to protect the document in comments only mode](#)
- [How to protect the document in track changes only mode](#)

Spell check in React Document editor component

Document Editor supports performing spell checking for any input text. You can perform spell checking for the text in Document Editor and it will provide suggestions for the mis-spelled words through dialog and in context menu. Document editor's spell checker is compatible with [hunspell dictionary files](#).

```
`ts
```

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, DocumentEditor, SpellChecker
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(SpellChecker);
function App() {
  let documentEditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function componentDidMount(): void {
    documentEditor.spellChecker.languageID = 1033 //LCID of "en-us";
    documentEditor.spellChecker.removeUnderline = false;
    documentEditor.spellChecker.allowSpellCheckAndSuggestion = true;
  }
  return (
    <DocumentEditorComponent id="container" ref={{(scope) => { documentEditor = scope; }}} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Features

- Supports context menu suggestions.
- Provides built-in options to Ignore, Ignore All, Change, Change All for error words in spell checker dialog.

Enable SpellCheck

To enable spell check in Document Editor, set [enableSpellCheck](#) property as `true` and then configure `SpellCheckSettings`.

Disable SpellCheck

To disable spell check in Document Editor, set [enableSpellCheck](#) property as `false` or remove [enableSpellCheck](#) property initialization code. The default value of this property is `false`.

Spell check settings

Remove Underline

By default, mis-spelled words are marked with squiggly line. You can also disable this behavior by enabling the [removeUnderline](#) API and now, the squiggly lines will never be rendered for mis-spelled words.

```
`ts
```

```
documentEditor.spellChecker.removeUnderline = false;
```

```
,
```

AllowSpellCheckAndSuggestion

By default, on performing spell check in Document Editor, both spelling and suggestions of the mis-spelled words will be retrieved, and this mis-spelled words can be corrected through context menu suggestions. You can modify this behavior using the [allowSpellCheckAndSuggestion](#) API, which will perform only spell check.

```
`ts
```

```
documentEditor.spellChecker.allowSpellCheckAndSuggestion = false;
```

```
,
```

LanguageID

Document Editor provides multi-language spell check support. You can add as many languages (dictionaries) in the server-side and to use that language for spell checking in Document Editor, it must be matched with [languageID](#) you pass in the Document Editor.

```
`ts
```

```
documentEditor.spellChecker.languageID = 1033; //LCID of "en-us";
```

```
,
```

- Refer to the [Spell checker](#) link for configuring spell checker in server-side.

EnableOptimizedSpellCheck

Document Editor provides option to spellcheck page by page when loading the documents. The default value of this property is `false`, so when opening the document spellcheck web API will be called for each

word in the document. To optimize the frequency of spellcheck web API calls, you can enable this property.

The following code example illustrates how to enable optimized spell checking.

```
`ts
documentEditor.spellChecker.enableOptimizedSpellCheck = true;
`
```

Spell check dictionary cache

Starting from **v20.1.0.xx**, we have optimized the performance and memory usage of spell checker by adding a static method to initialize the dictionaries with specified cache count.

By default, the spell checker holds only one language dictionary in memory. If you want to hold multiple dictionaries in memory, you need to set the cache limit by using **InitializeDictionaries** method as in the below example.

```
`csharp
List<DictionaryData> spellDictCollection = new List<DictionaryData>();
string personalDictPath = string.Empty;
int cacheCount = 2;
// Initialize dictionaries
SpellChecker.InitializeDictionaries(spellDictCollection, personalDictPath, cacheCount);
`
```

If dictionaries are initialized using **InitializeDictionaries** method, then we should use default constructor of the **SpellChecker** to check spelling and get suggestion as in the below example code, it will prevent reinitialization of already loaded dictionaries.

```
`csharp
public string SpellCheck([FromBody] SpellCheckJsonData spellChecker)
{
    try
    {
        SpellChecker spellCheck = new SpellChecker();
        spellCheck.GetSuggestions(spellChecker.LanguageID, spellChecker.TexttoCheck,
            spellChecker.CheckSpelling, spellChecker.CheckSuggestion, spellChecker.AddWord);
        return Newtonsoft.Json.JsonConvert.SerializeObject(spellCheck);
    }
    catch
    {
        return "{\"SpellCollection\":[],\"HasSpellingError\":false,\"Suggestions\":null}";
    }
}
```

```

}
}
`

```

Previously on every `SpellChecker.GetSuggestion()` method call, the `.aff` and dictionary data will be parsed to generate suggestion for miss spelled word. But, starting from `v20.1.0.xx`, the `.aff` and dictionary data will be parsed only for the first time alone while calling `SpellChecker.GetSuggestion()` method.

Add new root word and possible words to dictionary

If you find any root word is missing in the dictionary file, then you can add that new root word and the rule to form the possible words to dictionary file using `AddNewWord` API in the server-side Spell check library.

Note:

1. The rules are framed automatically using the root word, the possible words and affix file.
2. If you pass null for the parameters `affPath` and `possibleWords`, then it will add a single root word to dictionary.
3. This API is included starting from `v20.2.0.xx`.

The following code example demonstrates how to add a new root word to the dictionary along with the rule to form the possible words.

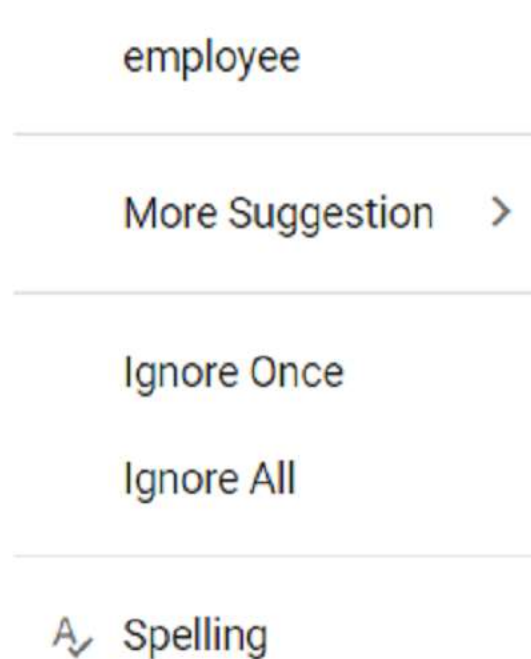
```

`csharp
SpellChecker spellChecker = new SpellChecker();
// Adds the specified new root word to the dictionary along with the rule to form the possible words.
spellChecker.AddNewWord("en.dic","en.aff", "construct", new string[] { "constructs", "reconstruct",
"constructed", "constructive" });
`

```

Context menu

Right click on error word to open the context menu with spell check options. Please see below screenshot for your reference.



Suggestions

Context menu shows the suggestions for mis-spelled words. By clicking on the required word from suggestion, the error word gets replaced automatically.

Add To Dictionary

Using this option, you can add the current word to the dictionary. So that the spell checker does not consider that word as error in future.

Ignore Once and Ignore All

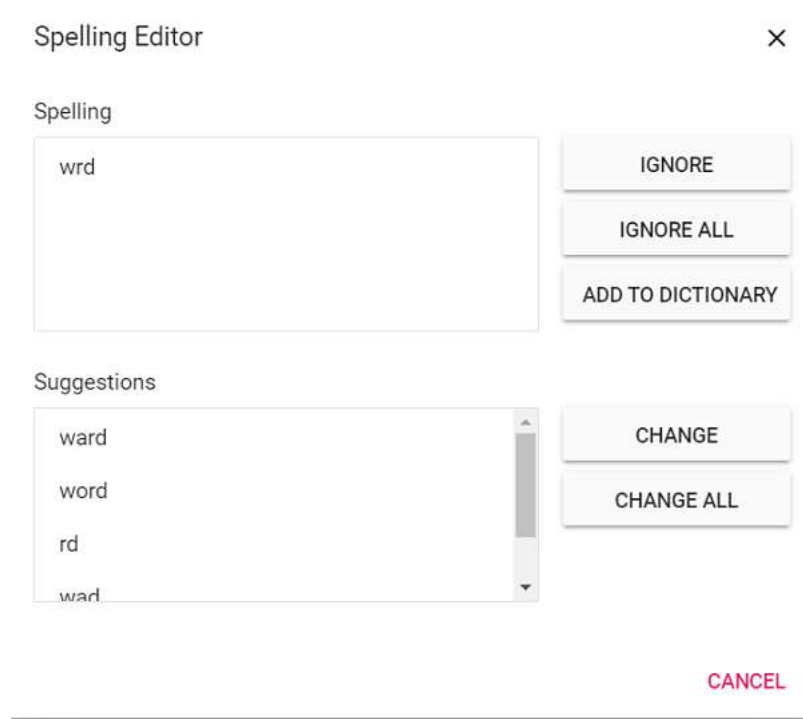
If you do not wish to add the word to dictionary and do not want to show error, use Ignore Once or Ignore All options.

Ignore: ignore only the current occurrence of a word from error.

Ignore All: ignore all occurrence of a word from error in the entire document.

Spelling

Using this option, you can open spell check dialog. Please see below screenshot for your reference.



Global local in React Document editor component

Localization

The **Localization** library allows you to localize default text content of the Document Editor. The document editor component has static text on some features (like find & replace, context-menu, dialogs) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the locale value and translation object. Please refer the sample link [RTL](#)

Note: Please refer the [Locale](#).

Document Editor

The following list of properties and its values are used in the document editor.

Locale keywords | Text

New | New

Open | Open

Undo | Undo

Redo | Redo

Image | Image

Table | Table

Link | Link

Bookmark | Bookmark

Table of Contents | Table of Contents

HEADING - - - - 1 | HEADING - - - - 1

HEADING ---- 2 | HEADING ---- 2

HEADING ---- 3 | HEADING ---- 3

Header | Header

Footer | Footer

Page Setup | Page Setup

Page Number | Page Number

Break | Break

Find | Find

Local Clipboard | Local Clipboard

Restrict Editing | Restrict Editing

Upload from computer | Upload from computer

By URL | By URL

Page Break | Page Break

Section Break | Section Break

Header And Footer | Header & Footer

Options | Options

Levels | Levels

Different First Page | Different First Page

Different header and footer for odd and even pages | Different header and footer for odd and even pages.

Different Odd And Even Pages | Different Odd & Even Pages

Different header and footer for first page | Different header and footer for first page.

Position | Position

Header from Top | Header from Top

Footer from Bottom | Footer from Bottom

Distance from top of the page to top of the header | Distance from top of the page to top of the header.

Distance from bottom of the page to bottom of the footer | Distance from bottom of the page to bottom of the footer.

Aspect ratio | Aspect ratio

W | W

H | H

Width | Width

Height | Height

Text | Text

Paragraph | Paragraph

Fill | Fill

Fill color | Fill color

Border Style | Border Style

Outside borders | Outside borders

All borders | All borders

Inside borders | Inside borders

Left border | Left border

Inside vertical border | Inside vertical border

Right border | Right border

Top border | Top border

Inside horizontal border | Inside horizontal border

Bottom border | Bottom border

Border color | Border color

Border width | Border width

Cell | Cell

Merge cells | Merge cells

Insert Or Delete | Insert / Delete

Insert columns to the left | Insert columns to the left

Insert columns to the right | Insert columns to the right

Insert rows above | Insert rows above

Insert rows below | Insert rows below

Delete rows | Delete rows

Delete columns | Delete columns

Cell Margin | Cell Margin

Top | Top

Bottom | Bottom

Left | Left

Right | Right

Align Text | Align Text

Align top | Align top

Align bottom | Align bottom

Align center | Align center

Number of heading or outline levels to be shown in table of contents | Number of heading or outline levels to be shown in table of contents.

Show page numbers | Show page numbers

Show page numbers in table of contents | Show page numbers in table of contents.

Right align page numbers | Right align page numbers

Right align page numbers in table of contents | Right align page numbers in table of contents.

Use hyperlinks | Use hyperlinks

Use hyperlinks instead of page numbers | Use hyperlinks instead of page numbers.

Font | Font

Font Size | Font Size

Font color | Font color

Text highlight color | Text highlight color

Clear all formatting | Clear all formatting

Bold Tooltip | Bold (Ctrl+B)

Italic Tooltip | Italic (Ctrl+I)

Underline Tooltip | Underline (Ctrl+U)

Strikethrough | Strikethrough

Superscript Tooltip | Superscript (Ctrl+Shift++)

Subscript Tooltip | Subscript (Ctrl+=)

Align left Tooltip | Align left (Ctrl+L)

Center Tooltip | Center (Ctrl+E)

Align right Tooltip | Align right (Ctrl+R)

Justify Tooltip | Justify (Ctrl+J)

Decrease indent | Decrease indent

Increase indent | Increase indent

Line spacing | Line spacing

Bullets | Bullets

Numbering | Numbering

Styles | Styles

Manage Styles | Manage Styles

Page | Page

of | of

Fit one page | Fit one page

Spell Check | Spell Check

Underline errors | Underline errors

Fit page width | Fit page width

Update | Update

Cancel | Cancel

Insert | Insert

No Border | No Border

Create a new document | Create a new document.

Open a document | Open a document.

Undo Tooltip | Undo the last operation (Ctrl+Z).

Redo Tooltip | Redo the last operation (Ctrl+Y).

Insert inline picture from a file | Insert inline picture from a file.

Insert a table into the document | Insert a table into the document

Create Hyperlink | Create a link in your document for quick access to web pages and files (Ctrl+K).

Insert a bookmark in a specific place in this document | Insert a bookmark in a specific place in this document.

Provide an overview of your document by adding a table of contents | Provide an overview of your document by adding a table of contents.

Add or edit the header | Add or edit the header.

Add or edit the footer | Add or edit the footer.

Open the page setup dialog | Open the page setup dialog.

Add page numbers | Add page numbers.

Find Text | Find text in the document (Ctrl+F).

Toggle between the internal clipboard and system clipboard | Toggle between the internal clipboard and system clipboard.

Access to system clipboard through script is denied due to browsers security policy. Instead,

1. You can enable internal clipboard to cut, copy and paste within the component.
2. You can use the keyboard shortcuts (Ctrl+X, Ctrl+C and Ctrl+V) to cut, copy and paste with system clipboard.

Current Page Number | The current page number in the document. Click or tap to navigate specific page.

Read only | Read only

Protections | Protections

Error in establishing connection with web server | Error in establishing connection with web server

Single | Single

Double | Double

New comment | New comment

Comments | Comments

Print layout | Print layout

Web layout | Web layout

Text Form | Text Form

Check Box | Check Box

DropDown | Drop-Down

Update Fields | Update Fields

Update cross reference fields | Update cross reference fields

Hide properties pane | Hide properties pane

Show properties pane | Show properties pane

[Color Picker](#)

The following list of properties and its values are used in the color picker.

Locale keywords |Text

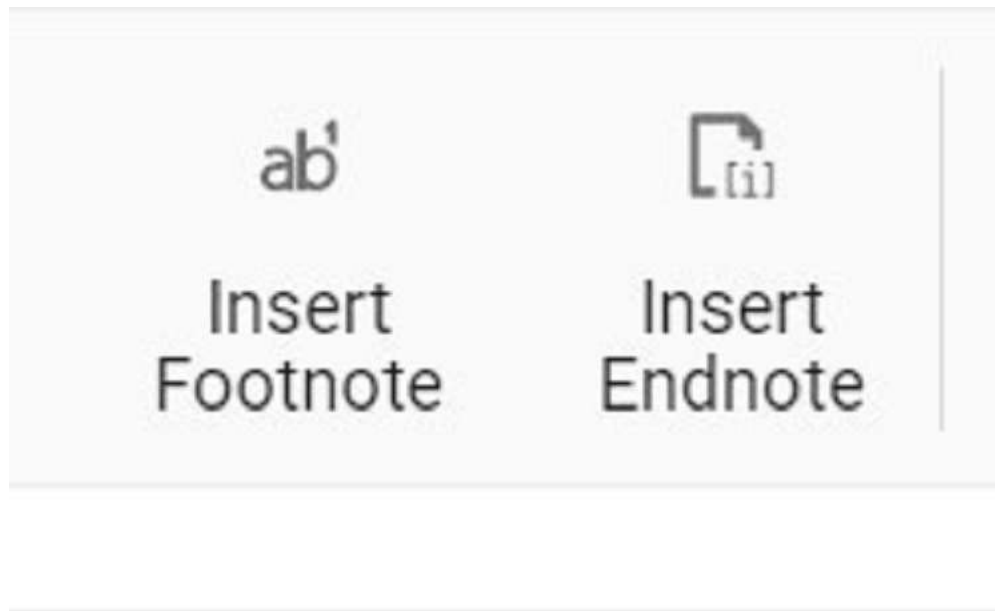
Apply | Apply

Cancel | Cancel

ModeSwitcher | Switch Mode

[Notes in React Document editor component](#)

DocumentEditorContainer component provides support for inserting footnotes and endnotes through the in-built toolbar. Refer to the following screenshot.



The Footnotes and endnotes are both ways of adding extra bits of information to your writing outside of the main text. You can use footnotes and endnotes to add side comments to your work or to place other publications like books, articles, or websites.

Insert footnotes

Document Editor exposes an API to insert footnotes at cursor position programmatically or can be inserted to the end of selected text.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
  ImageResizer, EditorHistory,
  ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog, TableOfContentsDialog,
  PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog, CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require module.
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);
function App() {
  let documenteditor: DocumentEditorComponent=new DocumentEditorComponent(undefined);
  return (
    <div>
      <button onClick={Footnote}>insert Footnote</button>
      <DocumentEditorComponent id="container" height={'330px'} ref={{scope} => { documenteditor =
scope; }} serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" isReadOnly={false} enablePrint={true}
enableSelection={true} enableEditor={true} enableEditorHistory={true}
enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
enableImageResizer={true} enableListDialog={true}
enablePageSetupDialog={true} enableSfdtExport={true}
enableStyleDialog={true} enableTableOfContentsDialog={true}
```

```

enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
enableTextExport={true} enableWordExport={true}/>
</div>
);
function Footnote(){
//Insert footnote.
documenteditor.editor.insertFootnote();
}
}
export default App();
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Insert endnotes

Document Editor exposes an API to insert endnotes at cursor position programmatically or can be inserted to the end of selected text.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory,
ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog, TableOfContentsDialog,
PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog, CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);
let documenteditor: DocumentEditorComponent= new DocumentEditorComponent(undefined);
function App (){
return (
<div>

```

```

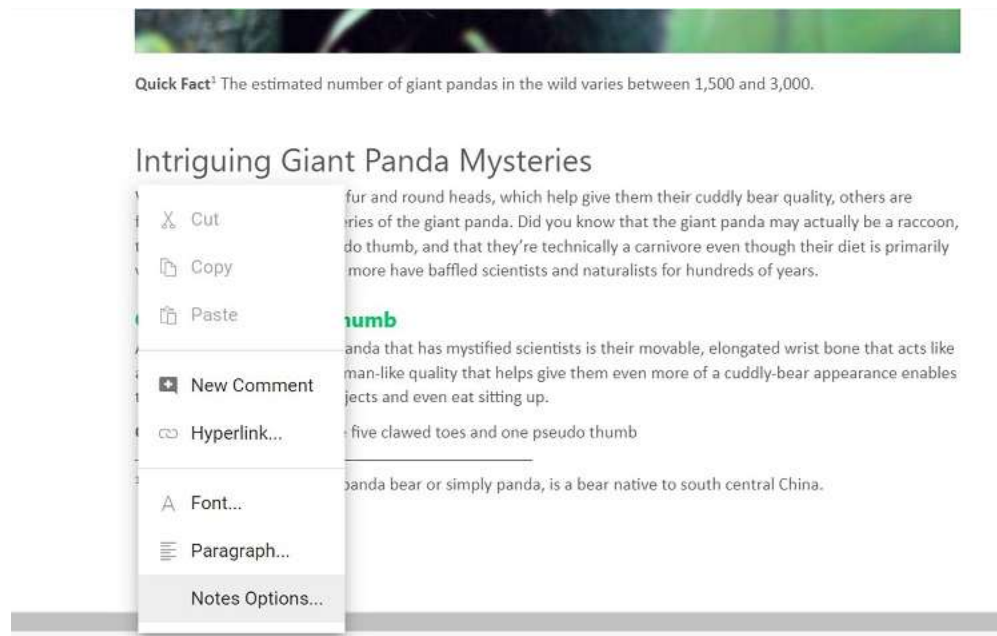
<button onClick={InsertEndnote}>insert Endnote</button>

<DocumentEditorComponent id="container" height={'330px'} ref={{(scope) => { documenteditor =
scope; }} serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" isReadOnly={false} enablePrint={true}
enableSelection={true} enableEditor={true} enableEditorHistory={true}
enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
enableImageResizer={true} enableListDialog={true}
enablePageSetupDialog={true} enableSfdtExport={true}
enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
enableTextExport={true} enableWordExport={true} />
</div>
);
function InsertEndnote() {
//Insert end note.
documenteditor.editor.insertEndnote();
}
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

[Update or edit footnotes and endnotes](#)

You can update or edit the footnotes and endnotes using the built-in context menu shown up by right-clicking it. The footnote endnote dialog box popup and you can customize the number format and start at. Refer to the following screenshot.



View in React Document Editor Component

Web Layout

Document Editor Container component allows you to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#).

```
<DocumentEditorContainerComponent id="container" layoutType={'Continuous'}
enableToolbar={true}/>
```

Note: Default value of [layoutType](#) in Document Editor Container component is [Pages](#).

Ruler

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor.

The following example illustrates how to enable ruler in Document Editor

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
  let container;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function created() {
    container.documentEditorSettings.showRuler = true;
  }
}
```



```

        container.enableAllModules();
    }
    function componentDidMount() {
        setTimeout(() => {
            created();
        });
    }
    function onClick() {
        container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
    }
    return (<div>
        <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
        <DocumentEditorComponent id="container" height={'590px'}
isReadOnly={false} ref={(scope) => {
            container = scope;
            created();
        }}
        serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/"
        />
    </div>

    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
    let container: DocumentEditorComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function created() {
        container.documentEditorSettings.showRuler = true;
        container.enableAllModules();
    }
    function componentDidMount() {
        setTimeout(() => {
            created();
        });
    }
    function onClick() {
        container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
    }
}

```

```

    return ( <div>
      <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
      <DocumentEditorComponent id="container" height={'590px'}
isReadOnly={false} ref={(scope) => {
        container = scope;
        created();
      }}
        serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/"
      />
    </div>
    );
  }
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>

```

```
</head>
<body>
  <div id='root'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>
```

Heading Navigation Pane

Using the heading navigation pane allows users to swiftly navigate documents by heading, enhancing their ability to move through the document efficiently.

The following example demonstrates how to enable the heading navigation pane in a document editor.

`ts

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let settings = {showNavigationPane: true};
  return (
    <DocumentEditorContainerComponent
      id="container"
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      documentEditorSettings={settings}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
```

How To

Override the keyboard shortcuts in React Document editor component

Document Editor triggers the [keyDown](#) event every time when any key is entered and provides an instance of [DocumentEditorKeyDownEventArgs](#). You can use the [isHandled](#) property to override the keyboard shortcut behavior.

Preventing default keyboard shortcut

The following code shows how to prevent the **CTRL + C** keyboard shortcut for copying selected content in document editor.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, DocumentEditorKeyDownEventArgs, SfdtExport, Selection, Editor }
  from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);

function App() {
  let documenteditor: DocumentEditorComponent;

  React.useEffect(() => {
    componentDidMount()
  }, []);

  function componentDidMount() {
    documenteditor.keyDown = function (args: DocumentEditorKeyDownEventArgs) {
      let keyCode: number = args.event.which || args.event.keyCode;

      let isCtrlKey: boolean = (args.event.ctrlKey || args.event.metaKey) ? true : ((keyCode === 17) ? true :
        false);

      //67 is the character code for 'C'
      if (isCtrlKey && keyCode === 67) {
        //To prevent copy operation set isHandled to true
        args.isHandled = true;
      }
    }
  }

  return (
    <DocumentEditorComponent
      id="container"
      ref={scope => {
```

```

documenteditor = scope;
}}
isReadOnly={false}
enableSelection={true}
enableEditor={true}
height={'330px'}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Override or define the keyboard shortcut

Override or define a new keyboard shortcut behavior instead of preventing the keyboard shortcut.

For example, **Ctrl + S** keyboard shortcut saves the document in SFDT format by default, and there is no behavior for **Ctrl + Alt + S**. The following code demonstrates how to override the **Ctrl + S** shortcut to save a document in DOCX format and define **Ctrl + Alt + S** to save the document in SFDT format.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';

import { DocumentEditorComponent, DocumentEditor, SfdtExport, Selection, Editor,
DocumentEditorKeyDownEventArgs } from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(SfdtExport, Selection, Editor);

function App() {
let documentEditor: DocumentEditorComponent;

React.useEffect(() => {
componentDidMount()
}, []);

function componentDidMount() {
documentEditor.keyDown = (args: DocumentEditorKeyDownEventArgs) => {
let keyCode: number = args.event.which || args.event.keyCode;

let isCtrlKey: boolean = (args.event.ctrlKey || args.event.metaKey) ? true : ((keyCode === 17) ? true :
false);

let isAltKey: boolean = args.event.altKey ? args.event.altKey : ((keyCode === 18) ? true : false);

// 83 is the character code for 'S'

```

```

if (isCtrlKey && !isAltKey && keyCode === 83) {
  //To prevent default save operation, set the isHandled property to true
  args.isHandled = true;
  //Download the document in docx format.
  documentEditor.save('sample', 'Docx');
  args.event.preventDefault();
} else if (isCtrlKey && isAltKey && keyCode === 83) {
  //Download the document in sfdt format.
  documentEditor.save('sample', 'Sfdt');
}
}
}
return (
  <DocumentEditorComponent
    id="container"
    ref={scope => {
      documentEditor = scope;
    }}
    isReadOnly={false}
    enableSelection={true}
    enableEditor={true}
    height={'330px'}
  />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Customize context menu in React Document editor component

How to customize context menu in Document Editor

Document Editor allows you to add custom option in context menu. It can be achieved by using the [addCustomMenu\(\)](#) method and custom action is defined using the [customContextMenuSelect](#)

Add Custom Option

The following code shows how to add custom option in context menu.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    onCreate();
  }, []);
  function onCreate() {
    // creating Custom Options
    let menuItems: MenuItemModel[] = [
      {
        text: 'Search In Google',
        id: 'searchingoogle',
        iconCss: 'e-icons e-de-ctnr-find',
      },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems, false);
    // custom Options Select Event
    container.documentEditor.customContextMenuSelect = (
      args: any
    ): void => {
      // custom Options Functionality
      let id: string = container.documentEditor.element.id;
      switch (args.id) {
        case id + 'searchingoogle':
```

```
// To get the selected content as plain text
let searchContent: string =
  container.documentEditor.selection.text;
if (
  !container.documentEditor.selection.isEmpty &&
  /\S/.test(searchContent)
) {
  window.open('http://google.com/search?q=' + searchContent);
}
break;
}
};
}
return (
  <DocumentEditorContainerComponent
    id="container"
    ref={(scope) => {
      container = scope;
    }}
    height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
    enableToolbar={true}
    created={onCreate}
  />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
,
```

[Customize custom option in context menu](#)

Document Editor allows you to customize the added custom option and also to hide/show default context menu.

Hide default context menu items

Using [addCustomMenu\(\)](#) method, you can hide the default context menu. By setting second parameter as true.

The following code shows how to hide default context menu and add custom option in context menu.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    onCreate();
  }, []);
  function onCreate(): void {
    // creating Custom Options
    let menuItems: MenuItemModel[] = [
      {
        text: 'Search In Google',
        id: 'searchingoogle',
        iconCss: 'e-icons e-de-ctnr-find',
      },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems, true);
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={({scope}) => {
```

```

container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
} export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Customize added context menu items

The following code shows how to hide/show added custom option in context menu using the [customContextMenuBeforeOpen](#).

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  React.useEffect(() => {
    onCreate()
  }, []);
  function onCreate() {
    // creating Custom Options
    let menuItems: MenuItemModel[] = [
      {
        text: 'Search In Google',
        id: 'searchingoogle',

```

```

iconCss: 'e-icons e-de-ctnr-find',
},
];
// adding Custom Options
container.documentEditor.contextMenu.addCustomMenu(menuItems, false);
// custom Options Select Event
container.documentEditor.customContextMenuSelect = (args: any): void => {
// custom Options Functionality
let id: string = container.documentEditor.element.id;
switch (args.id) {
case id + 'searchingoogle':
let searchContent: string = container.documentEditor.selection.text;
if (!container.documentEditor.selection.isEmpty && /\S/.test(searchContent)) {
window.open('http://google.com/search?q=' + searchContent);
}
break;
}
};
// custom options hide/show functionality
container.documentEditor.customContextMenuBeforeOpen = (args: any): void => {
let search: any = document.getElementById(args.ids[0]);
search.style.display = 'none';
let searchContent: string = container.documentEditor.selection.text;
if (!container.documentEditor.selection.isEmpty && /\S/.test(searchContent)) {
search.style.display = 'block';
}
};
}
return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
container = scope;

```

```

}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

The following is the output of custom context menu with customization.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar, } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container;
  React.useEffect(() => {
    onCreate();
  }, []);
  function onCreate() {
    // creating Custom Options
    let menuItems = [
      {
        text: 'Search In Google',
        id: 'search_in_google',
        iconCss: 'e-icons e-de-ctnr-find',
      },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems, false);
    // custom Options Select Event
    container.documentEditor.customContextMenuSelect = (args) => {
      // custom Options Functionality
      let id = container.documentEditor.element.id;
      switch (args.id) {
        case id + 'search_in_google':
          let searchContent =
            container.documentEditor.selection.text;
          if (!this.container.documentEditor.selection.isEmpty &&
            /\S/.test(searchContent)) {
            window.open('http://google.com/search?q=' +
              searchContent);

```

```

        }
        break;
    }
};
// custom options hide/show functionality
container.documentEditor.customContextMenuBeforeOpen = (args) => {
    let search = document.getElementById(args.ids[0]);
    search.style.display = 'none';
    let searchContent = container.documentEditor.selection.text;
    if (!container.documentEditor.selection.isEmpty) &&
/\S/.test(searchContent)) {
        search.style.display = 'block';
    }
};
}
return (<DocumentEditorContainerComponent id="container" ref={ (scope) =>
{
    container = scope;
    }} height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true} created={onCreate}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
    DocumentEditorContainerComponent,
    Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
    let container: DocumentEditorContainerComponent;
    React.useEffect(() => {
        onCreate()
    }, []);
    function onCreate() {
        // creating Custom Options
        let menuItems: MenuItemModel[] = [
            {
                text: 'Search In Google',
                id: 'search_in_google',
                iconCss: 'e-icons e-de-ctnr-find',
            },
        ];
    };
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems, false);
    // custom Options Select Event
    container.documentEditor.customContextMenuSelect = (args: any): void => {
        // custom Options Functionality
    }
}

```

```

    let id: string = container.documentEditor.element.id;
    switch (args.id) {
      case id + 'search_in_google':
        let searchContent: string =
container.documentEditor.selection.text;
        if (!this.container.documentEditor.selection.isEmpty &&
/\S/.test(searchContent)) {
          window.open('http://google.com/search?q=' + searchContent);
        }
        break;
    }
  };
  // custom options hide/show functionality
  container.documentEditor.customContextMenuBeforeOpen = (args: any): void
=> {
    let search: any = document.getElementById(args.ids[0]);
    search.style.display = 'none';
    let searchContent: string = container.documentEditor.selection.text;
    if (!container.documentEditor.selection.isEmpty &&
/\S/.test(searchContent)) {
      search.style.display = 'block';
    }
  };
}
return (
  <DocumentEditorContainerComponent
    id="container"
    ref={ (scope) => {
      container = scope;
    }}
    height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/"
    enableToolbar={true}
    created={onCreate}
  />
);
} export default App
ReactDOM.render(<App />, document.getElementById('sample'));
{% enddraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Customize tool bar in React Document editor component

How to customize existing toolbar in DocumentEditorContainer

Document Editor Container allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by [CustomToolbarItemModel](#) and with existing items in [toolbarItems](#) property. Newly added item click action can be defined in [toolbarclick](#).
- Show, Hide - Existing items can be shown or hidden using the [toolbarItems](#) property. Pre-defined toolbar items are available with [ToolbarItem](#).
- Enable, Disable - Toolbar items can be enabled or disable using [enableItems](#)

```
{% raw %}
```

```
`ts
```

```
import './App.css';
```

```
import * as React from "react";
```

```
import { DocumentEditorContainerComponent, Toolbar, CustomToolbarItemModel } from
"@syncfusion/ej2-react-documenteditor";
```

```
DocumentEditorContainerComponent.Inject(Toolbar);
```

```
export default class App extends React.Component {
  public container: DocumentEditorContainerComponent;
  render() {
    //Custom toolbar item.
    let toolItem: CustomToolbarItemModel = {
      prefixIcon: "e-de-ctnr-lock",
      tooltipText: "Disable Image",
      text: "Disable Image",
      id: "Custom"
    };
    let items = [
      toolItem,
      "Undo",
      "Redo",
      "Separator",
      "Image",
      "Table",
      "Hyperlink",
      "Bookmark",
      "Comments",
      "TableOfContents",
      "Separator",
      "Header",
      "Footer",
      "PageSetup",
      "PageNumber",
      "Break",
      "Separator",
      "Find",
      "Separator",
      "LocalClipboard",
      "RestrictEditing"
    ];
```



```

return (
<DocumentEditorContainerComponent
ref={scope => {
this.container = scope;
}}
id="container"
style={{ height: "590px" }}
toolbarItems={items}
toolbarClick={this.onToolbarClick.bind(this)}
enableToolbar={true}
/>
);
}

onToolbarClick = (args: ClickEventArgs): void => {
switch (args.item.id) {
case "Custom":
//Disable image toolbar item.
this.container.toolbar.enableItems(4, false);
break;
default:
break;
}
};
}
,

{% endraw %}

```

Note: Default value of `toolbarItems` is ['New', 'Open', 'Separator', 'Undo', 'Redo', 'Separator', 'Image', 'Table', 'Hyperlink', 'Bookmark', 'TableOfContents', 'Separator', 'Header', 'Footer', 'PageSetup', 'PageNumber', 'Break', 'InsertFootnote', 'InsertEndnote', 'Separator', 'Find', 'Separator', 'Comments', 'TrackChanges', 'Separator', 'LocalClipboard', 'RestrictEditing', 'Separator', 'FormFields', 'UpdateFields'].

Change document view in React Document editor component

[How to change the document view in DocumentEditor component](#)

Document Editor allows you to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#).

```
<DocumentEditorComponent id="container" layoutType={'Continuous'} />
```

Note: Default value of [layoutType](#) in Document Editor component is [Pages](#).

How to change the document view in DocumentEditorContainer component

Document Editor Container component allows you to change the view to web layout and print using the [layoutType](#) property with the supported [LayoutType](#).

```
<DocumentEditorContainerComponent id="container" layoutType={'Continuous'}
enableToolbar={true}/>
```

Note: Default value of [layoutType](#) in Document Editor Container component is [Pages](#).

Open default document in React Document editor component

In this article, we are going to see how to open a default document when Document Editor & Document Editor Container is initialized.

Opening a default document in DocumentEditor

By default, Document Editor will open blank document. You can use [open](#) API in Document Editor to open the sfdt content.

Document editor have [created](#) event which gets triggered once Document editor control created. So, if you want to open the document by default, you can use [open](#) and [created](#) API.

The following example illustrates how to open the default SFDT content once Document editor control gets created.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
  let container;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function created() {
    // load your default document here
    let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":
72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"te
xt":"Adventure Works
Cycles"}]}],"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFo
```

```

rmat":{}}, "characterFormat":{}, "inlines":[]]]}, "footer":{"blocks":[{"paragrap
hFormat":{"listFormat":{}}, "characterFormat":{}, "inlines":[]]]}}}, "character
Format":{"bold":false, "italic":false, "fontSize":11, "fontFamily":"Calibri", "un
derline":"None", "strikethrough":"None", "baselineAlignment":"Normal", "highligh
tColor":"NoColor", "fontColor":"empty", "fontSizeBidi":11, "fontFamilyBidi":"Cal
ibri", "allCaps":false}, "paragraphFormat":{"leftIndent":0, "rightIndent":0, "fir
stLineIndent":0, "textAlignment":"Left", "beforeSpacing":0, "afterSpacing":0, "li
neSpacing":1.0791666507720947, "lineSpacingType":"Multiple", "listFormat":{}}, "b
idi":false}, "defaultTabWidth":36, "trackChanges":false, "enforcement":false, "ha
shValue":"","saltValue":"","formatting":false, "protectionType":"NoProtection"
, "dontUseHTMLParagraphAutoSpacing":false, "formFieldShading":true, "styles":[{"
name":"Normal", "type":"Paragraph", "paragraphFormat":{"lineSpacing":1.14999997
6158142, "lineSpacingType":"Multiple", "listFormat":{}}, "characterFormat":{"fon
tFamily":"Calibri"}, "next":"Normal"}, {"name":"Default Paragraph
Font", "type":"Character", "characterFormat":{}}, {"name":"Heading 1
Char", "type":"Character", "characterFormat":{"fontSize":16, "fontFamily":"Calib
ri Light", "fontColor":"#2F5496"}, "basedOn":"Default Paragraph
Font"}, {"name":"Heading
1", "type":"Paragraph", "paragraphFormat":{"beforeSpacing":12, "afterSpacing":0,
"outlineLevel":"Level1", "listFormat":{}}, "characterFormat":{"fontSize":16, "fo
ntFamily":"Calibri
Light", "fontColor":"#2F5496"}, "basedOn":"Normal", "link":"Heading 1
Char", "next":"Normal"}, {"name":"Heading 2
Char", "type":"Character", "characterFormat":{"fontSize":13, "fontFamily":"Calib
ri Light", "fontColor":"#2F5496"}, "basedOn":"Default Paragraph
Font"}, {"name":"Heading
2", "type":"Paragraph", "paragraphFormat":{"beforeSpacing":2, "afterSpacing":6, "
outlineLevel":"Level2", "listFormat":{}}, "characterFormat":{"fontSize":13, "fon
tFamily":"Calibri
Light", "fontColor":"#2F5496"}, "basedOn":"Normal", "link":"Heading 2
Char", "next":"Normal"}, {"name":"Heading
3", "type":"Paragraph", "paragraphFormat":{"leftIndent":0, "rightIndent":0, "firs
tLineIndent":0, "textAlignment":"Left", "beforeSpacing":2, "afterSpacing":0, "lin
eSpacing":1.0791666507720947, "lineSpacingType":"Multiple", "outlineLevel":"Lev
el3", "listFormat":{}}, "characterFormat":{"fontSize":12, "fontFamily":"Calibri
Light", "fontColor":"#1F3763"}, "basedOn":"Normal", "link":"Heading 3
Char", "next":"Normal"}, {"name":"Heading 3
Char", "type":"Character", "characterFormat":{"fontSize":12, "fontFamily":"Calib
ri Light", "fontColor":"#1F3763"}, "basedOn":"Default Paragraph
Font"}, {"name":"Heading
4", "type":"Paragraph", "paragraphFormat":{"leftIndent":0, "rightIndent":0, "firs
tLineIndent":0, "textAlignment":"Left", "beforeSpacing":2, "afterSpacing":0, "lin
eSpacing":1.0791666507720947, "lineSpacingType":"Multiple", "outlineLevel":"Lev
el4", "listFormat":{}}, "characterFormat":{"italic":true, "fontFamily":"Calibri
Light", "fontColor":"#2F5496"}, "basedOn":"Normal", "link":"Heading 4
Char", "next":"Normal"}, {"name":"Heading 4
Char", "type":"Character", "characterFormat":{"italic":true, "fontFamily":"Calib
ri Light", "fontColor":"#2F5496"}, "basedOn":"Default Paragraph
Font"}, {"name":"Heading
5", "type":"Paragraph", "paragraphFormat":{"leftIndent":0, "rightIndent":0, "firs
tLineIndent":0, "textAlignment":"Left", "beforeSpacing":2, "afterSpacing":0, "lin
eSpacing":1.0791666507720947, "lineSpacingType":"Multiple", "outlineLevel":"Lev
el5", "listFormat":{}}, "characterFormat":{"fontFamily":"Calibri
Light", "fontColor":"#2F5496"}, "basedOn":"Normal", "link":"Heading 5
Char", "next":"Normal"}, {"name":"Heading 5
Char", "type":"Character", "characterFormat":{"fontFamily":"Calibri
Light", "fontColor":"#2F5496"}, "basedOn":"Default Paragraph

```

```
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal"}, {"name": "Heading 6 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml": []}`;

    // Open the default document
    container.open(data);
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }
  return (<DocumentEditorComponent id="container" height={'590px'}
ref={ (scope) => {
    container = scope;
    created();
  }} serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/" />);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
  let container: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function created() {
    // load your default document here
    let data =
`{"sections": [{"sectionFormat": {"pageWidth": 612, "pageHeight": 792, "leftMargin": 72, "rightMargin": 72, "topMargin": 72, "bottomMargin": 72, "differentFirstPage": false, "differentOddAndEvenPages": false, "headerDistance": 36, "footerDistance": 36, "bidi": false}, "blocks": [{"paragraphFormat": {"afterSpacing": 30, "styleName": "Heading 1", "listFormat": {}}, "characterFormat": {}, "inlines": [{"characterFormat": {}, "text": "Adventure Works Cycles"}]}], "headersFooters": {"header": {"blocks": [{"paragraphFormat": {"listFormat": {}}, "characterFormat": {}, "inlines": []]}}, "footer": {"blocks": [{"paragraphFormat": {"listFormat": {}}, "characterFormat": {}, "inlines": []]}]}}, "character`
```

```

Format":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph Font","type":"Character","characterFormat":{}},{name":"Heading 1 Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"},{"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"},{"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"},{"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"},{"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4 Char","next":"Normal"},{"name":"Heading 4 Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level5","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5 Char","next":"Normal"},{"name":"Heading 5 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"first

```

```

tLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level6","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 6 Char","next":"Normal"},{"name":"Heading 6 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"}], "lists":[], "abstractLists":[], "comments":[], "revisions":[], "customXml":[]}]`;

    // Open the default document
    container.open(data);
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }
  return (<DocumentEditorComponent id="container" height={'590px'}
ref={ (scope) => {
  container = scope;
  created();
}}
  serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
  />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-inputs/styles/fabric.css" rel="stylesheet" />

```

```
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>
<style>
    /** Document editor sample level font icons*/
@font-face {
    font-family: 'Sample brower icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSi8AAAEoAAAAVmNtYXRDQ0OvsAAACfAAAAALBnbHlma57A
AgAAA6wAAC90aGVhZBF0mKkAADQAAAAANmhoZWEIUAIAAAAAAArAAAACRobXR4/AAAAAAYAAAAAD8b
G9jYXhXbWwAAAMsAAAAAGlHeHABZQE/AAABCAAAACBuYWllChIPawAAMyAAAAAL9cG9zdAI/4kAAD
YgAAADOQABAAAEAAAAAFwEAAAAAAD8wABAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIA
AAACgAAAP8AAAAAIAAAQAAZAAABQAAOkCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMMAAAAAAIAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnPQQAAAAAXAQAAAAAAAAABAAAAAAAAAAAAAAAAQAAAAEAAAABA
AAAAQAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAA
EAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAA
AAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAAB
AAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAA
AAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAAB
AAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAA
AAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAAABAAAAAIAAAAAEAAA
ABAAAAIAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAEgATABQAFQAWABcAGAAZABoAGwAc
AB0AHgAfACAAIQAIACMAJAALACYAJwAoACkAKgArACwALQAUAC8AMAAxADIAMwA0ADUANgA3ADgAO
QA6ADsAPAA9AD4AAAAAAAFoAngDuAg4CWAJ4ApoCxcgMGA9QD8gVgBcoGSgAMByoHYggKCLII3AkICb
wJ3An4CjIKvAr4C8QL4AwADEIM6g0MDawNxxg42DoIOpA8yD2YPhA+2EFgQdhEWEcAR2BI4EyYTXhO
UE8AUPhRWFJAUnhVAFegWMBdiF4IXugAOAAAAAAPzA7UAAwAHAAsADwATABcAGwAfACMAJwArAC8A
MwA3AAALmZUjBzM1IwczNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUhNSEFITUhJTM1IwUzNSMHMzUjB
zM1IwO1Pz+7fX36fX36fX36fX0C7vr6/on6+v6J+voB9AH0/gz+DAF3/okC7vr6/si7u/p9fbw/P0
t9fX19fX19fX19fX19fX19fHx8fX19fX19fX0AAAAACAAAAAAN2A/MABAARAAABEwkBEQMfCTM/BAk
BHwYzPwkRIQM4Af7H/sg/AQIDBQYGCakJCQkJCQkIBwEKAQsFBQUGBgYGDawFCQgGBgUDAgH9EgO1
/JUBZ/6aA2r8lgoJCAgHBwUEAwEBawQFBwEx/s4FBAMDAgEBAGIEBQYICAgJCgOpAAAAABQAAAAAD8
wPUAAQACAAAC4AMgAAJRujJzcHITU3JQ8DHQEfBj8GNS8GDwETESchAQcRAYERIQ01j51SDf192g
HdAgICAgICBAUGBgYGBgYFBAMBAQMCBwUGCwkFrtp9/sfaPwPo/Bh9E5xR7c7bQgIDBQYHBgYFBQQ
DAQEBAQMEBAUGCwoFagYDAQECAwE9/UDzfQE42gIG/JYDqAAAAAIAAAAAA/MD8wB/AQUAAAEVDx0r
AS8dPQE/HTsBHx0FFR8HAQ8DHwgZPwQBHwc/Hy8fDx4DtQECAwMFBQUGBwgICQkKCgsLDAwNDQ4OD
g4PDw8QEBAQE8QDw8ODg4ODQwNDAsLCgoJCQgIBGcFBQQEAWIBAQIDBAQFBQYHCAGJCQoKCwsMDA
0NDg4ODg8PDxQEBAQDxAPDw4ODg4NDA0MCwsKCgkJCAgGBwUFBAQDAgH9UQEEBgKDA4P/s8GBQI
BAGMGCAQFBgsMDAwLBQUFAS0YGBobHB0dHhMTExITERIREBAQDw8ODg0MDAsLCQkJBwcGBQQDAwEB
AQEDAwQFBGcHCQkJCwsMDA0ODg8PEBAQERIRExITExMTExMSExESERARDw8PDg4NDAwLCwoJCAcHB
gUEBAIBAn0QEBAPDw8ODw0ODQ0MDAsLCgoJCQgIBwYFBQUDAwIBAQIDAwUFBQYHCAGJCQoKCwsMDA
0NDg0PDg8PDxQEBAQDxAPDw4ODg4NDA0LDAsKCgkJCAcHBwUFBAQDAgEBAGMEBAUFBwcHCAkJCgo
```

LDAsNDA00Dg4ODw8QDxAQDw8dHRwbGhgY/s4KCgsLCwsKCQUDBAQCAGQEAWUBLRAODAoHBgQBAQED
AwQFBGcHCAoJCwsMDA00Dg8PDxEQERIREXITExMTEXMSExESERAQEA8PDg4NDawLCwkJCQcHBgUEA
wMBAQEBAwMEBQYHBwkJCQsLDAwNDg4PDxAQEBESEhISExMAAAsAAAAA9QD1AADAACACwAPABMAFw
AbAB8AIwApAC8AACUzNSM1MzUjNzM1IwczNSMHMzUjBzM1IwczNSM3MzUjNTM1IycZIREhESMRFS
RIQHhPj4+Pvo+Pn0+Pn0+Pn0+Pn0+Pvo+Pj4++j4CcfzUPgOo/FjnPj8+Pz4+Pj4+Pj4+Pj8+Pz59
/NQDLpZUPgOoAAAAEAAAAAPzA/MAAwAHAAAsADwAANYE1ITUhNSE1ITUhNSE1IQwD6PwYA+j8GAPO/
BgD6PwYDD/6Pvo++j8AAAAAQAAAAADTQ01AAsAABMJARcJATcJAScJAUsBif53LAGJAYks/ncBiS
z+d/53A4n+d/53LAGJ/ncsAYkBiSz+dwGJAAAFAAAAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISU
XNyc3JxchNSE1ITUhDAPo/BgBOQKv/VH+x5IqaWkqpWkv/VH+xwPo/BgMP/o+fZwscHAsHz76PwAA
BwAAAAAD8wPzAAMABwATABcAGwAfACsAACUzNSMHMzUjNyMVMxUzNTM1IzUjJSE1ISUzNSMHMzUjF
yMVMxUzNTM1IzUjAn0+Pvo/P30+Pj8+Pj/+DAPo/BgCcT4++j8/ft4+Pz4+P8g+Pj4/P/r6Pz59Pr
w+Pj4+Pz4+P/oAAAAEAAAAAPzA/MAMAAzAGkApwAAJRUPDi8OPQE/Bx8GAQcnBQ8JFR8OPw81Lwk
BFQkCJwcXByEBNT8GOWEfBhEzETUvDg8OA6sBAGMDAwUEBgUGBwYHBwgHBwCHBgYGBwBAMCAGEB
AgYJChINDRsMCwkIBAL+pOriAsMBNBUJCggHBQMBawMFBgcJCQsLDA0NDg4PDw8ODQ0MCwoKCAcGB
QQCAQMEBgwJCgoVEzT94/7HAVgBloUwYBX98QECAQIDAUFwBgcGBgUFAwMCAT4CAGMEBQUGBwCICA
kJCQkKCQkCJAGHBwYFBQQDAgKuCQkICAGHBwCfBQUEAwIBAQBAGMEBQUFBwCHBwkICQkHCQgTFRU
fFRQpFRUVExiJAQ3i4iMCSCQSExQTExmRERAPDw4ODAsLCQgHBQQDAQEDBAUHCakLCwwODg8PEAgR
ExMTHRMTEiAcQgHUcP67/qgBh6AodBQBDIOGBgUFBAMCAGMEBQUGBv7nArkKCQkCJACIBgYGBAQDA
wEBAQEDAwQEBgYGCACJCAkJAAAAgAAAAAD8wPzAAMADAAANYE1ISUnBwkBJwCRiwwD6PwYafTkLA
EvAS8s4z8MP+blLP7OATIs5QLDAAAABgAAAAAD8wPzAB8AXwCfAOIA5QEYAAABFQ8FKwEvBj8GOWE
fBQcVHw4/Dy8OIw8OFw8PLw8/Dx8OJyMPAYcHFW8EJwcfBACXNx8DBxc3HwE/Ahc3Jz8DFzcnPwUn
By8DNycHLwM1IycjNSURHw8hNSEjLwU1ETU/BTMhFTMVMz0BLw8hDw4DEgICAwQEBAUFwBQDAwMBA
QEBAwMDBAUFBQQEBAwMCAm8CAGMDBQUFBwYHCAgICQkCJACIBwCHBgYFBAQDAgEBAQECAwQEBQYGBw
cHCAkICQkCJAGIBwYHBQUFAwMCAT4BAGMFBQcICQkLCwwMDQ0ODg4MDQwLCgoJBwCGBQMCAQECAwU
GBwCJCgoLDA0MDg4ODQ0MDAsLCQkIBwUFAwKiAhQTEhIiKiIJCwoIBDMKNAEDBQYvHDAODg8TFDQU
FBQPDwkUNBQSDw0QMBwvBQUEAQE0CjMICAoQIioiFRESFTgQkP3OAQECBAQEBgYGCACICQkCJCGW/
moGBgYEBAMCAGMEBQUGBwGw+j4BAwMEBAYGlgYICAgJCAoJ/mUKCQkCJACIBgYGBAQEAQEBBgUEBA
QDAgICAgMEBAQFBQUEAwMDAQEDAwMEBQUJCAkIBwCHBgYFBAQDAgEBAQECAwQEBQYGBwCHCAkICQk
CJCAgHCAYGBgUEBAMCAGEBAGMEBAUGBgYIBwgICQkODQ0MDAsLCQkIBwUFAwIBAQIDBQUHCakJCwsM
DA0NDg4NDQ0MCwoKCQcHBgQEAgEBAGQEBGcHCQoKCwwNDQ22BAYICikkKQoQERILCTcKGBQTEHsxH
A4NCww3FDgDAQECATgTOAoLDBECMBwNERMTDQk4CRQQBQpJCKLBwYENvqPdfzUCgkCJCAkHCAQEBG
QEAWMBAT8CAwQFBQYGAyWGBgUFBAMC+nyCCQkCJQgIBwFVBwUFBAMCAQEBAQIEBAQGBgYIBwgJCQk
AAAAABAAAAAADdgPzAAMABwAiAFMAADchNSEBFQc1AQ8KHQEHnzUvCSM7AR8PBzMVNZuzJz8PMzUj
FSE1I4kC7v0SABZ+ATIGBgOIBwUFAwMCAf6JAQIBAwQEBQcICgyECgoSEQ4MDAoIBwCFawMDAQEBA
m76bQIBAQICAwQFBggICgsNDhESFD/9kD8MFQF3UESUATgGBg0NDg4ODg8PDxBfYA8PDw4PDg4NDg
0MAwQFBwgJCgsLDQ4ODhAPIH76jW1+IA8QDg4ODQsLCgkIBwUEA7x9fQACAAAAAAPzA7UAVABgAAA
BDwUVPwY7AR8JFQ8QFTM1Iz8SLw8HBQkBFwkBNwkBJwkBA1cODg0MDQwMDAwMDQwNDACNDAAJBAMD
AgEBAGQGBwKRDDcODAsKAYCAGH6tAEBAGQECwxAGQ8MBQQEBAICAQEBAgIEBQUHBwgJCgoMDAwNE
PylATH+zzIBJgEmMf7QATAx/tr+2gOzAwMFBgcIOQoJBwYEBaICAUAHBQGBGQcGDgWMCwoKDgorCw
wMDQ4PCAgIJTMHBQYFBQsLMBUPDwgICakJCgoLDAsLCgkICAcGBQQEAWIBAQEm/nH+cCYBgV5/JQG
QAY8m/n4BggAACgAAAAAD8wPzAAMABwALAA8AEwAXABsAHwAjACgAAAEVizUjFSM1IxUjNQEVIZUj
FSM1IxUjNQEVIZUjFSM1IxUjNQMPAREhA7X6Pvo++gNq+j76PvoDavo++j76PwE5Aq/8GAFF+vr6+
vr6ATj6+vr6+voBOPr6+vr6+vxXA+gAAAAAQAAAAAD8wPzAIoAABMBNwEhMx8dHQEPHSsBFTM/Hy
8eIyEBJwBjSn+ygIQDw4ODg0ODQwNDawLCwsKCgkCJAGHBwYGBQUdAwMCAQECAwMDBQUGBGcHCAg
JCQoKCwsLDAwNDA00DQ4ODg9eXhIREREREBAQDw8ODg4NDawLCwoKCQgIBwYFBQQDAgEBAQECAwQF
BQYHCAgJCgoLCwwMDQ4ODg8PEBAQERERERL99wEtKQKY/q0vAQkCAQMDBAQFBgYHBwgICQoJCgsLD
AsMDQ0NDQ4NDg8ODw4ODg0ODQ0MDAwLCwsKCgkCJAGIBGcFBQUDBAICAT8BAQIDBAUFBgICakKCg
sLDAwNDg4ODw8QEBAREREREhIREREREBAQDw8ODg0NDQwLcwoKCQgHBwCfBQMDAwEBcI8AAAUAAAA
AA/MD8wALAA8AEwAXACcAACUjFTMVMzUzNSM1IxEVIZUjFSM1IxUjNQMHESMVIzUjFSM1IxUjNSMC
AH19P3x8PwG1+j76Pvo/A+g/+j76Pvo/yD99fT99AXb6+vr6+vr+yAJx+vr6+vr6AAAAAAPUA
9QAawAHAAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPafMAVwBbAF8AYwBnAGsAbw
AAJTM1IwczNSMHMzUjBzM1IwczNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSM
hMzUjBzM1IwczNSMFMzUjBzM1IwczNSMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjNSE1IQOW
Pj59Pz99Pz+7Pj68Pz99Pz98Pj4Daj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4BtT4+AbU+Pn0/P30/P
/6JPz99Pz98Pj4BtT4+AbU+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+A6j8WCw+Pj4+Pj4+Pj4+Pj4+Pj
8/Pz8/Pj8/Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+PgAFAAAAAAOWA/MAAwAfACIAQAC
FAAABByM3JyMVMwcjFTMHFzcZBxc3MzUjNzM1IzcnByM3JyUjNScVMxEPBiMhIy8GET8GMwCfRFR8O

IT8ONRE1Lw8hDw4CRxJ8EjZwZxJVTA0+DnwMPQ5vZhJVTA0+DnwMPQGIjz76AQIDBAQGBQf9kAcFB
gQEAWIBAQIDBAQGBQdeAgIDBAUFBgCHAgJCQkKAnAKCQkJCAgHBWYFBQQDAgICAgMEBQUG1gcHCA
gJCQkJ/mUKCQkJCAgHBWYFBQQDAgIBwn19Pj59P1kJY1kJYj59P1kJY1kJmI8s+v2vBgYFBQQDAgI
DBAUFBgYDLAYGBQUEAwIf/NQKCQkICQcIBgYGBAQDAWEBABQEDAwQEBgYGCACJCAkJCgJXCQkJCQgI
BwfVBgYFBAMCAQEBAQMDBAQGBgYIBwkICQkAAAAADAAAAAAPzA/MACAAMABUAACUXNxEzERc3JyUhN
SE1JwcXNycHESMBGypTP1Mqnf3tA+j8GAH0UyqcnCpTPvYvTP75AQdML419Pq9ML42NL0wBBwAFAA
AAAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISUXBxc3JwUhNSE1ITUhDAPo/BgBOQKv/VH+x29vLJu
bAQ0Cr/1R/scD6PwYDD/6PuxvbyybmX4++j8AAwAAAAADGQO1ACMARgCbAAABOWeFDg8OKwEREx8P
Dw8jEQcVESE/GzUvDzU/DzUvECEBzQ0NGRgVFBIQDw0LCQgGBQIBAQIEBgJCcWwODhERExUVF5F7F
RQSERAOdQwKCQgGBQMCAQECBAYHCAsLDg4PERITFBZtawEKHx4dDg0NDQwMDAsLCwoKCQgHBWYGBQ
QEAWICAQECEBQYICQsNDw8REhMUFhYSERAPdG0MCwoIBWYFAWIBAWQGBAUFBg0PERMVfHcZGxz+7gH
iAgMEBgCHCQsLBDQ4PEBITEhEQDw4NDQsKCAgGBAQCAToBdwEBAwMFBQcHCQkLCwWODhASEQ8PDg0L
CwoIBWYFAWIBAWDUP/3OAMGAWQFBQYGBWcICAKJCgoKCgsLDawMDQwODQ4WFRQTEhAQDw0MCgoHB
gUDAwYHCQkKCw0NDg8PEBAREhILFRUTCQkICRAPDQ0KCQcFAwIAAAAAABAAAAAD8wPzAAMABwALAA
8AADchNSE1ITUhNSE1ITUhNSEMAq/9UQPo/BgCr/1RA+j8GAw/+j76Pvo/AAAAAAMAAAAA7UD8wA
DAACACwAANYe1IQERIREDIRehyAJw/ZACr/0SPgNq/Ja9vAI8/JYDavaxA+gABQAAAAAD8wPzAAMA
BwATABcAJwAAARUjNRMVIZUFIxUzFTM1MzUjNSMnFSM1ITMVIxUzFSMVMxUjFSERIQI/+vr6AfN9f
T99fT/5+v7H+vr6+vr6AnH9jwFF+voBOPr6Pz59fT59+vr6+j76Pvo/A+gAAAAACAAAAAN2A/MAAw
B4AAA3ITUhExUfhj8eNREjEQcVDxQrAS8UNQMjiQLu/RI/AQIDAwQFBgYHCAGJCQoKCwsMDA0NDQ4
PDg8PDxAQEBAQEAS8PDw4PDg0NDQwMCwsKCgkJCAgHBgYFBAMDAgE+AQICAwMEBQUMDQ8RExMWFgW
DAwNDA0NDA0MDAwMCwsWEXMRDw0MCgQDAWICAT4MPwF3EQ8QDw8PDw4ODg0MDQsMCwoKCQgJBWcGB
gUEBAICAQEBAQICBAQFBgYHBwkICQoKCwWLDQwNDg4ODw8PDxAPEQIy/c4NDQwNDAsMDAsVFBIRDw
4LCgQEAgMBAQEBAWIEBAQGCW4PERIUFRcMCwWMDA0CPwAFAAAAAAPzA/MAAwAHABMAFWAoAAABFSM
1ExUjNQUjFTMVMzUzNSM1IyUVIZUDKQE1IzUzNSM1MzUjNTM1IQK7+fn5/sd9fT98fD8CMvk/ATgB
Ofr6+vr6+v2PAUT5+QE5+vo/Pn19Pn36+vr8Vz/6Pvo++j8AAAAADAAAAAN2A/MAJQBIAK8AAAEhO
wEfBRURFQ8FIyEjLwU1ETU/BTM1FSM1Pw47AR8NBRUjDw8RHw8hPw8RLw8jNS8PDw4BRQF2XgYGBg
QEAWICAwQFBQYG/c4GBgYEBAMCAGMEBQUGBgGW+gECAwQFBggICQkLCgWMDA0NDAwMCgsJCQgIBgU
EAwL+yV4KCQkJCACIBgYGBAQEAgEBAQECEBAQEAgYGCACICQkJCgIyCgkJCQgHCAYGBgQEBAIBAQE
AgQEBAyGBggHCAkJCQpEaQIFBgkCg0NDhAQERITExMTEhEQEA4NDQoKCAyFAgI+AgMEBAYFB/5LB
gYFBQQDAgIDBAUFBgYBtQcFBgQEAWL6u7sNDawMCwoKCQgHBgYFAWICAwUFBgICQoKCwWMDA27AQ
ECBAMFBgYGBWgICQkJCv5LCgkJCQgHCAYGBgQEBAIBAQAQEBAyGBggHCAkJCQoBtQoJCQkICAC
BgWUFBAMCAQG7EXMSEREpdG4MCwkIBgUDAQEDBQYICQsMDg4PERESEwADAAAAA01A/MAAwAHAAAsA
ABMhNSE1ESERAYERICgCcP2QAq/9Ej4DavyWAoe8cvyWA2r8VwPoAAMAAAAA5YDtQADAAcADwAAJ
TMRIyUhNSERIREzESE1IQHhPj7+iQMs/NQBdz4Bd/zUSwE4Pz4Bd/7HATk+AAADAAAAAAPzA7UADA
AQACcAACUHIY8DPQE/AyUJAw8HHwghNQUJAQIUP9GyAwICAgIDlQK0/qX+1AFb/bYGBQQDAWIBAQE
BAGMDBAUGxQMK/joBxv57xD2tAwQEQUEBASRWP6xASEBUP4fBgYHCAGICAgICAgIBWcGBr8+AgG3
AXcAAAAcAAAAAUA9QAaWAAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAV
wBbAF8AYwBnAGsAbwAAJTM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjJTM1IwUzNSM1MzUjBTM1IyUzNS
MHMzUjBzM1IwcZNSMHMzUjBzM1IwcZNSM1MzUjBTM1IyUzNSMFMzUjATMRIwcZNSMHMzUjBzM1Iwc
ZNSMHMzUjBzM1IwMZPz99Pz+7Pj68Pz99Pz98Pj4BtT4+/ks+PgG1Pj7+Sz4+Au0/P30/P30/P30/
P30/P30/P3w+PgG1Pj7+Sz4+AbU+Pv5LPj4Daj4+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+P
j4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj4+Pn0/Pz8+Pz8//NQDQd4+Pj4+Pj4+Pj4+PgAAAAAEAAAAA
PzA/MAAwAHAAAsADwAAJSE1ISUhNSE1ITUhJSE1IQFFAQ/9Uf7HA+j8GAe5AQ/9Uf7HA+j8GAw/+j7
6Pvo/AAMAAAAA/MDTQASAD0AgAAAATMfBRUHAYETPwQzAx8LMyEfBxUhDwcDETU/BgcRIRM/Ai8L
Iz0BLw0jIS8LkWIPDQOWBgQFBgYDAQGu/VjSAWIDCAGEQgUFBQV7BgCHBwCICAgBCACFBgQEAWIB/
LENDQwLCgoIA7ECAwQFBQYXgMiWQAQIBQHUHCAoJCwsMBmMCAgMEBQUGBwCICAKJCQR++AUFBQ
V7BgCHBwGHCAigCgkJCAkHCAYGBgQEBAIBA4BAGUGCAGFBf5zAaQEAWMFAGe5AQECA2IEBQMDAgI
BAQIDAwUFBgZeAQMEBgJCwX+nwJqBgYFBQMDAgEf/PMBtQwMCwWMCwoKCQgGBQQCav4JCQkJCAgH
BwYFBQQDAgIBAQIDYgUEAwMCAgECAGMEBQUGBwCICAKJCQAAAwAAAAAD8wPzAAMABwALAA3ITUhN
SE1ITUhNSEMA+j8GAPO/BgD6PwYDD/6u/r6AAAAAAUAAAAA/MD8wADACMAKwAvAE8AAAEVITUwD
MfBz8HLwYrAQ8BJREjNSEVIXEBESERAYsBDwCVazMVITUzAzUvBysBESECu/6KswQDAQEBAgIEBQY
FBgYGBQUEAwIBAQIDBAQGBQcGBQYDhRv+DLsCcP6KP7sHBgYLCgkGBQIB+gH0+gECAGYHCgoMBge7
/gwBRfr6sgUFBgYGBgUFBAMBAQEBAWQFBQYGBgYFBQQDAgIDQ/6Ku7sBdgF3/sGBOP7IAQIFBgkKC
wYG/kr9fQG8BgYGCgoHBgQBAXcAAAAABwAAAAAD8wPzAAMABwALAA8AEwALADEAAAAEVIzUjFSM1Ix
UjNQEVIZUTFSM1ITMVIxUzFSM1IxUjNSMRIREhBRCHfzCXNyc3JwcnA7X6Pvo++gNq+vr6/unZ+vr
6Pvo/A+j9sP5ocHAsCHatCHatCHABRfr6+vr6+gE4+voBOPr6+j76+vrd/awD6CxcwC1wcC1wcCw
cAADAaaaaAN2A/MAAwAGAA4AADchNSEBIRMBMzchFzMBI4kC7v0SAf3+84f+yE5OATHOTv7vTwX9A

bUBd/1R+voC7gAAABUAAAAA9QD1AADAACAcWAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMAUQ
BVAFkAXQAAJTM1IwcZNSMHMzUjBTM1IwcZNSMHMzUjJTM1IwUzNSMlMzUjBTM1IwEzNSMFMzUjJTM
1IwUzNSMlMzUjBzM1IwcZNSMHQHqEhFSErMxEhNSERiwcZNSMHMzUjBzM1IwOWPj59Pz99Pz/+iT8/
fT8/fT8/A2s+PvyVPz8Daz4+/JU/PwNrPj78lT8/A2s+PvyVPz8Daz4+fT8/fT8/u/5KAbY+Abb+S
j68Pz99Pz99Pz8sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/PwE4Pz8/Pj8/Pz4+Pj4+Pj59+j7+SwG1PgG1Pj
4+Pj4+AAAAABAAAAAD8wPzAAMADwATABsAAAEVITUBFwcXNxc3JzcnBycBFSE1ByMRMxEhESEDtF6
J/c5wcCxcwC1wcC1wcAN9/ok+Pj4B9P4MAUX6+gEMCHAsCHAsCHAsCHABOPr6+v6K/scD6AACAAAA
AAMvA/MAAwAAAA3ITUhNycHCQEnBxEj5wIy/c765CwBLwEvLOQ+DD/m5Sz+zwExLOUCwWAAAAEA
AAAAAPzA/QAAwAHAAsAGQAAJSE1IREhNSERITUhBRc3EScHFzcnBxEXNycBgwJx/Y8Ccf2PAnH9j/
6JKlNTKpydKlNTKp2JPwE4PgE5Pk8uS/z6Sy6Oji5LAWZLLo4AAAAAGwAAAAAD1APUAAMABwALAA8
AEwAXABsAHwAjACcAKwAvADMANwA7AD8AQwBHAESATwBTAFcAwWbFAGMAZwBrAAALMzUjBzM1IwcZ
NSMFMzUjBzM1IwcZNSMlMzUjBTM1IyUzNSMFMzUjJTM1IwcZNSMHMzUjBTM1IwcZNSMHMzUjJTM1I
wUzNSMlMzUjBTM1IyUzNSMHMzUjBzM1IwMzESMHMzUjBzM1IwcZNSMDlJ4+fT8/fT8//ok/P30/P3
w+PgNqPj78lJ4+A2o+PvyWPj4Daj4+fT8/fT8//ok/P30/P3w+PgNqPj78lJ4+A2o+PvyWPj4Daj4
+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+Pj4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj59Pz8/Pj8/
Pz4+Pj4+PvxYA6g+Pj4+Pj4AAgAAAAAD8wPzAAgADAAExc3ETMRfzcBJSE1IbIs5D7kLP7R/isD6
PwYAhYs5v08AsPlLAExbj8AAAAAQAAAAAD8wPzAIoAAAKBISMPHh8fMzUrAS8dPQE/HTMHARcJAQ
JAAS399xIREREREBAQDw8ODg4NDawLCwoKCQgIBWYFBQQDAgEBAQECAwQFBQYHCAGJCgoLCwwMDQ4
ODg8PEBAQERERERJeXg8ODg4NDg0MDQwMCwsLCgoJCQgIBWcGBgUFAwQCAgEBAgIEAwUFBgYHBwgI
CQkKCgsLCwwMDQwNDg0ODg4PAhD+yygBj51A8X+9gEDAwMFBQcHBwgJCgoLCwwNDQ0ODw4QDxAQE
REREhESERERERAE8PDg4ODQwMCwsKCgkICAcGBQUEAwIBAT8BAGMDAwUFBgYHBwgICQkKCgsLCw
wMDQwNDg0ODg4PDg8ODQ4NDQ0NDawLCwsKCgkICAgHBWYGBQQUEAwMCAf73LwFTAVwAAAAcAAAAAP
UA9QAAwAHAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMVwBbAF8AYwBnAGsA
bWAAANYe1ISUZNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSMhMzUjBzM1IwcZNSMFMzUjBzM1IwcZ
SMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjJTM1IwcZNSMHMzUjBTM1IwcZNSMHMzUjBTM1Iy
wDqPxYA2o+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+AbU+PgG1Pj59Pz99Pz/+iT8/fT8/fD4+AbU+PgG
1Pj7+Sz4+/ks+PgNqPj7+Sz4+/ks+PgNqPj59Pz99Pz/+iT8/fT8/fD4+AbU+Piw+Pj8/Pz8/Pj8/
Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+Pj4+Pj4+Pj4+Pj4+PgAAAQAAAAAD1APUAAsAA
AEhFSErMxEhNSERiWWh/koBtj4Btv5KPgIfPv5KAbY+AbYAAwAAAAADdgPzAAcAJABIAAABFSE1Mx
EhESUfBxUzFSE1Mz0BPwg7ARcnDwsjESERiY8ODwIBBgH0Pv2QAVUGBQqHBQIDAX3+in0BAwMEBgU
HCQsNEAdHBQYKCgwLBwMHAwIB+gLu+gECAwUFBgMDgoLCwwMDQwNDAM4fX39EwLEqMEBQoLBg4N
Nj8/JxYKCGkIBWcFBAMBNQIDBwcMDgoGEQsNDPyVA2sMDQsMCwoKDasHBQqEAgEBAgMAAAAAABgAAA
AAD8wPzAAMAQwBHAiCAiWDLAAALITUhBR8PPw8vDw8OASE1KQEfDz8PLw8PDgEhNSElHw8/Dy8PDw
4BRQKv/VH+xwEBAgQEBAyGBggHCAkJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCAcJCAkJCgo
JCQkIBWgGBgYEBAQCAQE4Aq/9Uf7HAQECBAMFBgYGBwgICQkJCgkKCQgJBWgGBgYEBAMDAQEBAQMD
BAQGBgYIBwkICQoJCgkJCQgIBWYGBgUDBAIBATgCr/1R/scBAQIEAwUGBgYHCAGJCQkKCQoJCAkHC
AYGBgQEAWMBAQEBAwMEBAYGBggHCQgJCgkKCQkICAgHBgYGBQMEAgFLPh8KCQkICQcIBgYGBAQDAw
EBAQEDAwQEBgYGCAcJCAkJCgoJCQgJBWgGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkBTj4KCQkICQc
IBgYGBAQDAwEBAQEDAwQEBgYGCAcJCAkJCgoJCQgJBWgGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkB
Lj8fCgkICAgHCAyGBgQEAWMBAQEBAwMEBAYGBggHCQgJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEB
gYGCAcJCAkJAAIAAAAAAPzA/MAAwAHAAsAEQAVABkAHQAHAABFSM1IxUjNSMVIzUTMyEVITUBFS
M1IxUjNSMVIzUDIREhA7X6Pvo++vo+AjL8lgNq+j76Pvo/A+j8GAFF+vr6+vr6ATj6+gE4+vr6+vr
6/FcD6AAABAAAAAD8wPzAAsADwATABsAAAEEXBxc3FzcnNycHJwERiXehESMRAYEVITUhESEBg3Bw
LHBwLHBwLHBwAgb5/on6PwE5AXYBOfwYARlxcCxcwCxcwSxwCAJw/ooBdv6KAXb+Sz4+AfQAAAAAB
QAAAAAD1APUAAMABwALAA8AEwAAAREhESMRIREBESERiXehEQMhESEDlv6JPv6JAYz+iT7+iT4DqP
xYAEH+iQF3/okBdwG1/okBd/6JAXf8lgOoAAAAAIAAAAAA/MDtQBTAf8AAAEpBRU/BjsBHwkVDx4M
VMzUjPxEvDisBCQIXCQE3CQEnCQEDVw4ODQwNDawMDAwNDA0MBw0MCgkEAWMCAQECEBAYHCREMNw4M
CwoIBgICAFq0AQECCAsMNImpDAUEBAQCAgEBAQICBAUFBwCICQoKDAwMDRD8pQEx/s8YASyBjH+0
AEwMf7a/toB/gMDBQYHCDkKCAgGBAQCAgQFBwUFBQUBH4MDAsKCg4KKwsMDA4ODggICSU0BgYFCw
sLKRwODwgICQkJCgoLDAsLCgkICAYGBgQEAWIBAZD+cP5xJgGB/n8mAY8Bjyb+fgGCAIAAAAAA/M
DtQADAAGAAAERIREDKQERIQJ9/c4/AnEBd/wYA3f9EgLu/NQDagAAAAGAAAAA/MD8wADAACAcWAP
ABMAFwAbAB8AACUzNSMFITUhJTM1IwUhNSElMzUjBSE1ISUZNSMFITUHA7U/P/xXAYz81AOpPz/8V
wG2/koDqT8//FcCcf2PA6k/P/xXAYz81Aw/Pz/6Pj4++j4+Pvo/Pz8AAQAAAAAC2gPzAAMAACUzAS
MBJUkBBuGMA+gAABsAAAAA9QD1AADAACAcWAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMARwB
LAE8AUwBXAFsAXwBjAGcAawAAJTM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFMzUjBTM1
IyUzNSMFMzUjBTM1IzUhNSElMzUjBTM1IwUzNSMlMzUjBTM1IwUzNSMlMzUjBzM1IwcZNSMHMzUjB
zM1IwcZNSMHMzUjA5Y+Pn0/P30/P7s+Prw/P30/P3w+PgNqPj7+Sz4+/ks+PgNqPj7+Sz4+/ks+Pg

```
Oo/FgDaj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4Daj4+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4
+Pj4+Pj4+Pz8/Pz8+Pz8/Pz99Pn0/Pz8/Pz4/Pz8/Pz4+Pj4+Pj4+Pj4+Pj4+ABWAAAAA9QD1AAD
AAcACwAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMARwBLAE8AUwBXAFsAXwBjAGcAawBvAAAlM
zUjBzM1IwczNSMHMzUjBzM1IwczNSMlMzUjBTM1IyUzNSMFMzUjJTM1IwczNSMHMzUjBzM1IwczNS
MHMzUjBzM1IyUzNSMFMzUjJTM1IwUzNSMlMzUjBzM1IwczNSMHMzUjBzM1IwczNSMDMxEjA5Y+Pn0
/P30/P7s+Prw/P30/PwLuPj7+Sz4+AbU+Pv5LPj4BtT4+fT8/fT8/fT8/fT8/fT8/fT8/Au4+Pv5L
Pj4BtT4+/ks+PgG1Pj59Pz99Pz+7Pj68Pz99Pz98Pj4sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/P30+Pj4+P
j4+Pj4+Pj4+fT8/Pz4/Pz8+Pj4+Pj4+Pj4+Pj4+Pj78WAOoAAAAAAGAAAAA/MD8wAFAAKAEQAZAB0AIw
AnADMAADcjFTM1IzMhNSkBMxUzNTM1IzcjFTM1IzUjMyE1KQEzFTM1IyUhNSErARUzFSMVMzUjNSO
Jfbw/vAKv/VH+xz8+P7w/P7w/PvoCr/1R/sd9P7wBOQKv/VH6Pz8/vD8+Sz99Pz8/Prw/Pz4+Pn36
Pj4/Pj68AAIAAAAA/MC+QCHARQAAAEfBzsBHw0dAg8NKwIvDT0BLwcPBxUfDyE/DzUvDyMPBgUVH
w8zPwY9AS8GKwEvDT0CPw07Ah8ZPwcvEyMPDgK7AQIDBAQFBgddDQwMDAsKCgkIBwYFBQMCAgMFBQ
YHCAkKCsMDAwN+gwNDAsLCgoJCAcGBGQJdAgECaWQEBgUHBgYFBQQAQAgEBaWUGCAkLDA0PDwgREhI
TAQMUEhIREQ8PDQwLCQQHBgQCAQMFbgJCwwNDw8IERISE2cHBQYEBAMC/VABAwUGCAkLDA0PDwgR
EhITZwcFBgQEAWICaWQEBgUHXQ0NDAsLCgoJCAcGBGQJdAgIDBAYGBwgJCgoLCwwNDfkKCQkJCAkIC
AcHBgYGBQYEBAMCAQIDBAQFBGcGBGUFaWMAQEDBQYGBwCJCQoKCwwMDA0NDg40+RMTEhERDw8NDA
sJCAYFAwLbBwUGBAQDAgECaWQGBGcICQoKCwsMDQx9DQ0MCwsKCgkIBwYGBAMCAgMEBgYHCAkKCs
LDA0NRQcFBgQEAWIBAQIDBAQGBQdFFBISEREpdw0MCwkEBwYEAgEDBQYICQsMDQ8PCBESEhOGFBIS
EREpdw0MCwkEBwYEAgECaGQFBQaifRQSEhERDw8NDAsJBACGBAIBAgIEBQUGBgCfBgQEAWIBAgMEB
gYHCAkKCsLDA0MfQ0MDAwLCgoJCAcGBQUDAgECaGMEBAUFBgYHBwCJCAwMDBMBGQUEAgIBAQICBA
UFBgYTEXIRDQwMCwoJCQgHBgUFaWMAQEDBQYICQsMDQ8PERESEgAAAAQAAAAA/MD8wADAACAwA
PAAA3ITUhJyE1ITChNSEnITUhqAKw/VCCa+j8GJwCsP1QnAPo/BgMP/o++j76PwAFAAAAAAPzA/MA
AwAHAAsAGwAnAAABFSM1IxUjNSMVIzUDMzUzFTM1MxUzNTMVMxEhJSMVMxUzNTM1IzUjA7X6Pvo++
j8/+j76Pvo//BgB9H19Pn19PgI++fn5+fn5/c76+vr6+voCcfo/fX0/fQAAAAASAN4AAQAAAAAAA
ABAAAAAQAAAAAAQAZAAEAAQAAAAAAAGAHABoAAQAAAAAAAwAZACEAAQAAAAAABAAZADoAAQAAAAA
ABQALAFMAAQAAAAAABgAZAF4AAQAAAAAACgAsAHCAAQAAAAAACwASAKMAAwABBakAAAAACALUAAwAB
BAkAAQAYALcAAwABBakAAgAOAOkaAwABBakAAwAyAPcAAwABBakABAAYASkAAwABBakABQAWAVsAA
wABBakABgAYAXEAAwABBakACgBYAaMAAwABBakACwAkAfsgRmluYWwgU2FtcGx1IGJyb3dlciBpY2
9uc1JlZ3VsYXJGaW5hbCBTYW1wbGUgYnJvd2VyIGljb25zRmluYWwgU2FtcGx1IGJyb3dlciBpY29
uc1ZlcnNpb24gMS4wRmluYWwgU2FtcGx1IGJyb3dlciBpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5n
IFN5bmNmdXNpb24gTWFV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAARgBpAG4AYQBsACAAU
wBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAA
RgBpAG4AYQBsACAAUwBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBGAG8AbgB0A
CAAZwBLAG4AZQByAGEAdABLAGQAIAb1AHMAAQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQ
BLAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0
AAAAAGAAAAAAAKAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA/AQIBAwEEAQUBBgEHAQgBCQEKAQsB
DAENAQ4BDwEQAREBEgETARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAELASyBjWEOA
SkBKgErASwBLQEuAS8BMAExATIBMWEOATUBNgE3ATgBOQE6ATsBPAE9AT4BPwFAAAATdHJva2VTdH
lsZQhCb29rbWfYawdQaWN0dXJlBEZpbmQNT3V0c2lkZUJvcmlRlCgdKdXN0aWZ5BUNsb3N1dKrlY3J
lYXN1SW5kZW50FVBpeGVsQWxpZ25DZW50ZXJUYWJsZQ9CYWNrZ3JvdW5kQ29sb3ILQWxpZ25Cb3R0
b20JUGFnZVNldHVWdKhpb2hsaWdodENvbG9yC1NlcGVyc2NyaXB0BVRhYmx1BFVvZG8LSW5zZXJ0Q
mVsb3cJVG9wQm9yZGVyC1BhZ2V0dW1iZXIQQWxpZ25DZW50ZXJUYWJsZQ5JbmNyZWZfZUluZGVudA
RCb2xkCUFsaWduTGVMdAZGb290ZXILSW5zZXJ0UmlnaHQJVW5kZXJsaW51Ckluc2VydExlZnQETG9
jawZIZWfKZXINU3RyaWtldGhyb3VnaAhDbGVhckFsbAtSaWdodEJvcmlRlCgpbBbGlnblJpZ2h0BE9w
ZW4KU3Ryb2t1U216ZQVQcmludAtEZwXldGVUYWJsZQ1Gb250Q29sb3INSW5zaWRlQm9yZGVycwpeZ
WxldGVSB3dzCERvd25sb2FkC0xpbnVtcGFjaW5nFEluc2lkZVZlcnRyY2FsQm9yZGVyCEFsawduVE
9wBFJlZG8MQm90dG9tQm9yZGVyA05ldwVQYXN0ZQdCdWxsZXRxZBENlbGwNRGVSzXRlQ29sdWlucwp
BbGxCb3JkZXJzCVN1YnNjcmlwdBBTaG93SGlkZVByb3B1cnR5d1RhYmx1T2ZDb250ZW50Bk10YWxp
YXZJbnNpZGVib3Jpem9uZGFsYm9yZGVyC0xlZnRcb3JkZXJzCU51bWJlcm1uZWwMaW5rC0FsaWduQ
2VudGVyC0luc2VydEFib3ZlAAAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="e-de-icon-"],
[class*=" e-de-icon-"] {
```

```
font-family: 'Sample brower icons';
}
.e-de-icon-Bullets:before {
  content: "\e730";
}
.e-de-icon-Numbering:before {
  content: "\e73a";
}
</style>
```

Opening a default document in DocumentEditorContainer

By default, Document Editor Container will open a blank document. You can use [open](#) API in Document Editor to open the SFDT content.

Document editor Container have [created](#) event which gets triggered once Document editor container control created. So, if you want to open the document by default, you can use `open` and `created` API.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let documenteditorcontainer;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function created() {
    // load your default document here
    let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading 1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}],"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}],"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":"None","strikethrough":"None","baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"},"name":"Default Paragraph Font","type":"Character","characterFormat":{}},{name":"Heading 1
```

```

Char", "type": "Character", "characterFormat": {"fontSize": 16, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 0, "outlineLevel": "Level1", "listFormat": {}}, "characterFormat": {"fontSize": 16, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 1 Char", "next": "Normal"}, {"name": "Heading 2 Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6, "outlineLevel": "Level2", "listFormat": {}}, "characterFormat": {"fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2 Char", "next": "Normal"}, {"name": "Heading 3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {}}, "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal"}, {"name": "Heading 3 Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}}, "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal"}, {"name": "Heading 4 Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal"}, {"name": "Heading 5 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal"}, {"name": "Heading 6 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml": []}`;

    // Open the default document
    documenteditorcontainer.documentEditor.open(data);
  }

  function componentDidMount() {

```

```

        setTimeout(() => {
            created();
        });
    }
    return (<DocumentEditorContainerComponent height={'590px'} id="container"
ref={(scope) => {
        documenteditorcontainer = scope;
        created();
    }} serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-
react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
    let documenteditorcontainer: DocumentEditorContainerComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function created() {
        // load your default document here
        let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":
72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"te
xt":"Adventure Works
Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFo
rmat":{},"characterFormat":{},"inlines":[]]}]},"footer":{"blocks":[{"paragrap
hFormat":{"listFormat":{},"characterFormat":{},"inlines":[]]}]}]},"character
Format":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","un
derline":"None","strikethrough":"None","baselineAlignment":"Normal","highligh
tColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Cal
ibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"li
neSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"b
idi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"ha
shValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection"
,"dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"
name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.14999997
6158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fon
tFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph
Font","type":"Character","characterFormat":{}}, {"name":"Heading 1
Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph

```



```

Font"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 0,
"outlineLevel": "Level1", "listFormat": {}}, "characterFormat": {"fontSize": 16, "fo
ntFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 1
Char", "next": "Normal"}, {"name": "Heading 2
Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6, "
outlineLevel": "Level2", "listFormat": {}}, "characterFormat": {"fontSize": 13, "fon
tFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2
Char", "next": "Normal"}, {"name": "Heading
3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el3", "listFormat": {}}, "characterFormat": {"fontSize": 12, "fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3
Char", "next": "Normal"}, {"name": "Heading 3
Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calib
ri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el4", "listFormat": {}}, "characterFormat": {"italic": true, "fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4
Char", "next": "Normal"}, {"name": "Heading 4
Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el5", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5
Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml
": []}`;

    // Open the default document
    documenteditorcontainer.documentEditor.open(data);
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }

```

```

    });
  }
  return (<DocumentEditorContainerComponent height={'590px'} id="container"
ref={(scope) => {
    documenteditorcontainer = scope;
    created();
  }}
    serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'))
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
  <script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>

```



```
<div id='loader'>Loading....</div>
</div>
</body>
</html>
<style>
  /** Document editor sample level font icons*/
@font-face {
  font-family: 'Sample browser icons';
  src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSi8AAAEoAAAAVmNtYXQ0OvsAAACfAAAAALBnbHlmA57A
AgAAA6wAAC90aGVhZBF0mKkAAADQAAAAANmhoZWElUARAAAAAArAAACRobXR4/AAAAAAAAAYAAAD8b
G9jYXhXbWwAAAMsAAAAAGlHeHABZQE/AAABCAAAACBuYwllChIPawAAMyAAAAAL9cG9zdAI/4kAAAD
YgAAADOQABAAAEAAAAAFwEAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAPwABAAAAAQAAAGurlbV8PPPU
ACwQAAAAAAAAncxqdwAAAAA1zGp3AAAAAAD8wP0AAAAACAACAAAAAAAAAAAAEAAAAA/ATMAHAAAAAAAAGAA
AAoACgAAAP8AAAAAAAAAQQAQAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAAAAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wDnPPQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAAEAAAAABA
AAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAE
ABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAE
AAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAQAA
AAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAEAAAAABAAAA
QAAAAEAAAAAAGAAAAAMAAAAUAAAMAAQAAABQABACcAAAAABAAEAAEEAAoC9//8AAoCa//8AAAAABAAQ
AAAABAAIAAAwAAEAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAEgATABQAFQAWABcAGAAZABoAGwAc
AB0AHgAfACAAIQaiACMAJAALACYAJwAoACKAKgArACwAlQAUAC8AMAAxADIAMwA0ADUANgA3ADgAO
QA6ADsAPAA9AD4AAAAAFoAngDuAg4CWAJ4ApoCcxgMGA9QD8gVgBcoGSgAMByoHYggKCLII3AkICb
uJ3An4CjIKvAr4C8QL4AwADEIM6g0MDAwNxxg42DoIoPa8yD2YPhA+2EFgQdhEWEcAR2BI4EyYTXhO
UE8AUPhRWFUJAnhVAFegWMBdiF4IXugAOAAAAAAPzA7UAAwAHAAsADwATABcAGwAFACMAJwArAC8A
MwA3AAALMzUjBzM1IwcZNSMHMzUjBzM1IyUzNSMFmzUjBTM1IyUhNSEFITUhJTM1IwUzNSMHMzUjB
zM1IwO1Pz+7fX36fX36fX36fX0C7vr6/on6+v6J+voB9AH0/gz+DAF3/okC7vr6/si7u/p9fbw/P0
t9fX19fX19fX19fX19fX19fHx8fX19fX19fX0AAAAACAAAAAAN2A/MABAArAAAABEwkBEQMfCTM/BAK
BHwYzPwkRIQM4Af7H/sg/AQIDBQYGCakJCQkJCQkIBwEKAQsFBQUGBgYGDawFCQgGBgUDAgH9EgO1
/JUBZ/6aA2r8lgoJCAgHBWUEAwEBawQFBwEx/s4FBAMDAGEBAGIEBQYICAgJCgOpAAAAABQAAAAAD8
wPUAAQACAAAnAC4AMgAAJRujJzCHITU3JQ8DHQEFbj8GNS8GDwETEScHAQcRAyERIQO1j51SDf192g
HdAgICAgICBAUGBgYGBgYFbAMBAMQMCBwUGCwkFrtp9/sfaPwPo/Bh9E5xR7c7bQgIDBQYHBgYFBQQ
DAQEBAQMEBAUGCwoFAgYDAQECAwE9/UDZfQE42gIG/JYDgAAAAAIAAAAAA/MD8wB/AQUAAAEVDx0r
AS8dPQE/HTsBHx0FFR8HAQ8DHwgZPwQBHwc/Hy8fDx4DtQECAwMFBQUGBwgICQkKCgsLDAwNDQ40D
g4PDw8QEBAQE8QDw8ODg4ODQwNDAsLCgoJCQgIBGcFBQQEAwIBAQIDBAQFQBQYHCAGJCQoKCwsMDA
0NDg4ODg8PDxAQEBAQDxAPDw4ODg4NDA0MCwsKCGkJCAgGBwUFBAQDAGH9UQEEBgcKDA4P/s8GBQI
BAGMGCAQFBGsMDAwLBQUFAS0YGBobHB0dHhMTEXiTERIREBAQDw8ODg0MDAsLCQkJBwcGBQQDAwEB
AQEDAwwQFBgcHCQkJCwsMDA0ODg8PEBAQERIREXiTEXMTEXMSExESEERARDw8PDg4NDAwLCwoJCAcHB
gUEBAIBAN0QEBAQDw8ODw0ODQ0MDAsLCgoJCQgIBwYFBQUdAwIBAQIDAwUFbQYHCAGJCQoKCwsMDA
0NDg0PDg8PDxAQEBAQDxAPDw4ODg4NDA0LDAsKCGkJCAcHBwUFBAQDAGEBAGMEBAUFbwcHCAkJCgo
LDAsNDA0ODg4ODw8QDxAQDw8dHRwbGhgY/s4KCgsLCwsKCQUDBAQCAgQEAWUBLRAODAoHBgQBAQED
AwQFBGcHCAoJCwsMDA0ODg8PDxEQERIREXiTEXMTEXMSExESERAQE8PDg4NDAwLCwkJCQcHBgUEA
wMBAQEBAwMEBQYHBwkJCQsLDAwNDg4PDxAQEBESEhISEXMAAAsAAAAA9QD1AADAACACwAPABMAFw
AbAB8AIwApAC8AACUzNSM1MzUjNzM1IwcZNSMHMzUjBzM1IwcZNSM3MzUjNTM1IyczIREhESMRfSE
RIQHhPj4+Pvo+Pn0+Pn0+Pn0+Pn0+Pvo+Pj4++j4CcFzUPgOo/FjnPj8+Pz4+Pj4+Pj4+Pj8+Pz59
/NQDLPzUPgOoAAAAEAAAAAAPzA/MAAwAHAAAsADwAAANYe1ITUhNSE1ITUhNSE1IQwD6PwYA+j8GAPO/
BgD6PwYDD/6Pvo++j8AAAAAAQAAAAADtQO1AAsAABMJARcJATcJAScJAUsBif53LAGJAYks/ncBiS
z+d/53A4n+d/53LAGJ/ncsAYkBiSz+dwGJAAAFAAAAAAPzA/MAAwAHAA0AEQAVAAA3ITUhJSE1ISU
XNyc3JxchNSE1ITUhDAPo/BgBOQKv/VH+x5IqaWkqpwKv/VH+xwPo/BgMP/o+fZwscHAsHz76PwAA
BwAAAAAD8wPzAAMABwATABcAGwAFACsAACUzNSMHMzUjNyMVMxUzNTM1IzUjJSE1ISUzNSMHMzUjF
yMVMxUzNTM1IzUjAn0+Pvo/P30+Pj8+Pj+/DAPo/BgCcT4++j8/fT4+Pz4+P8g+Pj4/P/r6Pz59Pr
w+Pj4+Pz4+P/oAAAAEAAAAAAPzA/MAMAAZAGKApwAAJRUPDi8OPQE/Bx8GAQcnBQ8JFR8OPw81Lwk
BFQkCJwcXByEBNT8GOWEfBhEzETUvDg8OA6sBAGMDAwUEBgUGBwYHBWgHBwCHBgYGBQUEBAMCAGEB
AgYJChINDRSMcwkIBAL+p0riASMBNBUJCggHBQMBAwMFBGcJCQsLDA0NDg4PDw8ODQ0MCwoKCACGB
QQCAQMEBgwJCGoVEZT94/7HAVGBlOUwYBX98QCEBAQIDAwUEFgCGBgUFfAwMCAT4CAGMEBQUGBwCICA
kJCQkKCQkJCAgHBWYFBQQDAgKuCQkICAgHBwCfBQUEAwIBAQEBAQMEBQUFBwCHBwkICQkHCQqTFRU
```

fFRQpFRUVExIJAQ3i4iMCSQSExQTExMRERAPDw4ODAsLCQgHBQQDAQEDBAUHCakLCwwODg8PEAgR
ExMTHRMTEiAcQgHUcP67/qgBh6AodBQBDIoGBgUFBAMCAgMEBQUGBv7nArkKCQkJCAcIBgYGBAQDA
wEBAQEDAwQEBgYGCACJCAkJAAAAgAAAAAD8wPzAAMADAAANyE1ISUnBwkBJwCRiwwD6PwYAfTkLA
EvAS8s4z8MP+blLP7OATIs5QLDAAAABgAAAAAD8wPzAB8AXwCfAOIA5QEYAAABFQ8FKwEvBj8GOWE
fBQcVHW4/Dy8OIw8OFw8PLw8/Dx8OJyMPAYcHFW8EJwcfBACXNx8DBxc3HwE/Ahc3Jz8DFzcnPwUn
By8DNyCHLwM1IycjNSURHw8hNSEjLwU1ETU/BTMhFTMVMz0BLw8hDw4DEgICAwQEBAUFBQQDAwMBA
QEBAwMDBAUFBQQEBAMCAm8CAgMDBQUFBwYHCAgICQkJCAkIBwCHBgYFBAQDAgEBAQECAwQEBQYGBw
cHCAkICQkJCAgIBwYHBUFAwMCA4t4BAGMFBQcICQkLCwwMDQ0ODg4MDQwLCgoJBwCGBQMCAQECAwU
GBwCJCgoLDA0MDg4ODQ0MDAsLCQkIBwUFAwKiAhQTEhIiKiIJCwoIBDMKNAEDBQYvHDAODg8TFDQU
FBQPDwkUNBQSDw0QMBwvBQUEAQE0CjMICAoQIioiFRESFTgQkP3OAQECBAQEBgYGCACICQkJCgGW/
moGBgYEBAMCAgMEBQUGBgGW+j4BAwMEBAYG1gYICAgJCAoJ/mUKCQkJCAcIBgYGBAQEAgEBBgUEBA
QDAgICAgMEBAQFBQEAwMDAQEDAwMEBQUJCAkIBwCHBgYFBAQDAgEBAQECAwQEBQYGBwCHCAkICQkJ
JCAgHCAYGBgUEBAQEAwMEBAUGBgYIBwGICQkODQ0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCwSM
DA0NDg4NDQ0MCwoKCQcHBgQEAgEBAgQEBGcHCQoKCwwNDQ22BAYICikKQoQERILCTcKGBQTEhsxH
A4NCww3FDgDAQECATgTOAoLDBEcMBwNERMTDQk4CRQEBQpJCKLBwYENvqPdjfUCgkJCAkHCAYGBg
QEAWMBAT8CAwQFBQYGAywGBgUFBAMC+nyCCQkJCQgIBwFVBwUFBAMCAQEBAQIEBAQGBgYIBwGJCQk
AAAAABAAAAAADdgPzAAMABwAiAFMAADchNSEBFQc1AQ8KHQEHnzUvCSM7AR8PBzMVNZUzJz8PMzUj
FSE1I4kC7v0SABZ+ATIGBgOIBwUFAwMCAf6JAQIBAwQEBQcICgyECgoSEQ4MDAoIBwCFawMDAQEBA
m76bQIBAQICAwQFBgGICgsNDhESFD/9kD8MfQF3UESUATgGBg0NDg4ODg8PDxBfYA8PDw4PDg4NDg
0MAwQFBwGJCgsLDQ4ODhAPIH76jW1+IA8QDg4ODQsLCgkIBwUEA7x9fQACAAAAAAPzA7UAVABgAAA
BDwUVPwY7AR8JFQ8QFTM1Iz8SLw8HBQkBFwkBNwkBJwkBA1cODg0MDQwMDAwMDQwNDACNDAAoJBAMD
AgEBAgQGBwKRDDcODAsKCAYCAgH6tAEBAgQECwAGQ8MBQQEBAICAQEBAgIEBQUHBwGJCgoMDAwNE
PylATH+zzIBJgEmMf7QATAx/tr+2gOzAwMFBGcIOQoJBwYEBaICAUBHQGBGQcGDgWMCwoKDgorCw
wMDQ4PCAgIJTMHBQYFBQsLMBUPDwgICakJCgoLDAsLCgkICAcGBQQAeAwIBAqEm/nH+cCYBgV5/JQG
QAY8m/n4BggAACgAAAAAD8wPzAAMABwALAA8AEwAXABsAHwAjACgAAAEVIZUjFSM1IxUjNQEVIZUj
FSM1IxUjNQEVIZUjFSM1IxUjNQMPAREhA7X6Pvo++gNq+j76PvoDavo++j76PwE5Aq/8GAFF+vr6+
vr6ATj6+vr6+voBOPr6+vr6+vxxA+gAAAAAAQAAAAAD8wPzAIoAABMBNwEHmx8dHQEPHSsBFTM/Hy
8eIyEBJwBjSn+ygIQDw4ODg0ODQwNDawLCwsKCgkJCAgHBwYGBQUdAwMCAQECAwMDBQUGBgCHCAg
JCQoKCwsLDawNDA0ODQ4ODg9eXhIREREREBAQDw8ODg4NDawLCwoKCQgIBwYFBQQAQDAgEBAQECAwQF
BQYHCAgJCgoLCwwMDQ4ODg8PEBAQERERERL99wEtKQKY/q0vAQkCAQMDBAQFBgYHBwGICQoJCgsLD
AsMDQ0NDQ4NDg8ODw4ODg0ODQ0MDAwLCwsKCgkJCAgIBgcFBQUDBAICAT8BAQIDBAUFBGcICakKCg
sLDawNDg4ODw8QEBAEREREREhIREREREBAQDw8ODg0NDQwLCwoKCQgHBwCFBQMDAwEBcI8AAAUAAAA
AA/MD8wALAA8AEwAXACCAACUjFTMVMzUzNSM1IwEVIzUjFSM1IxUjNQMHESMVIzUjFSM1IxUjNSMC
AH19P3x8PwG1+j76Pvo/A+g/+j76Pvo/yD99fT99AXb6+vr6+vr+yAJx+vr6+vr6AAACAAAAAAPUA
9QAawAHAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPafMAVwBbAF8AYwBnAGsAbw
AAJTM1IwczNSMHMzUjBzM1IwczNSMHMzUjBzM1IyUzNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSM
hMzUjBzM1IwczNSMFMzUjBzM1IwczNSMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjNSE1IQOW
Pj59Pz99Pz+7Pj68Pz99Pz98Pj4Daj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4BtT4+AbU+Pn0/P30/P
/6JPz99Pz98Pj4BtT4+AbU+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+A6j8WCw+Pj4+Pj4+Pj4+Pj4+Pj
8/Pz8/Pj8/Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+PgAFAAAAAAOWA/MAAwAfACIAQAC
FAAABByM3JyMVMwCjFTMHFzcZBxc3MzUjNzM1IzcnByM3JyUjNScVMxEPBiMhIy8GET8GMwCRFR8O
IT8ONRE1Lw8hDw4CRxJ8EjZwZxJVTA0+DnwMPQ5vZhJVTA0+DnwMPQGIjz76AQIDBAQGBQf9kAcFB
gQEAWIBAQIDBAQGBQdeAgIDBAUFBGcCHCAgJCQkJAnAKCQkJCAgHBwYFBQQAQDAgICAgMEBQUg1gcHCA
gJCQkJ/mUKCQkJCAgHBwYFBQQAQDAgIBwn19Pj59P1kJY1kJYj59P1kJY1kJmI8s+v2vBgYFBQQAQDAgI
DBAUFBGyYDLAYGBQUEAwIf/NQKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgJXCQkJCQgI
BwfVBgYFBAWCAQEBAQMDBAQGBgYIBwkICQkAAAAADAAAAAAPzA/MACAAMABUAACUXNxERc3JyUhN
SELJwCXNycHESMBgyPTP1Mqnf3tA+j8GAH0UyqcnCpTPvYvTP75AQdML419Pq9ML42NL0wBBwAFAA
AAAAAPr/MAAwAHAA0AEQAVAAA3ITUhJSE1ISUXBxc3JwUhNSE1ITUhDAPo/BgBOQKv/VH+x29vLJu
baAQ0Cr/1R/scD6PwYDD/6PuxvbyybmX4++j8AAwAAAAADGQ01ACMARgCbAAABOEdfDg8KwEREx8P
Dw8jEQcVESE/GzUvDzU/DzUvECEBzQ0NGRgVFBQIDw0LCQgGBQIBAQIEBgcJCwwOdHdERExUVF5F7F
RQSERAODQwKCQgGBQMCAQECEBAYHCAAsLDg4PERITFBZtawEKHx4dDg0NDQwMDAsLCwoKCQgHBwYGBQ
QEAWICAQECEBQYICQsNDw8REhMUFhYSERAPDg0MCwoIBwYFAwIBAwQGBAUFBG0PERMVfHcZGxz+7gH
iAgMEBgCHCQsLDQ4PEBITEhEQDw4NDQsKCAgGBAQCAToBdwEBawMFBQcHCQkLCwwODhASEQ8PDg0L
CwoIBwUFAwIBARudP/3OAQMGAwQFBQYGBwCICakJCgoKCgsLDawMDQwODQ4WFRQTEhAQDw0MCgoHB
gUDAwYHCQkKCw0NDg8PEBAREhILFRUTCQkICRAPDQ0KCQcFAwIAAAAAABAAAAAD8wPzAAMABwALAA
8AADchNSE1ITUhNSE1ITUhNSEMAq/9UQPo/BgCr/1RA+j8GAw/+j76Pvo/AAAAAAMAAAAA7UD8wA
DAACaCwAANyE1IQERIREDIRehyAJw/ZACr/0SPgNq/Ja9vAI8/JYDavyXA+gABQAAAAAD8wPzAAMA

BwATABcAJwAAARUjNRMVizUFIXUzFTM1MzUjNSMnFSM1ITMVIxUzFSMVMxUjFSERIQI/+vr6AfN9f
T99fT/5+v7H+vr6+vr6AnH9jwFF+voBOPr6Pz59fT59+vr6+j76Pvo/A+gAAAACAAAAAN2A/MAAw
B4AAA3ITUhExUfhj8eNREjEQcVDxQrAS8UNQMjiQLu/RI/AQIDAwQFBgYHCAGJCQoKCwsMDA0NDQ4
PDg8PDxAQEBAQEAE8PDw4PDg0NDQwMCwsKCgkJCAGHBgYFBAMDAgE+AQICAwMEBQUMDQ8RExMWFgW
DAwNDA0NDA0MDAwMCwsWExMRDw0MCgQDAwICAT4MPwF3EQ8QDw8PDw4ODg0MDQsMCwoKCQgJBwCGB
gUEBAICAQEBAQICBAQFBgYHBwkICQoKCwwLDQwNDg4ODw8PDxAPEQIy/c4NDQwNDAsMDAsVFBIRDw
4LCgQEAgMBAQEBAwIEBAQGCw4PERIUFRcMCwwNDA0CPwAFAAAAAAPzA/MAAwAHABMAFwAoAAABFSM
1ExUjNQUjFTMVMzUzNSM1IyUVIzUDKQE1IzUzNSM1MzUjNTM1IQK7+fn5/sd9fT98fD8CMvk/ATgB
Ofr6+vr6+v2PAUT5+QE5+vo/Pn19Pn36+vr8Vz/6Pvo++j8AAAADAAAAAN2A/MAJQBIAK8AAAEhO
wEfBRURFQ8FIyEjLwU1ETU/BTM1FSM1Pw47AR8NBRUjDw8RHw8hPw8RLw8jNS8PDw4BRQF2XgYGBg
QEAWICAwQFBQYG/c4GBgYEBAMCAGMEBQUGBgGW+gECawQFBggICQkLCgWMDA0NDAwMCgsJCQgIBgU
EAwL+yV4KCQkJCACIBgYGBAQEAgEBAQECEBAQEAgYGCACICQkJCgIyCgkJCQgHCAYGBgQEBAIBAQB
AgQEABYBggHCAkJCQpEQIFBggKCg0NDhAQERITExMTEhEQEA4NDQoKCAyFAgi+AgMEBAYFB/5LB
gYFBQQDAgIDBAUFBgYBtQcFBgQEAWL6u7sNDAwMCwoKCQgHBgUFawICAwUFBgICQoKCwwMDA27AQ
ECBAMFBgYGBWgICQkJCv5LCgkJCQgHCAYGBgQEBAIBAQBAGQEBAyGBggHCAkJCQoBtQoJCQkICAc
GBwUFBAMCAQG7ExMSEREpdg4MCwkIBgUDAQEDBQYICQsMDg4PERESEwADAAAAA01A/MAAwAHAAAsA
ABMhNSE1ESERAYERICgCcP2QAQ/9Ej4DavyWAoe8cvyWA2r8VwPoAAMAAAAA5YDtQADAAcADwAAJ
TMRIyUhNSERIREzESE1IQHhPj7+iQMs/NQBdz4Bd/zUSwE4Pz4Bd/7HATk+AAADAAAAAAPzA7UADA
AQACcAACUHIy8DPQE/AyUJAw8HHwghNQUJAQIUP9GyAwICAgID1QK0/qX+1AFb/bYGBQQDAwIBAQE
BAGMDBAUGxQMK/joBxv57xD2tAwQEBAQEASRWP6xASEBUP4fBgYHCAGICAgICAgIBwCGBr8+AgG3
AXcAAAAcAAAAAPUA9QAawAHAAAsADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAV
wBbAF8AYwBnAGsAbwAAJTM1IwcZNSMHMzUjBzM1IwcZNSMHMzUjJTM1IwUzNSM1MzUjBTM1IyUzNS
MHMzUjBzM1IwcZNSMHMzUjBzM1IwcZNSM1MzUjBTM1IyUzNSMFMzUjATMRIwcZNSMHMzUjBzM1Iwc
ZNSMHMzUjBzM1IwMZPz99Pz+7Pj68Pz99Pz98Pj4BtT4+/ks+PgG1Pj7+Sz4+Au0/P30/P30/P30/
P30/P30/P3w+PgG1Pj7+Sz4+AbU+Pv5LPj4Daj4+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+P
j4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj4+Pn0/Pz8+Pz8//NQDQD4+Pj4+Pj4+Pj4+PgAAAAEAAAAA
PzA/MAAwAHAAAsADwAAJSE1ISUhNSE1ITUhJSE1IQFFAQ/9Uf7HA+j8GAE5AQ/9Uf7HA+j8GAw/+j7
6Pvo/AAMAAAAA/MDtQASAD0AgAAAATmfBRUHAYETPwQzAx8LMYEfBxUhDwcDETU/BgcRIRM/Ai8L
Iz0BLw0jIS8LkWiPDQOWBgQFBgYDAQGu/VjSAwIDCAGEQgUFbQV7BgCHBwCICAgBCAcFBgQEAWIB/
lENDQwLCgoIA7ECAwQFBQYXgMiWQAQBAQICBUHCAoJCwsMBmMCAGMEBQUGBwCICakJCQR++AUFbQ
V7BgCHBwGHCAigCgkHCAyGBgQEBAIBA4BAGUGCAGFBf5zAaQEAWMFAG5AQECA2IEBQMDAgI
BAQIDAwUFBgZeAQMEBgcJCwX+nwJqBgYFBQMDAgEf/PMBtQwMCwwMCwoKCQgGBQQCav4JCQkHCAgH
BwYFBQQDAgIBAQIDYgUEAWMCAGECAGMEBQUGBwCICakJCQAAAwAAAAAD8wPzAAMABwALAAA3ITUhN
SE1ITUhNSEMA+j8GAPO/BgD6PwYDD/6u/r6AAAAAUAaaaa/MD8wADACMAKwAvAE8AAAEVITUnDw
MfBz8HLwYrAQ8BJREjNSEVIxEBESERAYsBDwCvAZMVITUzAzUvBysBESECu/6KswQDAQEBAgIEBQY
FBgYGBQEAWIBAQIDBAQGBQcGBQYDhRv+DLsCcP6KP7sHBgYLCgkGBQIB+gH0+gECAGYHCgoMBge7
/gwBRfr6sgUFBgYGBgUFBAMBAQEBAwQFBQYGBgYFBQMDAgIDQ/6Ku7sBdgF3/sGBOp7IAQIFBgkKC
wYG/kR9fQG8BgYGCgoHBgQBAXcAAAAABwAAAAAD8wPzAAMABwALAA8AEwALADEAAAEVizUjFSM1Ix
UjNQEVizUTFSM1ITMVIxUzFSM1IxUjNSMRIREhBRcHFzcXNyc3JwcnA7X6Pvo++gNq+vr6/unZ+vr
6Pvo/A+j9sP5ocHAscHAtcHAtcHABRfr6+vr6+gE4+voBOPr6+j76+vrd/awD6CxcwC1wc1wcCxcw
cAADAAAAAN2A/MAAwAGAA4AADchNSEBIRMBMzchFzMBI4kC7v0SAf3+84f+yE5OATHotv7vTwX9A
bUBd/1R+voC7gAAABUAAAAA9QD1AADAACACwAPABMAFwAbAB8AIwAnAcSALwAzADcAOwA/AEMAUQ
BVAfKAXQAAJTM1IwcZNSMHMzUjBTM1IwcZNSMHMzUjJTM1IwUzNSM1MzUjBTM1IwEzNSMFMzUjJTM
1IwUzNSM1MzUjBzM1IwcZNSMHMQEHFSERMxEhNSERIwcZNSMHMzUjBzM1IwOWPj59Pz99Pz/+iT8/
fT8/fT8/A2s+PvyVPz8Daz4+/JU/PwNrPj781T8/A2s+PvyVPz8Daz4+fT8/fT8/u/5KAbY+Abb+S
j68Pz99Pz99Pz8sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/PwE4Pz8/Pj8/Pz4+Pj4+Pj59+j7+SwG1PgG1Pj
4+Pj4+AAAABAAAAAD8wPzAAMADwATABsAAAEVITUBFwcXNxc3JzcnBycBFSE1ByMRMxEhESEDtF6
J/c5wcCxcwC1wcC1wCAn9/ok+Pj4B9P4MAUX6+gEMcHAscHAscHABOPr6+v6K/SCD6AACAAAA
AAMvA/MAAwAAAA3ITUhNycHCQEnBxEj5wIy/c765CwBLHEvLOQ+DD/m5Sz+zwEx/LOUCwwAAAAEA
AAAAAPzA/QAAwAHAAAsAGQAAJSE1IREhNSERITUhBRc3ESCHFzcnBxEXNycBgWJx/Y8Ccf2PanH9j/
6JKlNTKpydKlNTKp2JPwE4PgE5Pk8uS/z6Sy60ji5LAWZLLo4AAAAAGwAAAAAD1APUAAMABwALAA8
AEwAXABsAHwAjACcAKwAvADMANwA7AD8AQwBHAESAtwBTAFcAwWbFAGMAZwBrAAALMzUjBzM1IwcZ
NSMFMzUjBzM1IwcZNSM1MzUjBTM1IyUzNSMFMzUjJTM1IwcZNSMHMzUjBTM1IwcZNSMHMzUjJTM1I
wUzNSM1MzUjBTM1IyUzNSMHMzUjBzM1IwMzESMHMzUjBzM1IwcZNSMD1j4+fT8/fT8//ok/P30/P3
w+PgNqPj781j4+A2o+PvyWPj4Daj4+fT8/fT8//ok/P30/P3w+PgNqPj781j4+A2o+PvyWPj4Daj4
+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4+Pj4+Pj8/Pz4/Pz99Pj4+Pj4+Pj4+Pj59Pz8/Pj8/
Pz4+Pj4+PvxYA6g+Pj4+Pj4AAgAAAAAD8wPzAAGADAAAEExc3ETMRfzcBJSE1IbIs5D7kLP7R/isD6

PwYAhYs5v08AsPlLAExbj8AAAAAAQAAAAAD8wPzAIoAAAKBISMPHh8fMzUrAS8dPQE/HTMhArcJAQ
JAAS399xIREREREBAQDw8ODg4NDawLCwoKCQgIBwYFBQQDAgEBAQECAwQFBQYHCagJCgoLCwwMDQ4
ODg8PEBAQERERERJeXg8ODg4NDg0MDQwMCwsLCgoJCQgIBwCGBgUFAwQCAgEBAgIEAwUFBgYHBwgI
CQkKCgsLCwwMDQwNDg0ODg4PAhD+yygBj f51A8X+9gEDAwMFBQcHBwgJCgoLCwwNDQ0ODw4QDxAQE
RERERhESERERERAQE8PDg4ODQwMCwsKCgkICAcGBQUEAwIBAT8BAgMDAwUFBgYHBwgICQkKCgsLCw
wMDQwNDg0ODg4PDg8ODQ4NDQ0NDawLCwsKCgkJCAgHBwYGBQQEAWMCAf73LwFTAVwAAAAcAAAAAAP
UA9QAaWAAHAAADwATABcAGwAfACMAJwArAC8AMwA3ADsAPwBDAECASwBPAFMAVwBbAF8AYwBnAGsA
bwAAnyE1ISUzNSMFMzUjBTM1IyUzNSMFMzUjBTM1IyUzNSMhMzUjBzM1IwczNSMFMzUjBzM1IwczN
SMhMzUjJTM1IwUzNSMFMzUjJTM1IwUzNSMFMzUjJTM1IwczNSMhMzUjBTM1IwczNSMhMzUjBTM1Iy
wDqPxYA2o+Pv5LPj7+Sz4+A2o+Pv5LPj7+Sz4+AbU+PgG1Pj59Pz99Pz/+iT8/fT8/fD4+AbU+PgG
1Pj7+Sz4+/ks+PgNqPj7+Sz4+/ks+PgNqPj59Pz99Pz/+iT8/fT8/fD4+AbU+Piw+Pj8/Pz8/Pj8/
Pz8/Pj8+Pj4+Pj4+Pj4+Pj8+Pz8/Pz8+Pz8/Pz8+Pj4+Pj4+Pj4+Pj4+PgAAAQAAAAAD1APUAAsAA
AEhFSERMxEhNSERIwHh/koBtj4Btv5KPgIfPv5KAbY+AbYAAwAAAAAdgPzAAcAJABIAAABFSE1Mx
EhESUfBxUzFSE1Mz0BPwg7ARcnDwsjESERIy8ODwIBBgH0Pv2QAVUGBQQHBQIDAX3+in0BAWMEBgU
HCQsNEAdHBQYKCGwLBwMHAwIB+gLu+gECAwUFBggMDgoLCwwMDQwNDAM4fX39EwLteQMEBQoLBg4N
Nj8/JxYKCgkIBwCFBAMBNQIDBwCMDgoGEQsNDPyVA2sMDQsMCwoKDasHBQQEAgEBAgMAAAAABgAAA
AAD8wPzAAMAQwBHAiCaiwDLAAALITUhBR8PPw8vDw8OASE1KQEfDz8PLw8PDgEhNSElHw8/Dy8PDw
4BRQKv/VH+xwEBAgQEBAYGBggHCAkJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgo
JCQkIBwGGBgYEBAQCAQE4Aq/9Uf7HAQECBAMFBgYGBwgICQkJCgkKCQgJBwGGBgYEBAMDAQEBAQMD
BAQGBgYIBwkICQoJCgkJCQgIBwYGBgUDBAIBATgCr/1R/scBAQIEAwUGBgYHCagJCQkKCQoJCAkHC
AYGBgQEAWMBAQEBAWMEBAYGBggHCQgJCgkKCQkJCAgHBgYGBQMEAgFLPh8KCQkICQcIBgYGBAQDAw
EBAQEDAwQEBgYGCACJCAkJCgoJCQgJBwGGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQkBTj4KCQkICQc
IBgYGBAQDAwEBAQEDAwQEBgYGCACJCAkJCgoJCQgJBwGGBgYEBAMDAQEBAQMDBAQGBgYIBwkICQk
Lj8fCgkJCAkHCAYGBgQEAWMBAQEBAWMEBAYGBggHCQgJCQoKCQkICQcIBgYGBAQDAwEBAQEDAwQEB
gYGCACJCAkJAAIAAAAAAPzA/MAAwAAHAAAEQAVABkAHQAhAAABFSM1IxUjNSMVIzUTMyEVITUBFS
M1IxUjNSMVIzUDIREhA7X6Pvo++vo+AjL8lgNq+j76Pvo/A+j8GAFF+vr6+vr6ATj6+gE4+vr6+vr
6/FcD6AAABAAAAAAD8wPzAAsADwATABsAAAEEXBxc3FzcnNycHJwERIxehESMRAYEVITUhESEBg3Bw
LHBwLHBwLHBwAgb5/on6PwE5AXYBOfwYARlxcCxwcCxwcSxwcAJw/ooBdv6KAXb+Sz4+AfQAAAAAB
QAAAAAD1APUAAMABwALAA8AEwAAAREhESMRIREBESERIxEhEQMhESEDlv6JPv6JAYz+iT7+iT4DqP
xYAEh+iQF3/okBdwG1/okBd/6JAXf8lgOoAAAAAIAAAAA/MDtQBTAf8AAAEpBRU/BjsBHwkVDxA
VMzUjPxEvDisBCQIXCQE3CQEnCQEDVw4ODQwNDawMDAwNDAM0MBw0MCgkEAWMCAQEGBAYHCREMnw4M
CwoIBgICafq0AQECCAsMniMPDAUEBAQCAGQEBQICBAUFBwCICQoKDAwMDRD8pQEx/s8YASYBJjh+0
AEwMf7a/toB/gMDBQYHCDkKCAgGBAQCAgQFBwUFBQUHBg4MDAsKCg4KKwsMDA4ODggICSU0BgYFCw
sLKRwODwgICQkJCgoLDAsLCgkICAYGBgQEAWIBAZD+cP5xJgGB/n8mAY8Bjyb+fgGCAIAAAAAA/M
DtQADAAgAAAERIREDKQERIQJ9/c4/AnEBd/wYA3f9EgLu/NQDagAAAAGAAAAA/MD8wADAAcACwAP
ABMAFwAbAB8AACUzNSMFITUhJTM1IwUhNSElMzUjBSE1ISUzNSMFITUhA7U/P/xXAYz81AOpPz/8V
wG2/koDqT8//FcCcf2PA6k/P/xXAYz81Aw/Pz/6Pj4++j4+Pvo/Pz8AAQAAAAAC2gPzAAMAACUzAS
MBJUkBBUgMA+gAABsAAAAA9QD1AADAACACwAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMARwB
LAE8AUwBXAFsAXwBjAGcAawAAJTM1IwczNSMhMzUjBzM1IwczNSMhMzUjBzM1IyUzNSMFMzUjBTM1
IyUzNSMFMzUjBTM1IzUhNSElMzUjBTM1IwUzNSMlMzUjBTM1IwUzNSMlMzUjBzM1IwczNSMhMzUjB
zM1IwczNSMhMzUjA5Y+Pn0/P30/P7s+Prw/P30/P3w+PgNqPj7+Sz4+/ks+PgNqPj7+Sz4+/ks+Pg
Oo/FgDaj4+/ks+Pv5LPj4Daj4+/ks+Pv5LPj4Daj4+fT8/fT8/uz4+vD8/fT8/fD4+LD4+Pj4+Pj4
+Pj4+Pj4+Pz8/Pz8+Pz8/Pz99Pn0/Pz8/Pz4/Pz8/Pz4+Pj4+Pj4+Pj4+Pj4+ABwAAAAA9QD1AAD
AAcACwAPABMAFwAbAB8AIwAnACsALwAzADcAOwA/AEMARwBLAE8AUwBXAFsAXwBjAGcAawBvAAALM
zUjBzM1IwczNSMhMzUjBzM1IwczNSMlMzUjBTM1IyUzNSMFMzUjJTM1IwczNSMhMzUjBzM1IwczNS
MhMzUjBzM1IyUzNSMFMzUjJTM1IwUzNSMlMzUjBzM1IwczNSMhMzUjBzM1IwczNSMDMxEjA5Y+Pn0
/P30/P7s+Prw/P30/PwLuPj7+Sz4+AbU+Pv5LPj4BtT4+fT8/fT8/fT8/fT8/fT8/Au4+Pv5L
Pj4BtT4+/ks+PgG1Pj59Pz99Pz+7Pj68Pz99Pz98Pj4sPj4+Pj4+Pj4+Pj4+Pz8/Pj8/P30/Pz4+P
j4+Pj4+Pj4+fT8/Pz4/Pz8+Pj4+Pj4+Pj4+Pj4+Pj78WAOoAAAAAAGAAAAA/MD8wAFAAaAEQAZAB0AIw
AnADMAADcjFTM1IzMHNSkBMxUzNTM1IzcjFTM1IzUjMyE1KQEzFTM1IyUhNSErAUzFSMVMzUjNSO
Jfbw/vAKv/VH+xz8+P7w/P7w/PvoCr/1R/sd9P7wBOQKv/VH6Pz8/vD8+Sz99Pz8/Prw/Pz4+Pn36
Pj4/Pj68AAIAAAAAA/MC+QCHARQAAAEfBzsBHw0dAg8NKwIvDT0BLwcPBxUfDyE/DzUvDyMPBgUVH
w8zPwY9AS8GKwEvDT0CPw07Ah8ZPwcvEyMPDgK7AQIDBAQFBgddDQwMDAsKCgkIBwYFBQMCAGMFBQ
YHCAkKCgsMDAwN+gwNDAsLCgoJCAcGBgQDAgECAwQEBgYHBgYFBQQCAgEBAwUGCAkLDA0PDwgREhI
TAQMUEhIREQ8PDQwLCQQHBgQCAQMFBggJCwwNDw8IERISE2cHBQYEBAMC/VABAwUGCAkLDA0PDwgR
EhITZwCFBgQEAWICAwQEBgUHXQ0NDAsLCgoJCAcGBgQDAgIDBAYGBwgJCgoLCwwNDfkKCQkJCAkIC
AcHBgYGBQUEBAMCAQIDBAQFBgCGBgUFAwMCAQEDBQYGBwCJCQoKCwwMDA0NDg40+RMTEhERDw8NDA

```

SJCAYFAwLbBwUGBAQDAGECAwQGBgcICQoKCwsMDQx9DQ0MFCwsKCGkIBwYGBAMCAGMEBgYHCakKCgs
LDA0NRQcFbBgQEAWIBAQIDBAQGBQdFFBISEREPDw0MCwkEBwYEAgEDBQYICQsMDQ8PCBESEhOGFBIS
EREPDw0MCwkEBwYEAgECAGQFBQaifRQSEhERDw8NDAsJBACGBAIBAgIEBQUGBgcFbBgQEAWIBAgMEB
gYHCAkKCgsLDA0MfQ0MDAwLCgoJCACGBQUdAgECAGMEBAUFbBgYHBwcJCAwMDBMGBQUeAgIBAQICBA
UFBgYtEXIRDQwMCwoJCQgHBgUFaWMBaQEDBQYICQsMDQ8PERESEgAAAAQAAAAAA/MD8wADAAcACwA
PAAA3ITUhJyE1ITChNSEnITUhQAKw/VCCa+j8GJwCsPlQnAPo/BgMP/o++j76PwAFAAAAAAAPzA/MA
AwAHAAAsAGwAnAAABFSM1IxUjNSMVIzUDMzUzFTM1MxUzNTMVMxEhJSMVMxUzNTM1IzUjA7X6Pvo++
j8/+j76Pvo//BgB9H19Pn19PgI++fn5+fn5/c76+vr6+voCcfo/fX0/fQAAAAASAN4AAQAAAAAAA
ABAAAAQAAAAAAQAZAAEAAQAAAAAAAgAHABoAAQAAAAAAAwAZACEAAQAAAAAABAAZADoAAQAAAAA
ABQALAFMAAQAAAAABgAZAF4AAQAAAAAAACgAsAHCaAQAAAAAAACwASAKMAAwABBakAAAAACALUAaWAB
BAkAAQAYaLcAAwABBakAAgAOAOkAAwABBakAAwAyAPcAAwABBakABAAYaSKAAwABBakABQAWAVsAA
wABBakABgAYAXEAAwABBakACgBYAaMAAwABBakACwAkAfsgrmluYWwgU2FtcGx1IGJyb3dlciBpY2
9uc1JlZ3VsYXJGaW5hbCBTYWlwYUgYUjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2Vj
uc1ZlcnNpb24gMS4wRmluYWwgU2FtcGx1IGJyb3dlciBpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5n
IFN5bmdNmdXNpb24gT2VW0cm8gU3RlZGlvd3d3LnN5bmdNmdXNpb24uY29tACAARgBpAG4AYQBsACAAU
wBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAEYAaQBuAG
EAbAAgAFMAYQBTaHAAbAB1ACAAyGByAG8AdwB1AHIAIABpAGMAbwBuAHMARgBpAG4AYQBsACAAUwB
hAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBWAGUAUcBzAGkAbwBuACAAMQAuADAA
RgBpAG4AYQBsACAAUwBhAG0AcABsAGUAIABiAHIAbwB3AGUAcgAgAGkAYwBvAG4AcwBGAG8AbgB0A
CAAzwB1AG4AZQByAGEAdABLAGQAIABlAHMAaQBuAGCAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQ
BlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcAlGbzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0
AAAAAaQAAAAAAAKAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAa
AAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAa
DAENAQ4BDwEQAREBEGeTARQBFQEWARcBGAEZARoBGwEcAR0BHGEfASABIQEiASMBJAELASyBjWEOA
SkBKgErASwBLQEuAS8BMAExATIBMWEOATUBNGE3ATGBOQE6ATsBPAE9AT4BPwFAAAATdHJva2VtdH
lsZQhCb29rbWfYawdQaWN0dXJlBEZpbmQNT3V0c2lkZUJvcmlRlZGdKdXN0aWZ5BUNsb3N1DkRlY3J
lYXNlSW5kZW50FVBpeGVsQWxpZ25DZW50ZXJUYWJsZQ9CYWNrZ3JvdW5kQ29sb3JlQWxpZ25Cb3R0
b20JUGFnZVNldHVWdDkhpZ2hsaWdodENvbG9yY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2VjY2
mVsb3cJVG9wQm9yZGVyClBhZ2V0dWliZXIQQWxpZ25DZW50ZXJUYWJsZQ5JbmNyZWZzZUluZGVudA
RCb2xkCUFsaWduTGVMdAZGb290ZXILSW5zZXJ0UmlnaHQJVW5kZXJsaW51Ckluc2VydExlZnQETG9
jawZIZWfKZXINU3RyaWtldGhyb3VnaAhDbGVhckFsbAtSaWdodEJvcmlRlZGpBbGlub1JpZ2h0BE9w
ZW4KU3Ryb2t1U2l6ZQVQcm1udAtEZWxldGVUYWJsZQ1Gb250Q29sb3INSW5zaWRlQm9yZGVyY2VjY2Vj
WxldGVVsb3dzCERvd25sb2FkC0xpbmVtCGFjaW5nFELuc2lkZVZlcnRyY2FsQm9yZGVyY2VjY2VjY2Vj
9wBFJlZG8MQm90dG9tQm9yZGVyA05ldwVQYXN0ZQdCdWxsZXRxZBENlbGwNRGVsZXRLQ29sdWlucwp
BbGxCh3JkZXJzCVNlYnNjcm1wdBBTaG93SGlkZVByb3BlcnR5DlRhYmxlT2ZDb250ZW50Bk10YXxp
YXZJbnNpZGVib3Jpem9uZGFsYm9yZGVyC0xlZnRCb3JkZXJzCU51bWJlcm1uZwRMaW5rC0FsaWduQ
2VudGVyC0luc2VydEFib3JlZAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAaQAAAAAa
format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="e-de-icon-"],
[class*=" e-de-icon-"] {
font-family: 'Sample browser icons';
}
.e-de-icon-Bullets:before {
content: "\e730";
}
.e-de-icon-Numbering:before {
content: "\e73a";
}
}
</style>

```

Read by default in React Document editor component

In this article, we are going to see how to open a document in read only mode by default in Document Editor & Document Editor Container.

Opening a document in read only mode by default in DocumentEditor

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog } from '@syncfusion/ej2-react-
documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
function App() {
  let documentEditor;
  React.useEffect(() => {
    componentDidMount();
  }, []);
  function onLoadDefault() {
    // load your default document here
    let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":
72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"te
xt":"Adventure Works
Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFo
rmat":{},"characterFormat":{},"inlines":[]]}]},"footer":{"blocks":[{"paragrap
hFormat":{"listFormat":{},"characterFormat":{},"inlines":[]]}]}]},"character
Format":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","un
derline":"None","strikethrough":"None","baselineAlignment":"Normal","highligh
tColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Cal
ibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"li
neSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"b
idi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"ha
shValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection"
,"dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"
name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.14999997
6158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fon
tFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph
Font","type":"Character","characterFormat":{}}, {"name":"Heading 1
Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,
```



```

"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"},{"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"},{"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"},{"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"},{"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4 Char","next":"Normal"},{"name":"Heading 4 Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level5","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5 Char","next":"Normal"},{"name":"Heading 5 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level6","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 6 Char","next":"Normal"},{"name":"Heading 6 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"}],{"lists":[],"abstractLists":[],"comments":[],"revisions":[],"customXml":[]}`;

    // Open the default document
    documentEditor.open(data);
    //Enable read only mode.
    documentEditor.isReadOnly = true;
  }
  function componentDidMount() {
    setTimeout(() => {
      onLoadDefault();
    });
  }

```

```

    });
  }
  return (<DocumentEditorComponent id="container" ref={scope} => {
    documentEditor = scope; })
  serviceUrl="https://ej2services.syncfusion.com/production/web-
  services/api/documenteditor/" isReadOnly={false} enablePrint={true}
  enableSelection={true} enableEditor={true} enableEditorHistory={true}
  enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
  enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
  enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
  enableHyperlinkDialog={true} enableImageResizer={true}
  enableListDialog={true} enablePageSetupDialog={true} enableSfdtExport={true}
  enableStyleDialog={true} enableTableOfContentsDialog={true}
  enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
  enableTextExport={true} enableWordExport={true} height={'330px'}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, Print, SfdtExport, WordExport, TextExport,
  Selection, Search, Editor, ImageResizer, EditorHistory,
  ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
  TableOfContentsDialog,
  PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
  BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
  CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';
//Inject require modules.
DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport,
Selection, Search, Editor, ImageResizer, EditorHistory, ContextMenu,
OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog,
TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog,
ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog,
CellOptionsDialog, StylesDialog);
function App() {
  let documentEditor: DocumentEditorComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onLoadDefault() {
    // load your default document here
    let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":
72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading

```



```

1", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat": {}, "text": "Adventure Works Cycles"}]}, "headersFooters": {"header": {"blocks": [{"paragraphFormat": {"listFormat": {}, "characterFormat": {}, "inlines": []}}, "footer": {"blocks": [{"paragraphFormat": {"listFormat": {}, "characterFormat": {}, "inlines": []}}]}}, "characterFormat": {"bold": false, "italic": false, "fontSize": 11, "fontFamily": "Calibri", "underline": "None", "strikethrough": "None", "baselineAlignment": "Normal", "highlightColor": "NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi": "Calibri", "allCaps": false}, "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listFormat": {}, "bidi": false}, "defaultTabWidth": 36, "trackChanges": false, "enforcement": false, "hashValue": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection", "dontUseHTMLParagraphAutoSpacing": false, "formFieldShading": true, "styles": [{"name": "Normal", "type": "Paragraph", "paragraphFormat": {"lineSpacing": 1.149999976158142, "lineSpacingType": "Multiple", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri"}, "next": "Normal"}, {"name": "Default Paragraph Font", "type": "Character", "characterFormat": {}}, {"name": "Heading 1 Char", "type": "Character", "characterFormat": {"fontSize": 16, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 0, "outlineLevel": "Level1", "listFormat": {}, "characterFormat": {"fontSize": 16, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 1 Char", "next": "Normal"}, {"name": "Heading 2 Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6, "outlineLevel": "Level2", "listFormat": {}, "characterFormat": {"fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2 Char", "next": "Normal"}, {"name": "Heading 3 Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal"}, {"name": "Heading 3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {}, "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal"}, {"name": "Heading 4 Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}, "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal"}, {"name": "Heading 5 Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5

```

```

Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml
": []}`;

    // Open the default document
    documentEditor.open(data)
    //Enable read only mode.
    documentEditor.isReadOnly = true;
  }
  function componentDidMount() {
    setTimeout(() => {
      onLoadDefault();
    });
  }
  return (
    <DocumentEditorComponent id="container" ref={(scope) => {
documentEditor = scope; }}
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" isReadOnly={false} enablePrint={true}
enableSelection={true} enableEditor={true}
enableEditorHistory={true}
enableContextMenu={true} enableSearch={true}
enableOptionsPane={true}
enableBookmarkDialog={true} enableBordersAndShadingDialog={true}
enableFontDialog={true} enableTableDialog={true} enableParagraphDialog={true}
enableHyperlinkDialog={true} enableImageResizer={true}
enableListDialog={true}
enablePageSetupDialog={true} enableSfddtExport={true}
enableStyleDialog={true} enableTableOfContentsDialog={true}
enableTableOptionsDialog={true}
enableTablePropertiesDialog={true}
enableTextExport={true} enableWordExport={true} height={'330px'}
/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />

```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<meta name="description" content="Essential JS 2 for React Components" />
<meta name="author" content="Syncfusion" />
<link href="index.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
    <div id='sample'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

Opening a document in ready only mode by default in DocumentEditorContainer

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-
react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
    let documenteditorcontainer;
    React.useEffect(() => {
        componentDidMount();
    }, []);
    function onLoadDefault() {
        // load your default document here
    }
}

```

```

let data =
`{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":false,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"Heading 1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"text":"Adventure Works Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"footer":{"blocks":[{"paragraphFormat":{"listFormat":{},"characterFormat":{},"inlines":[]}}]},"characterFormat":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","underline":false,"strikethrough":false,"baselineAlignment":"Normal","highlightColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Calibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"bidi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"hashValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection","dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.149999976158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fontFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph Font","type":"Character","characterFormat":{}},{name":"Heading 1 Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1 Char","next":"Normal"},{"name":"Heading 2 Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2 Char","next":"Normal"},{"name":"Heading 3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3 Char","next":"Normal"},{"name":"Heading 3 Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"},{"name":"Heading 4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4 Char","next":"Normal"},{"name":"Heading 4 Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph

```

```
Font"}, {"name": "Heading
5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml": []} `;

// Open the default document
documenteditorcontainer.documentEditor.open(data);
//Enable read only mode.
documenteditorcontainer.restrictEditing = true;
}
function componentDidMount() {
  setTimeout(() => {
    onLoadDefault();
  });
}
return (<DocumentEditorContainerComponent id="container" ref={ (scope) =>
{ documenteditorcontainer = scope; onLoadDefault(); }}
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" height={'590px'} enableToolbar={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
[% endraw %]
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let documenteditorcontainer: DocumentEditorContainerComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onLoadDefault() {
    // load your default document here
    let data =
`{"sections": [{"sectionFormat": {"pageWidth": 612, "pageHeight": 792, "leftMargin"
```

```

:72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false,"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"te
xt":"Adventure Works
Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFo
rmat":{},"characterFormat":{},"inlines":[]]}]},"footer":{"blocks":[{"paragrap
hFormat":{"listFormat":{},"characterFormat":{},"inlines":[]]}]},"character
Format":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","un
derline":"None","strikethrough":"None","baselineAlignment":"Normal","highligh
tColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Cal
ibri","allCaps":false,"paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"li
neSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"b
idi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"ha
shValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection"
,"dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"
name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.14999997
6158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fon
tFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph
Font","type":"Character","characterFormat":{}}, {"name":"Heading 1
Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,
"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fo
ntFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1
Char","next":"Normal"}, {"name":"Heading 2
Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
2","type":"Paragraph","paragraphFormat":{"beforeSpacing":2,"afterSpacing":6,"
outlineLevel":"Level2","listFormat":{},"characterFormat":{"fontSize":13,"fon
tFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 2
Char","next":"Normal"}, {"name":"Heading
3","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firs
tLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lin
eSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Lev
el3","listFormat":{},"characterFormat":{"fontSize":12,"fontFamily":"Calibri
Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 3
Char","next":"Normal"}, {"name":"Heading 3
Char","type":"Character","characterFormat":{"fontSize":12,"fontFamily":"Calib
ri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
4","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firs
tLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lin
eSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Lev
el4","listFormat":{},"characterFormat":{"italic":true,"fontFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 4
Char","next":"Normal"}, {"name":"Heading 4
Char","type":"Character","characterFormat":{"italic":true,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
5","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firs

```

```

tLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level5","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 5 Char","next":"Normal"},{"name":"Heading 5 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph Font"},{"name":"Heading 6","type":"Paragraph","paragraphFormat":{"leftIndent":0,"rightIndent":0,"firstLineIndent":0,"textAlignment":"Left","beforeSpacing":2,"afterSpacing":0,"lineSpacing":1.0791666507720947,"lineSpacingType":"Multiple","outlineLevel":"Level6","listFormat":{},"characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Normal","link":"Heading 6 Char","next":"Normal"},{"name":"Heading 6 Char","type":"Character","characterFormat":{"fontFamily":"Calibri Light","fontColor":"#1F3763"},"basedOn":"Default Paragraph Font"}],"lists":[],"abstractLists":[],"comments":[],"revisions":[],"customXml":[]}]`
    // Open the default document
    documenteditorcontainer.documentEditor.open(data);
    //Enable read only mode.
    documenteditorcontainer.restrictEditing = true;
  }
  function componentDidMount() {
    setTimeout(() => {
      onLoadDefault();
    });
  }
  return (<DocumentEditorContainerComponent id="container" ref={ (scope) =>
  { documenteditorcontainer = scope; onLoadDefault(); }}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/" height={'590px'} enableToolbar={true} />);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-base/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='sample'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Note: You can use the `restrictEditing` in `DocumentEditorContainerComponent` and `isReadOnly` in `DocumentEditorComponent` based on your requirement to change component to read only mode.

Open document by address in React Document editor component

[How to open a document from URL in DocumentEditor](#)

In this article, we are going to see how to open a document from URL in DocumentEditor

please refer below example for client-side code

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent, Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let contentChanged: boolean = false;
  function onClick() {
    let http: XMLHttpRequest = new XMLHttpRequest();

```



```
//add your url in which you want to open document inside the ""
let content = { fileUrl: "" };
let baseUrl: string = "/api/documenteditor/ImportFileURL";
http.open("POST", baseUrl, true);
http.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
http.onreadystatechange = () => {
  if (http.readyState === 4) {
    if (http.status === 200 || http.status === 304) {
      //open the SFDT text in Document Editor
      container.documentEditor.open(http.responseText);
    }
  }
};
http.send(JSON.stringify(content));
}
return (
  <div>
    <button id='import' onClick={onClick}>Import</button>
    <DocumentEditorContainerComponent id="container" ref={{scope} => { container = scope; }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
    />
  </div>
);
}
```

export default App;

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

please refer below example for server-side code

```
`csharp
[AcceptVerbs("Post")]
public string ImportFileURL([FromBody]FileInfo param)
```

```

{
try {
using(WebClient client = new WebClient())
{
MemoryStream stream = new MemoryStream(client.DownloadData(param.fileUrl));
WordDocument document = WordDocument.Load(stream, FormatType.Docx);
string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
document.Dispose();
stream.Dispose();
return json;
}
}
catch (Exception) {
return "";
}
}

public class FileUrlInfo {
public string fileUrl { get; set; }
public string Content { get; set; }
}
`

```

Deploy document editor component for mobile in React Document editor component *Document editor component for Mobile*

At present, Document editor component is not responsive for mobile, and we haven't ensured the editing functionalities in mobile browsers. Whereas it works properly as a document viewer in mobile browsers.

Hence, it is recommended to switch the Document editor component as read-only in mobile browsers. Also, invoke [fitPage](#) method with `FitPageWidth` parameter in document change event, such as to display one full page by adjusting the zoom factor.

The following example code illustrates how to deploy Document Editor component for Mobile.

```

{% raw %}
`ts
//Initialize Document Editor Container component.
import { DocumentEditorContainer, Toolbar, DocumentEditorContainerComponent } from
'@syncfusion/ej2-react-documenteditor';

```

```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let hosturl = 'https://ej2services.syncfusion.com/production/web-services/api/documenteditor/';
  function onDocumentChange() {
    let proxy = container;
    //To detect the device
    let isMobileDevice = /Android|Windows Phone|webOS/i.test(navigator.userAgent);
    if (isMobileDevice) {
      proxy.restrictEditing = true;
      setTimeout(() => {
        proxy.documentEditor.fitPage("FitPageWidth");
      }, 50);
    }
    else {
      proxy.restrictEditing = false;
    }
  }
  return (
    <div className="App">
      <DocumentEditorContainerComponent id="container" ref={{scope} => { container = scope; }} style={{
        'height': '590px' }} enableToolbar={true} documentChange={onDocumentChange} serviceUrl={hosturl}
        height={'590px'} />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
{% enddraw %}

```

You can download the complete working example from this [GitHub location](#)

Note: You can use the [restrictEditing](#) in `DocumentEditorContainer` and [isReadOnly](#) in `DocumentEditor` based on your requirement to change component to read only mode.

Disable optimized text measuring in React Document editor component

Starting from v19.3.0.x, the accuracy of text size measurements in Document editor is improved such as to match Microsoft Word pagination for most Word documents. This improvement is included as default behavior along with an optional API [enableOptimizedTextMeasuring](#) in Document editor settings.

If you want the Document editor component to retain the document pagination (display page-by-page) behavior like v19.2.0.x and older versions. Then you can disable this optimized text measuring improvement, by setting `false` to [enableOptimizedTextMeasuring](#) property of React Document Editor component.

Disable optimized text measuring in `DocumentEditorContainer` instance

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditorContainer` instance.

```
{% raw %}
`ts
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  // Disable optimized text measuring improvement
  let settings = { enableOptimizedTextMeasuring: false };
  return (
    <DocumentEditorContainerComponent
      id="container"
      style={{ height: '590px' }}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      documentEditorSettings={settings}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
{% endraw %}
```

Disable optimized text measuring in `DocumentEditor` instance

The following example code illustrates how to disable optimized text measuring improvement in `DocumentEditor` instance.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorComponent, DocumentEditor, RequestNavigateEventArgs, ViewChangeEventArgs,
  Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor, ImageResizer, EditorHistory,
  ContextMenu, OptionsPane, HyperlinkDialog, TableDialog, BookmarkDialog, TableOfContentsDialog,
  PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog, BulletsAndNumberingDialog, FontDialog,
  TablePropertiesDialog, BordersAndShadingDialog, TableOptionsDialog, CellOptionsDialog, StylesDialog
} from '@syncfusion/ej2-react-documenteditor';

DocumentEditorComponent.Inject(Print, SfdtExport, WordExport, TextExport, Selection, Search, Editor,
ImageResizer, EditorHistory, ContextMenu, OptionsPane, HyperlinkDialog, TableDialog,
BookmarkDialog, TableOfContentsDialog, PageSetupDialog, StyleDialog, ListDialog, ParagraphDialog,
BulletsAndNumberingDialog, FontDialog, TablePropertiesDialog, BordersAndShadingDialog,
TableOptionsDialog, CellOptionsDialog, StylesDialog);

function App() {
  // Disable optimized text measuring improvement
  let settings = { enableOptimizedTextMeasuring: false };

  return (
    <DocumentEditorComponent id="container" height={'330px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      isReadOnly={false} enablePrint={true}

      enableSelection={true} enableEditor={true} enableEditorHistory={true}
      enableContextMenu={true} enableSearch={true} enableOptionsPane={true}
      enableBookmarkDialog={true} enableBordersAndShadingDialog={true} enableFontDialog={true}
      enableTableDialog={true} enableParagraphDialog={true} enableHyperlinkDialog={true}
      enableImageResizer={true} enableListDialog={true}
      enablePageSetupDialog={true} enableSfdtExport={true}
      enableStyleDialog={true} enableTableOfContentsDialog={true}
      enableTableOptionsDialog={true} enableTablePropertiesDialog={true}
      enableTextExport={true} enableWordExport={true} documentEditorSettings={settings} />
  );
}
```

```
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Get the selected content in React Document editor component

You can get the selected content from the React Document Editor component as plain text and SFDT (rich text).

Get the selected content as plain text

You can use [text](#) API to get the selected content as plain text from React Document Editor component.

The following example code illustrates how to add search in google option in context menu for the selected text.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
import { MenuItemModel } from '@syncfusion/ej2-navigations';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let contentChanged: boolean = false;
  function onCreate() {
    // creating Custom Options
    let menuItems: MenuItemModel[] = [
      {
        text: 'Search In Google',
        id: 'searchingoogle',
        iconCss: 'e-icons e-de-ctnr-find',
      },
    ];
    // adding Custom Options
    container.documentEditor.contextMenu.addCustomMenu(menuItems, false);
    // custom Options Select Event
```

```
container.documentEditor.customContextMenuSelect = (
  args: any
): void => {
  // custom Options Functionality
  let id: string = container.documentEditor.element.id;
  switch (args.id) {
    case id + 'searchingoogle':
      // To get the selected content as plain text
      let searchContent: string =
        container.documentEditor.selection.text;
      if (
        !container.documentEditor.selection.isEmpty &&
        /\S/.test(searchContent)
      ) {
        window.open('http://google.com/search?q=' + searchContent);
      }
      break;
    }
  };
}

return (
  <DocumentEditorContainerComponent
    id="container"
    ref={({scope}) => {
      container = scope;
    }}
    height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
    enableToolbar={true}
    created={onCreate}
  />
);
}
```

```
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

You can add the following custom options using this API,

- Save or export the selected text as text file.
- Search the selected text in Google or other search engines.
- Show synonyms for the selected word in context menu and replace with selected synonym using the setter method of same API.

Get the selected content as SFDT (rich text)

You can use [sfdt](#) API to get the selected content as plain text from React Document Editor component.

The following example code illustrates how to get the content of a bookmark and export it as SFDT.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let contentChanged: boolean = false;
  function onCreate() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select all the content in document
    container.documentEditor.selection.selectAll();
    // Insert bookmark to selected content
    container.documentEditor.editor.insertBookmark('Bookmark1');
    //Select the bookmark
    container.documentEditor.selection.selectBookmark('Bookmark1');
    // To get the selected content as sfdt
    let selectedContent: string = container.documentEditor.selection.sfdt;
    // Insert the sfdt content in cursor position using paste API
```



```

container.documentEditor.editor.paste(selectedContent);
}
return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

You can add the following custom options using this API,

- Save or export the selected content as SFDT file.
- Get the content of a bookmark in Word document as SFDT by selecting a bookmark using [selectbookmark](#) API.
- Create template content that can be inserted to multiple documents in cursor position using [paste](#) API.

Set default format in document editor in React Document editor component

You can set the default character format, paragraph format and section format in Document editor.

Set the default character format

You can use [setDefaultCharacterFormat](#) method to set the default character format. For example, Document editor default font size is 11 and you can change it as any valid value.

The following example code illustrates how to change the default font size in Document editor.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {

```

```

DocumentEditorContainerComponent,
Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    container.documentEditor.setDefaultCharacterFormat({ fontSize: 20 });
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreate}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Similarly, you can change the required [CharacterFormatProperties](#) default value.

The following example code illustrates how to change other character format default value in Document editor.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,

```

```
Toolbar,CharacterFormatProperties
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    let defaultCharacterFormat:CharacterFormatProperties = {
      bold: false,
      italic: false,
      baselineAlignment: 'Normal',
      underline: 'None',
      fontColor: "#000000",
      fontFamily: 'Algerian',
      fontSize: 12
    };
    container.documentEditor.setDefaultCharacterFormat(defaultCharacterFormat);
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={({scope}) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreate}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
```

Set the default paragraph format

You can use [setDefaultParagraphFormat](#) API to set the default paragraph format. You can change the required [ParagraphFormatProperties](#) default value.

The following example code illustrates how to change the paragraph format(before spacing, line spacing etc.,) default value in Document editor.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar, ParagraphFormatProperties
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    let defaultParagraphFormat: ParagraphFormatProperties = {
      beforeSpacing: 8,
      lineSpacing: 1.5,
      leftIndent: 24,
      textAlignment: "Center"
    };
    container.documentEditor.setDefaultParagraphFormat(defaultParagraphFormat);
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreate}
```

```

/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Set the default section format

You can use [setDefaultSectionFormat](#) API to set the default section format. You can change the required [SectionFormatProperties](#) default value.

The following example code illustrates how to change the section format(header and footer distance, page width and height, etc.,) default value in Document editor.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar, SectionFormatProperties
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    let defaultSectionFormat: SectionFormatProperties = {
      pageWidth: 500,
      pageHeight: 800,
      headerDistance: 56,
      footerDistance: 48,
      leftMargin: 12,
      rightMargin: 12,
      topMargin: 0,
      bottomMargin: 0
    };
    container.documentEditor.setDefaultSectionFormat(defaultSectionFormat);
  }
}

```

```

return (
<DocumentEditorContainerComponent
id="container"
ref={{(scope) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Show hide spinner in React Document editor component

Using [spinner](#) component, you can show/hide spinner while opening document in DocumentEditor .

Example code snippet to show/hide spinner

```

`ts
// showSpinner() will make the spinner visible
showSpinner(document.getElementById('container'));

// hideSpinner() method used hide spinner
hideSpinner(document.getElementById('container'));
`

```

Refer to the following example.

INDEX.JSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
import { showSpinner, hideSpinner, createSpinner } from '@syncfusion/ej2-popups';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container;
  React.useEffect(() => {

```

```

    componentDidMount();
  }, []);
  function componentDidMount() {
    createSpinner({
      // Specify the target for the spinner to show
      target: document.getElementById('container')
    });
  }
  function onClick() {
    // load your default document here
    let data =
'{"sections":[{"sectionFormat":{"pageWidth":612, "pageHeight":792, "leftMargin":
72, "rightMargin":72, "topMargin":72, "bottomMargin":72, "differentFirstPage":fa
lse, "differentOddAndEvenPages":false, "headerDistance":36, "footerDistance":36,
"bidi":false}, "blocks":[{"paragraphFormat":{"afterSpacing":30, "styleName": "He
ading
1", "listFormat": {}, "characterFormat": {}, "inlines": [{"characterFormat": {}, "te
xt": "Adventure Works
Cycles"}]}], "headersFooters": {"header": {"blocks": [{"paragraphFormat": {"listFo
rmat": {}, "characterFormat": {}, "inlines": []}]}, "footer": {"blocks": [{"paragrap
hFormat": {"listFormat": {}, "characterFormat": {}, "inlines": []}]}}], "character
Format": {"bold": false, "italic": false, "fontSize": 11, "fontFamily": "Calibri", "un
derline": "None", "strikethrough": "None", "baselineAlignment": "Normal", "highligh
tColor": "NoColor", "fontColor": "empty", "fontSizeBidi": 11, "fontFamilyBidi": "Cal
ibri", "allCaps": false}, "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "fir
stLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 0, "afterSpacing": 0, "li
neSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "listFormat": {}, "b
idi": false}, "defaultTabWidth": 36, "trackChanges": false, "enforcement": false, "ha
shValue": "", "saltValue": "", "formatting": false, "protectionType": "NoProtection"
, "dontUseHTMLParagraphAutoSpacing": false, "formFieldShading": true, "styles": [{"
name": "Normal", "type": "Paragraph", "paragraphFormat": {"lineSpacing": 1.14999997
6158142, "lineSpacingType": "Multiple", "listFormat": {}, "characterFormat": {"fon
tFamily": "Calibri"}, "next": "Normal"}, {"name": "Default Paragraph
Font", "type": "Character", "characterFormat": {}}, {"name": "Heading 1
Char", "type": "Character", "characterFormat": {"fontSize": 16, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
1", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 12, "afterSpacing": 0,
"outlineLevel": "Level1", "listFormat": {}}, "characterFormat": {"fontSize": 16, "fo
ntFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 1
Char", "next": "Normal"}, {"name": "Heading 2
Char", "type": "Character", "characterFormat": {"fontSize": 13, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6, "
outlineLevel": "Level2", "listFormat": {}}, "characterFormat": {"fontSize": 13, "fon
tFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2
Char", "next": "Normal"}, {"name": "Heading
3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el3", "listFormat": {}}, "characterFormat": {"fontSize": 12, "fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3
Char", "next": "Normal"}, {"name": "Heading 3
Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calib

```

```

ri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el4", "listFormat": {}}, "characterFormat": {"italic": true, "fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4
Char", "next": "Normal"}, {"name": "Heading 4
Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Calib
ri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el5", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5
Char", "next": "Normal"}, {"name": "Heading 5
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph
Font"}, {"name": "Heading
6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firs
tLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lin
eSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Lev
el6", "listFormat": {}}, "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6
Char", "next": "Normal"}, {"name": "Heading 6
Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri
Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph
Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml
": []}';

// showSpinner() will make the spinner visible
showSpinner(document.getElementById('container'));
// Open the default document
container.documentEditor.open(data);
setInterval(function () {
    // hideSpinner() method used hide spinner
    hideSpinner(document.getElementById('container'));
}, 5000);
}
return (<div>
    <button id='import' onClick={onClick}>Load Document</button>
    <DocumentEditorContainerComponent id="container" ref={ (scope) =>
{
    container = scope;
    }} height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true}/>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.TSX

```
{% raw %}
```



```

import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
    DocumentEditorContainerComponent,
    Toolbar, SectionFormatProperties
} from '@syncfusion/ej2-react-documenteditor';
import { showSpinner, hideSpinner, createSpinner } from '@syncfusion/ej2-
popups';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
    let container: DocumentEditorContainerComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function componentDidMount() {
        createSpinner({
            // Specify the target for the spinner to show
            target: document.getElementById('container')
        });
    }
    function onClick() {
        // load your default document here
        let data: string =
'{"sections":[{"sectionFormat":{"pageWidth":612,"pageHeight":792,"leftMargin":
:72,"rightMargin":72,"topMargin":72,"bottomMargin":72,"differentFirstPage":fa
lse,"differentOddAndEvenPages":false,"headerDistance":36,"footerDistance":36,
"bidi":false},"blocks":[{"paragraphFormat":{"afterSpacing":30,"styleName":"He
ading
1","listFormat":{},"characterFormat":{},"inlines":[{"characterFormat":{},"te
xt":"Adventure Works
Cycles"}]}]},"headersFooters":{"header":{"blocks":[{"paragraphFormat":{"listFo
rmat":{},"characterFormat":{},"inlines":[]]}],"footer":{"blocks":[{"paragrap
hFormat":{"listFormat":{},"characterFormat":{},"inlines":[]]}]}]},"character
Format":{"bold":false,"italic":false,"fontSize":11,"fontFamily":"Calibri","un
derline":"None","strikethrough":"None","baselineAlignment":"Normal","highligh
tColor":"NoColor","fontColor":"empty","fontSizeBidi":11,"fontFamilyBidi":"Cal
ibri","allCaps":false},"paragraphFormat":{"leftIndent":0,"rightIndent":0,"fir
stLineIndent":0,"textAlignment":"Left","beforeSpacing":0,"afterSpacing":0,"li
neSpacing":1.0791666507720947,"lineSpacingType":"Multiple","listFormat":{},"b
idi":false},"defaultTabWidth":36,"trackChanges":false,"enforcement":false,"ha
shValue":"","saltValue":"","formatting":false,"protectionType":"NoProtection"
,"dontUseHTMLParagraphAutoSpacing":false,"formFieldShading":true,"styles":[{"
name":"Normal","type":"Paragraph","paragraphFormat":{"lineSpacing":1.14999997
6158142,"lineSpacingType":"Multiple","listFormat":{},"characterFormat":{"fon
tFamily":"Calibri"},"next":"Normal"},{"name":"Default Paragraph
Font","type":"Character","characterFormat":{}}, {"name":"Heading 1
Char","type":"Character","characterFormat":{"fontSize":16,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading
1","type":"Paragraph","paragraphFormat":{"beforeSpacing":12,"afterSpacing":0,
"outlineLevel":"Level1","listFormat":{},"characterFormat":{"fontSize":16,"fo
ntFamily":"Calibri
Light","fontColor":"#2F5496"},"basedOn":"Normal","link":"Heading 1
Char","next":"Normal"}, {"name":"Heading 2
Char","type":"Character","characterFormat":{"fontSize":13,"fontFamily":"Calib
ri Light","fontColor":"#2F5496"},"basedOn":"Default Paragraph
Font"}, {"name":"Heading

```

```

2", "type": "Paragraph", "paragraphFormat": {"beforeSpacing": 2, "afterSpacing": 6, "outlineLevel": "Level2", "listFormat": {}, "characterFormat": {"fontSize": 13, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 2 Char", "next": "Normal"}, {"name": "Heading 3", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level3", "listFormat": {}, "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 3 Char", "next": "Normal"}, {"name": "Heading 3 Char", "type": "Character", "characterFormat": {"fontSize": 12, "fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 4", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level4", "listFormat": {}, "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 4 Char", "next": "Normal"}, {"name": "Heading 4 Char", "type": "Character", "characterFormat": {"italic": true, "fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 5", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level5", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Normal", "link": "Heading 5 Char", "next": "Normal"}, {"name": "Heading 5 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#2F5496"}, "basedOn": "Default Paragraph Font"}, {"name": "Heading 6", "type": "Paragraph", "paragraphFormat": {"leftIndent": 0, "rightIndent": 0, "firstLineIndent": 0, "textAlignment": "Left", "beforeSpacing": 2, "afterSpacing": 0, "lineSpacing": 1.0791666507720947, "lineSpacingType": "Multiple", "outlineLevel": "Level6", "listFormat": {}, "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Normal", "link": "Heading 6 Char", "next": "Normal"}, {"name": "Heading 6 Char", "type": "Character", "characterFormat": {"fontFamily": "Calibri Light", "fontColor": "#1F3763"}, "basedOn": "Default Paragraph Font"}], "lists": [], "abstractLists": [], "comments": [], "revisions": [], "customXml": []}';

// showSpinner() will make the spinner visible
showSpinner(document.getElementById('container'));
// Open the default document
container.documentEditor.open(data);
setInterval(function () {
    // hideSpinner() method used hide spinner
    hideSpinner(document.getElementById('container'));
}, 5000);
}
return (
    <div>
        <button id='import' onClick={onClick}>Load Document</button>
        <DocumentEditorContainerComponent
            id="container"
            ref={ (scope) => {

```

```

        container = scope;
    }}
    height={'590px'}

    serviceUrl="https://ej2services.syncfusion.com/production/web-
    services/api/documenteditor/"
    enableToolbar={true}
    />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
    documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
    popups/styles/fabric.css" rel="stylesheet" />
    <link
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    rel="stylesheet" />
    <script
    src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
    ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='root'>

```

```
        <div id='loader'>Loading....</div>
      </div>
</body>
</html>
```

Note: In above example, we have used `setInterval` to hide spinner, just for demo purpose.

Resize document editor in [React Document editor component](#)

In this article, we are going to see how to change height and width of Documenteditor.

Change height of Document Editor

DocumentEditorContainer initially render with default height. You can change height of documenteditor using [height](#) property, the value which is in pixel.

The following example code illustrates how to change height of Document Editor.

```
`ts
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
this.container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
/>
`
```

Similarly, you can use [height](#) property for DocumentEditor also.

Change width of Document Editor

DocumentEditorContainer initially render with default width. You can change width of documenteditor using [width](#) property, the value which is in percent.

The following example code illustrates how to change width of Document Editor.

```
`ts
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
this.container = scope;
}}
width={'100%'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
```

```
enableToolbar={true}  
/>  
,
```

Similarly, you can use [width](#) property for DocumentEditor also.

Resize Document Editor

Using [resize](#) method, you change height and width of Document editor.

The following example code illustrates how to fit Document Editor to browser window size.

```
`ts  
import * as ReactDOM from 'react-dom';  
import * as React from 'react';  
import {  
  DocumentEditorContainerComponent,  
  Toolbar,  
} from '@syncfusion/ej2-react-documenteditor';  
DocumentEditorContainerComponent.Inject(Toolbar);  
function App() {  
  let container: DocumentEditorContainerComponent;  
  function onCreate() {  
    setInterval(() => {  
      updateDocumentEditorSize();  
    }, 100);  
    //Adds event listener for browser window resize event.  
    window.addEventListener('resize', onWindowResize);  
  }  
  function onWindowResize() {  
    //Resizes the document editor component to fit full browser window automatically whenever the  
    browser resized.  
    updateDocumentEditorSize();  
  }  
  function updateDocumentEditorSize() {  
    //Resizes the document editor component to fit full browser window.  
    var windowWidth = window.innerWidth;  
    var windowHeight = window.innerHeight;  
    container.resize(windowWidth, windowHeight);  
  }  
}
```

```

}
return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreate}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

Export document as pdf in React Document editor component

In this article, we are going to see how to export the document as Pdf format. You can export the document as Pdf in following ways:

Export the document as pdf in client-side

Use [pdf export component](#) in application level to export the document as pdf using [exportasimage](#) API. Here, all pages will be converted to image and inserted as pdf pages(works like print as PDF). There is one limitation we can't search the text because we are exporting the pdf as image.

Note: You can install the pdf export packages from this [link](#).

The following example code illustrates how to export the document as pdf in client-side.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
DocumentEditorContainerComponent, ImageFormat, Toolbar } from '@syncfusion/ej2-react-
documenteditor';
import {
PdfBitmap,
PdfDocument,

```

```
PdfPageOrientation,
PdfPageSettings,
PdfSection,
SizeF,
} from '@syncfusion/ej2-pdf-export';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
let container: DocumentEditorContainerComponent;
let contentChanged:boolean=false;
function onClick() {
let obj = container;
let pdfdocument: PdfDocument = new PdfDocument();
let count: number = obj.documentEditor.pageCount;
obj.documentEditor.documentEditorSettings.printDevicePixelRatio = 2;
let loadedPage = 0;
for (let i = 1; i <= count; i++) {
setTimeout(() => {
let format: ImageFormat = 'image/jpeg' as ImageFormat;
// Getting pages as image
let image = obj.documentEditor.exportAsImage(i, format);
image.onload = function () {
let imageHeight = parseInt(
image.style.height.toString().replace('px', ''))
);
let imageWidth = parseInt(
image.style.width.toString().replace('px', ''))
);
let section: PdfSection = pdfdocument.sections.add() as PdfSection;
let settings: PdfPageSettings = new PdfPageSettings(0);
if (imageWidth > imageHeight) {
settings.orientation = PdfPageOrientation.Landscape;
}
settings.size = new SizeF(imageWidth, imageHeight);
```

```

(section as PdfSection).setPageSettings(settings);
let page = section.pages.add();
let graphics = page.graphics;
let imageStr = image.src.replace('data:image/jpeg;base64,', '');
let pdfImage = new PdfBitmap(imageStr);
graphics.drawImage(pdfImage, 0, 0, imageWidth, imageHeight);
loadedPage++;
if (loadedPage == count) {
// Exporting the document as pdf
pdfdocument.save(
(obj.documentEditor.documentName === ''
? 'sample'
: obj.documentEditor.documentName) + '.pdf'
);
}
};
}, 500);
}
}
return (
<div>
<button id='export' onClick={onClick}>Export</button>
<DocumentEditorContainerComponent id="container" ref={{(scope) => {container = scope; }}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
/>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```


Export document as pdf in server-side using Syncfusion DocIO

With the help of [Syncfusion DocIO](#), you can export the document as Pdf in server-side. Here, you can search the text.

The following way illustrates how to convert the document as Pdf:

- Using [serialize](#) API, convert the document as Sfdt and send it to server-side.

The following example code illustrates how to convert the document to sfdt and pass it to server-side.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent, Toolbar
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let contentChanged: boolean = false;
  function onClick() {
    let obj = container;
    let http: XMLHttpRequest = new XMLHttpRequest();
    // Replace your running web service Url here
    http.open('POST', 'http://localhost:62869/api/documenteditor/ExportPdf');
    http.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
    http.responseType = 'json';
    //Serialize document content as SFDt.
    let sfdt: any = { content: obj.documentEditor.serialize() };
    //Send the sfdt content to server side.
    http.send(JSON.stringify(sfdt));
  }
  return (
    <div>
      <button id='export' onClick={onClick}>Export</button>
      <DocumentEditorContainerComponent id="container" ref={{scope} => { container = scope; }}
        height={'590px'}
    </div>
  );
}
```

```

serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
/>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

- Using Save API in server-side, you can convert the sfdt to stream.
- Finally, convert the stream to Pdf using [Syncfusion.DocIO.Renderer.Net.Core](#) library.

The following example code illustrates how to process the sfdt in server-side.

```

`c#
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("ExportPdf")]
public void ExportPdf([FromBody]SaveParameter data)
{
    // Converts the sfdt to stream
    Stream document = WordDocument.Save(data.content, FormatType.Docx);
    Syncfusion.DocIO.DLS.WordDocument doc = new Syncfusion.DocIO.DLS.WordDocument(document,
    Syncfusion.DocIO.FormatType.Docx);
    //Instantiation of DocIO.Renderer for Word to PDF conversion
    DocIO.Renderer render = new DocIO.Renderer();
    //Converts Word document into PDF document
    PdfDocument pdfDocument = render.ConvertToPDF(doc);
    // Saves the document to server machine file system, you can customize here to save into databases or
    file servers based on requirement.
    FileStream fileStream = new FileStream("sample.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
    //Saves the PDF file
    pdfDocument.Save(fileStream);
    pdfDocument.Close();
}
`

```

```
fileStream.Close();
document.Close();
}
`
```

Get the complete working sample in this [link](#).

Customize font family drop down in React Document editor component

Document editor provides an options to customize the font family drop down list values using [fontfamilies](#) in Document editor settings. Fonts which are added in fontFamilies of [documentEditorSettings](#) will be displayed on font drop down list of text properties pane and font dialog.

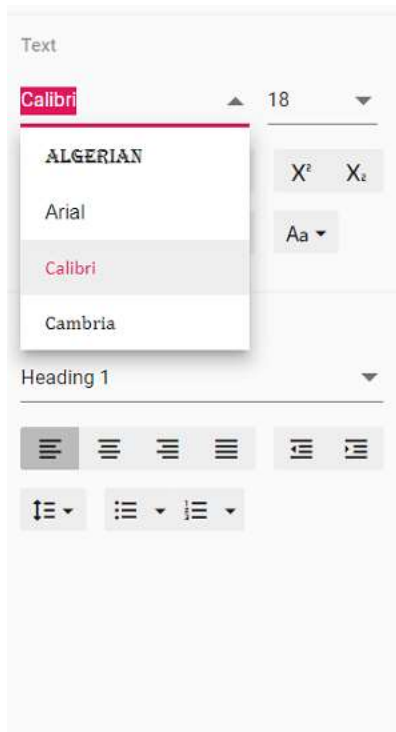
Similarly, you can use [documentEditorSettings](#) property for DocumentEditor also.

The following example code illustrates how to change the font families in Document editor container.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  // Add required font families to list it in font drop down
  let fontFamilies = {
    fontFamilies: ['Algerian', 'Arial', 'Calibri', 'Cambria'],
  };
  return (
    <DocumentEditorContainerComponent
      id="container"
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      documentEditorSettings={fontFamilies}
    />
  );
}
```

```
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
```

Output will be like below:



[Auto save document in document editor in React Document editor component](#)

In this article, we are going to see how to autosave the document in AWS S3. You can automatically save the edited content in regular intervals of time. It helps reduce the risk of data loss by saving an open document automatically at customized intervals.

The following example illustrates how to auto save the document in AWS S3.

- In the client-side, using content change event, we can automatically save the edited content in regular intervals of time. Based on `contentChanged` boolean, the document send as Docx format to server-side using [saveAsBlob](#) method.

```
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
```

```
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  let contentChanged: boolean = false;
  React.useEffect(() => {
    onCreate()
    onContentChange()
  }, []);
  function onCreate() {
    setInterval(() => {
      if (contentChanged) {
        //You can save the document as below
        container.documentEditor.saveAsBlob('Docx').then((blob: Blob) => {
          console.log('Saved sucessfully');
          let exportedDocument: Blob = blob;
          //Now, save the document where ever you want.
          let formData: FormData = new FormData();
          formData.append('fileName', 'sample.docx');
          formData.append('data', exportedDocument);
          / tslint:disable /
          var req = new XMLHttpRequest();
          // Replace your running Url here
          req.open(
            'POST',
            'http://localhost:62869/api/documenteditor/SaveToS3',
            true
          );
          req.onreadystatechange = () => {
            if (req.readyState === 4) {
              if (req.status === 200 || req.status === 304) {
                console.log('Saved sucessfully');
              }
            }
          }
        });
      }
    });
  }
}
```

```

};
req.send(formData);
});
contentChanged = false;
}
}, 1000);
}
function onContentChange() {
contentChanged = true;
}
return (
<DocumentEditorContainerComponent
id="container"
ref={(scope) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true} created={onCreate} contentChange={onContentChange}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

- In server-side, configure the access key and secret key in `web.config` file and register profile in `startup.cs`.

In `web.config`, add key like below format:

```

`c#
<appSettings>
<add key="AWSProfileName" value="sync_development" />
<add key="AWSAccessKey" value="" />

```

```
<add key="AWSSecretKey" value="" />
</appSettings>
```

In `startup.cs`, register profile in below format:

```
`c#
Amazon.Util.ProfileManager.RegisterProfile("sync_development", "", "");
```

- In server-side, Receives the stream content from client-side and process it to save the document in aws s3. Add Web API in controller file like below to save the document in aws s3.

```
[AcceptVerbs("Post")]
[HttpPost]
[EnableCors("AllowAllOrigins")]
[Route("SaveToS3")]
public string SaveToS3()
{
    IFormFile file = HttpContext.Request.Form.Files[0];
    Stream stream = new MemoryStream();
    file.CopyTo(stream);
    UploadFileStreamToS3(stream, "documenteditor", "", "GettingStarted.docx");
    stream.Close();
    return "Sucess";
}

public bool UploadFileStreamToS3(System.IO.Stream localFilePath, string bucketName, string
subDirectoryInBucket, string fileNameInS3)
{
    AWSCredentials credentials = new StoredProfileAWSCredentials("sync_development");
    IAmazonS3 client = new AmazonS3Client(credentials, Amazon.RegionEndpoint.USEast1);
    TransferUtility utility = new TransferUtility(client);
    TransferUtilityUploadRequest request = new TransferUtilityUploadRequest();
    if (subDirectoryInBucket == "" || subDirectoryInBucket == null)
    {
        request.BucketName = bucketName; //no subdirectory just bucket name
```

```

}
else
{ // subdirectory and bucket name
request.BucketName = bucketName + @"/" + subDirectoryInBucket;
}
request.Key = fileNameInS3; //file name up in S3
request.InputStream = localFilePath;
utility.Upload(request); //commencing the transfer
return true; //indicate that the file was sent
}
,

```

Get the complete working sample in this [link](#).

Retrieve the bookmark content as text in React Document editor component

You can get the bookmark or whole document content from the React Document Editor component as plain text and SFDT (rich text).

Get the bookmark content as plain text

You can [selectBookmark](#) API to navigate to the bookmark and use [text](#) API to get the bookmark content as plain text from React Document Editor component.

The following example code illustrates how to get the bookmark content as plain text.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
let container: DocumentEditorContainerComponent;
function App() {
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select all the content in document
    container.documentEditor.selection.selectAll();
  }
}

```



```
// Insert bookmark to selected content
container.documentEditor.editor.insertBookmark('Bookmark1');
// Provide your bookmark name to navigate to specific bookmark
container.documentEditor.selection.selectBookmark('Bookmark1');
// To get the selected content as text
let selectedContent = container.documentEditor.selection.text;
}
return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
  container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreated}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

To get the bookmark content as SFDT (rich text), please check this [link](#)

[Get the whole document content as text](#)

You can use [text](#) API to get the whole document content as plain text from React Document Editor component.

The following example code illustrates how to get the whole document content as plain text.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
```

```

} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select all the content in document
    container.documentEditor.selection.selectAll();
    // To get the content as text
    let selectedContent: string = container.documentEditor.selection.text;
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={{scope} => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreated}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Get the whole document content as SFDT(rich text)

You can use [serialize](#) API to get the whole document content as SFDT string from React Document Editor component.

The following example code illustrates how to get the whole document content as SFDT.

```

`ts
import * as ReactDOM from 'react-dom';

```

```

import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To get the content as SFDT
    let selectedContent: string = container.documentEditor.serialize();
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreated}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Get the header content as text

You can use [goToHeader](#) API to navigate the selection to the header and then use [text](#) API to get the content as plain text.

The following example code illustrates how to get the header content as plain text.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To navigate the selection to header
    container.documentEditor.selection.goToHeader();
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select all the content in document
    container.documentEditor.selection.selectAll();
    // To get the selected content as text
    let selectedContent: string = container.documentEditor.selection.text;
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreated}
    />
  );
}
```

```
export default App()
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Similarly, you can use [goToFooter](#) API to navigate the selection to the footer and then use [text](#) API to get the content as plain text.

Get current word in React Document editor component

You can get the current word or paragraph content from the React Document Editor component as plain text and SFDT (rich text).

Select and get the word in current cursor position

You can use [selectCurrentWord](#) API in selection module to select the current word at cursor position and use [text](#) API to get the selected content as plain text from React Document Editor component.

The following example code illustrates how to select and get the current word as plain text.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // Move selection to previous character
    container.documentEditor.selection.moveToPreviousCharacter();
    // To select the current word in document
    container.documentEditor.selection.selectCurrentWord();
    // To get the selected content as text
    let selectedContent: string = container.documentEditor.selection.text;
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
```

```

ref={scope} => {
  container = scope;
}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreated}
/>
);
}

export default App
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

To get the bookmark content as SFDT (rich text), please check this [link](#)

Select and get the paragraph in current cursor position

You can use [selectParagraph](#) API in selection module to select the current paragraph at cursor position and use [text](#) API or [sfdt](#) API to get the selected content as plain text or SFDT from React Document Editor component.

The following example code illustrates how to select and get the current paragraph as SFDT.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To select the current paragraph in document
    container.documentEditor.selection.selectParagraph();
  }
}

```

```
// To get the selected content as SFDT
let selectedContent: string = container.documentEditor.selection.sfdt;
}
return (
<DocumentEditorContainerComponent
id="container"
ref={{scope}}=> {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreated}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
,
```

Insert page number and navigate to page in React Document editor component

You can insert page number and navigate to specific page in React Document Editor component by following ways.

Insert page number

You can use [insertPageNumber](#) API in editor module to insert the page number in current cursor position. By default, Page number will insert in Arabic number style. You can change it, by providing the number style in parameter.

Note: Currently, Documenteditor have options to insert page number at current cursor position.

The following example code illustrates how to insert page number in header.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
DocumentEditorContainerComponent,
Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
```

```

DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreate() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To move the selection to header
    container.documentEditor.selection.goToHeader();
    // Insert page number in the current cursor position
    container.documentEditor.editor.insertPageNumber();
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreate}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Also, you use [insertField](#) API in Editor module to insert the Page number in current position

```

`ts
//Current page number
this.container.documentEditor.editor.insertField('PAGE * MERGEFORMAT', '1');

```


Get page count

You can use [pageCount](#) API to get the total number of pages in Document.

The following example code illustrates how to get the number of pages in Document.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    // To insert text in cursor position
    container.documentEditor.editor.insertText('Document editor');
    // To get the total number of pages
    let pageCount = container.documentEditor.pageCount;
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      created={onCreated}
    />
  );
}
export default App;
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

,

Navigate to specific page

You can use [goToPage](#) API in Selection module to move selection to the start of the specified page number.

The following example code illustrates how to move selection to specific page.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import {
```

```
  DocumentEditorContainerComponent,
```

```
  Toolbar,
```

```
} from '@syncfusion/ej2-react-documenteditor';
```

```
DocumentEditorContainerComponent.Inject(Toolbar);
```

```
function App() {
```

```
  let container: DocumentEditorContainerComponent;
```

```
  function onCreated() {
```

```
    // To move selection to page number 2
```

```
    container.documentEditor.selection.goToPage(2);
```

```
  }
```

```
  return (
```

```
    <DocumentEditorContainerComponent
```

```
      id="container"
```

```
      ref={(scope) => {
```

```
        container = scope;
```

```
      }}
```

```
      height={'590px'}
```

```
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
```

```
      enableToolbar={true}
```

```
      created={onCreated}
```

```
    />
```

```
  );
```

```
}
```

```
export default App;
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Move selection to specific position in React Document editor component

Using [select](#) API in selection module, You can set cursor position to anywhere in the document.

Selects content based on start and end hierarchical index

Hierarchical index will be in below format.

```
sectionIndex;blockIndex;offset
```

The following code snippet illustrate how to select using hierarchical index.

```
`ts
// Selection will occur between provided start and end offset
this.documentEdContainerIns.documentEditor.editor.insertText("Welcome");
// The below code will select the letters "We" from inserted text "Welcome"
this.documentEdContainerIns.documentEditor.selection.select("0;0;0", "0;0;2");
```

The following table illustrates about Hierarchical index in detail.

Element	Hierarchical Format	Explanation
Move selection to Paragraph	sectionIndex;blockIndex;offset	 Ex: 0;0;0 It moves the cursor to the start of paragraph.
Move selection to Table	sectionIndex;tableIndex;rowIndex;cellIndex;blockIndex;offset	 Ex: 0;0;0;0;1;0 It moves the cursor to the second paragraph which is inside first row and cell of table.
Move selection to header	pageIndex;H;sectionIndex;blockIndex;offset	 Ex: 1;H;0;0;0 It moves cursor to the header in second page.
Move selection to Footer	pageIndex;F;sectionIndex;blockIndex;offset	 Ex: 1;F;0;0;0 It moves cursor to the footer in second page.

Get the selection start and end hierarchical index

Using [startOffset](#), you can get start hierarchical index which denotes the start index of current selection.

Similarly, using [endOffset](#), you can get end hierarchical index which denotes the end index of current selection.

The following code snippet illustrate how to get the selection start and end offset on selection changes in document.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
```

```

Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function selectionChanges() {
    //Get the start index of current selection
    let startOffset: string =
      container.documentEditor.selection.startOffset;
    //Get the end index of current selection
    let endOffset: string = container.documentEditor.selection.endOffset;
  }
  return (
    <DocumentEditorContainerComponent
      id="container"
      ref={(scope) => {
        container = scope;
      }}
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={true}
      selectionChange={selectionChanges}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Document editor have [selectionChange](#) event which is triggered whenever the selection changes in Document.

Selects the content based on left and top position

Here, you can specify the [selection settings](#) to select the content based on left and top position.

x denotes the left position and y denotes the top position and extend denotes whether to extend or update selection.

Please check below code sample for reference.

```
`ts
```

```
this.container.documentEditor.selection.select({ x: 188.4814208984375 , y: 662.00005, extend: true });
```

```
,
```

Disable header and footer edit in document editor in React Document editor component

Disable header and footer edit in DocumentEditorContainer instance

You can use [restrictEditing](#) property to disable header and footer editing based on selection context type.

RestrictEditing allows you to restrict the document modification and makes the Document read only mode. So, by using this property, and if selection inside header or footer, you can set this property as true.

The following example code illustrates how to header and footer edit in `DocumentEditorContainer` instance.

```
`ts
```

```
import * as ReactDOM from 'react-dom';
```

```
import * as React from 'react';
```

```
import {
```

```
DocumentEditorContainerComponent,
```

```
Toolbar,
```

```
} from '@syncfusion/ej2-react-documenteditor';
```

```
DocumentEditorContainerComponent.Inject(Toolbar);
```

```
function App() {
```

```
let container: DocumentEditorContainerComponent;
```

```
function selectionChanges() {
```

```
// Check whether selection is in header
```

```
if (container.documentEditor.selection.contextType.indexOf('Header') > -1 ||
```

```
// Check whether selection is in Footer
```

```
container.documentEditor.selection.contextType.indexOf('Footer') > -1) {
```

```
// Change the document to read only mode
```

```
container.restrictEditing = true;
```

```
} else {
```

```
// Change the document to editable mode
```

```
container.restrictEditing = false;
```

```
}
```

```
};
```

```

return (
  <DocumentEditorContainerComponent
    id="container" ref={(scope) => {
      container = scope;
    }}
    height={'590px'}
    serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
    enableToolbar={true}
    selectionChange={selectionChanges}
  />
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Otherwise, you can disable clicking inside Header or Footer by using [closeHeaderFooter](#) API in selection module.

The following example code illustrates how to close header and footer when selection is inside header or footer in `DocumentEditorContainer` instance.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function selectionChanges() {
    // Check whether selection is in header
    if (container.documentEditor.selection.contextType.indexOf('Header') > -1 ||
    // Check whether selection is in Footer
    container.documentEditor.selection.contextType.indexOf('Footer') > -1) {

```

```
// Close header Footer
container.documentEditor.selection.closeHeaderFooter();
}
};
return (
<DocumentEditorContainerComponent
id="container" ref={{(scope) => {
container = scope;
}}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
selectionChange={selectionChanges}
/>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Disable header and footer edit in DocumentEditor instance

Like restrictEditing, you can use [isReadOnly](#) property in Document editor to disable header and footer edit.

The following example code illustrates how to header and footer edit in **DocumentEditor** instance.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
DocumentEditorComponent,
Print,
SfdtExport,
WordExport,
TextExport,
Selection,
Search,
```

```
Editor,  
ImageResizer,  
EditorHistory,  
ContextMenu,  
OptionsPane,  
HyperlinkDialog,  
TableDialog,  
BookmarkDialog,  
TableOfContentsDialog,  
PageSetupDialog,  
StyleDialog,  
ListDialog,  
ParagraphDialog,  
BulletsAndNumberingDialog,  
FontDialog,  
TablePropertiesDialog,  
BordersAndShadingDialog,  
TableOptionsDialog,  
CellOptionsDialog,  
StylesDialog,  
} from '@syncfusion/ej2-react-documenteditor';  
DocumentEditorComponent.Inject(  
Print,  
SfdtExport,  
WordExport,  
TextExport,  
Selection,  
Search,  
Editor,  
ImageResizer,  
EditorHistory,  
ContextMenu,  
OptionsPane,
```



```
HyperlinkDialog,  
TableDialog,  
BookmarkDialog,  
TableOfContentsDialog,  
PageSetupDialog,  
StyleDialog,  
ListDialog,  
ParagraphDialog,  
BulletsAndNumberingDialog,  
FontDialog,  
TablePropertiesDialog,  
BordersAndShadingDialog,  
TableOptionsDialog,  
CellOptionsDialog,  
StylesDialog  
);  
function App(){  
  let documentEditor: DocumentEditorComponent= new DocumentEditorComponent(undefined);  
  React.useEffect(() => {  
    selectionChanges();  
  }, []);  
  return (  
    <DocumentEditorComponent  
      id="documentEditor"  
      ref={(scope) => {  
        documentEditor = scope;  
      }}  
      height={'330px'}  
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"  
      isReadOnly={false}  
      enablePrint={true}  
      enableSelection={true}  
      enableEditor={true}
```

```
enableEditorHistory={true}
enableContextMenu={true}
enableSearch={true}
enableOptionsPane={true}
enableBookmarkDialog={true}
enableBordersAndShadingDialog={true}
enableFontDialog={true}
enableTableDialog={true}
enableParagraphDialog={true}
enableHyperlinkDialog={true}
enableImageResizer={true}
enableListDialog={true}
enablePageSetupDialog={true}
enableSfdtExport={true}
enableStyleDialog={true}
enableTableOfContentsDialog={true}
enableTableOptionsDialog={true}
enableTablePropertiesDialog={true}
enableTextExport={true}
enableWordExport={true}
selectionChange={selectionChanges}
/>
);
function selectionChanges() {
// Check whether selection is in header
if (documentEditor.selection.contextType.indexOf('Header') > -1 ||
// Check whether selection is in Footer
documentEditor.selection.contextType.indexOf('Footer') > -1
) {
// Change the document to read only mode
documentEditor.isReadOnly = true;
} else {
// Change the document to editable mode
```

```

documentEditor.isReadOnly = false;
}
}
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Insert text in current position in React Document editor component

You can insert the text, paragraph and rich-text content in React Document Editor component.

Insert text in current cursor position

You can use [insertText](#) API in editor module to insert the text in current cursor position.

The following example code illustrates how to add the text in current selection.

```

`ts

// It will insert the provided text in current selection
this.container.documentEditor.editor.insertText('Syncfusion');

import * as ReactDOM from 'react-dom';
import * as React from 'react';

import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);

export class Default extends React.Component {
  insertText(){
    // It will insert the provided text in current selection
    this.container.documentEditor.editor.insertText('Syncfusion');
  };

  render() {
    return (
      <button id='insert' onClick={this.insertText.bind(this)}>Insert Text</button>
      <DocumentEditorContainerComponent
        id="container"
        ref={(scope) => {
          this.container = scope;

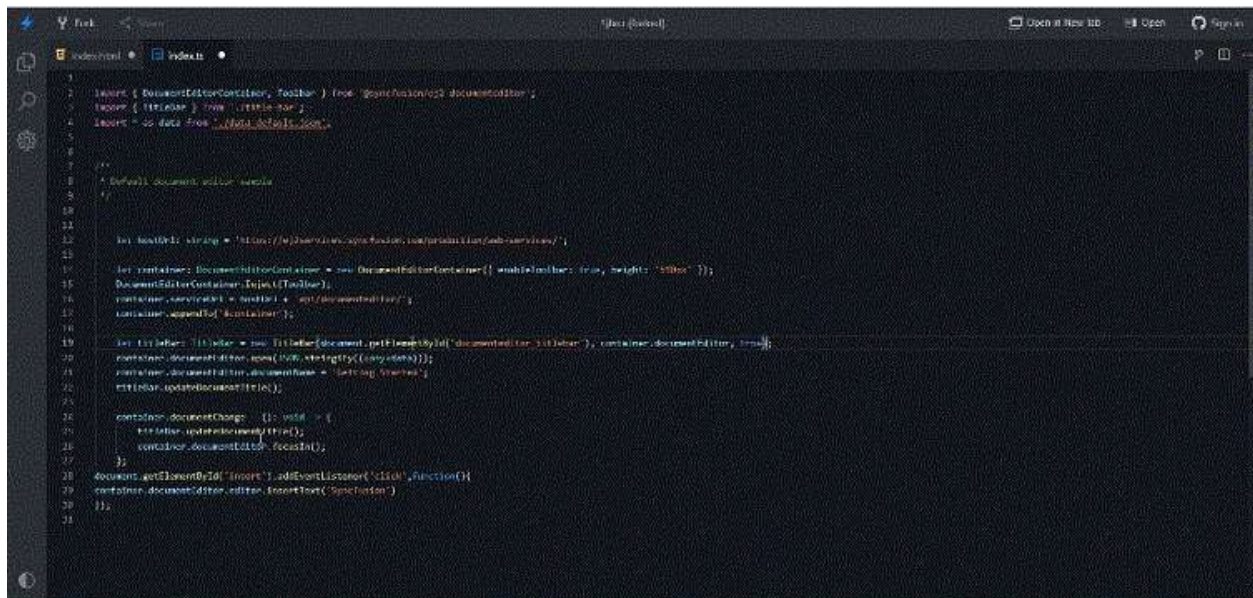
```

```

}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
/>
);
}
}
ReactDOM.render(<Default />, document.getElementById('sample'));

```

Please check below gif which illustrates how to insert text in current cursor position on button click:



Insert paragraph in current cursor position

To insert new paragraph at current selection, you can use [insertText](#) API with parameter as `\r\n` or `\n`.

The following example code illustrates how to add the new paragraph in current selection.

```

`ts
// It will add the new paragraph in current selection
this.container.documentEditor.editor.insertText("\n");

```

Insert the rich-text content

To insert the HTML content, you have to convert the HTML content to SFDT Format using [web service](#). Then use [paste](#) API to insert the sfdt at current cursor position.

Note: Html string should be welformatted html. [DocIO](#) support only welformatted XHTML.

The following example illustrates how to insert the HTML content at current cursor position.

- Send the HTML content to server side for SFDT conversion. Refer to the following example to send the HTML content to server side and inserting it in current cursor position.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let container: DocumentEditorContainerComponent;
  function onCreated() {
    let htmltags: string =
      "<?xml version='1.0' encoding='utf - 8'?><!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Strict//EN'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd><html xmlns='http://www.w3.org/1999/xhtml' xml:lang='en' lang='en'><body><h1>The img element</h1><img src='https://www.w3schools.com/images/lamp.jpg' alt='Lamp Image' width='500' height='600'/></body></html>";
    document.getElementById('export').addEventListener('click', () => {
      let http: XMLHttpRequest = new XMLHttpRequest();
      http.open('POST', 'http://localhost:5000/api/documenteditor/LoadString');
      http.setRequestHeader('Content-Type', 'application/json;charset=UTF-8');
      http.responseType = 'json';
      http.onreadystatechange = function () {
        if (http.readyState === 4) {
          if (http.status === 200 || http.status === 304) {
            // Insert the sfdt content in cursor position using paste API
            container.documentEditor.editor.paste(http.response);
          }
        }
      };
    });
  }
}
```

```

    } else {
    alert('failed;');
    }
    }
};

let htmlContent: any = { content: htmltags };
http.send(JSON.stringify(htmlContent));
});
}

return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreated}
/>
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`

```

- Please refer the following code example for server-side web implementation for HTML conversion using DocumentEditor.

```

`c#
//API controller for the conversion.
[HttpPost]
public string LoadString([FromBody]InputParameter data)
{
// You can also load HTML file/string from server side.

```

```
Syncfusion.EJ2.DocumentEditor.WordDocument document =
Syncfusion.EJ2.DocumentEditor.WordDocument.LoadString(data.content, FormatType.Html); // Convert
the HTML to SFDT format.
```

```
string json = Newtonsoft.Json.JsonConvert.SerializeObject(document);
```

```
document.Dispose();
```

```
return json;
```

```
}
```

```
public class InputParameter
```

```
{
```

```
public string content {get; set; }
```

```
}
```

```
,
```

Note: The above example illustrates inserting HTML content. Similarly, you can insert any rich-text content by converting any of the supported file formats (DOCX, DOC, WordML, HTML, RTF) to SFDT.

[Change the cursor color in document editor in React Document editor component](#)

Document Editor default cursor color is black. The user can change the color by overriding the css property using class name. The Document editor cursor css have a class named `e-de-blink-cursor`.

Please refer the below code snippet to change the cursor color to red.

```
`css
```

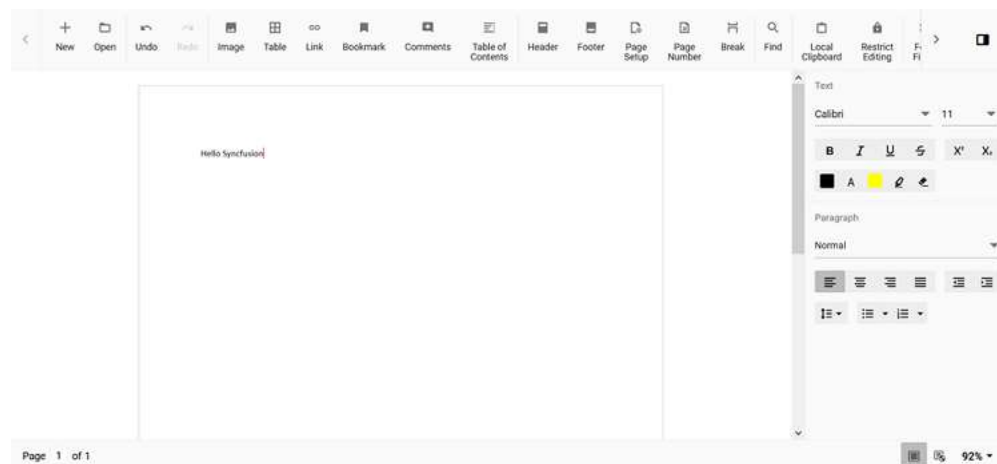
```
.e-de-blink-cursor {
```

```
border-left: 1px solid red!important;
```

```
}
```

```
,
```

Output will be like below:



Hide tool bar and properties pane in React Document editor component

Document editor container provides the main document view area along with the built-in toolbar and properties pane.

Document editor provides just the main document view area. Here, the user can compose, view, and edit the Word documents. You may prefer to use this component when you want to design your own UI options for your application.

Hide the properties pane

By default, Document editor container has built-in properties pane which contains options for formatting text, table, image and header and footer. You can use [showPropertiesPane](#) API in [DocumentEditorContainer](#) to hide the properties pane.

The following example code illustrates how to hide the properties pane.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  return (
    <DocumentEditorContainerComponent
      id="container"
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      showPropertiesPane={false}
    />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Note: Positioning and customizing the properties pane in Document editor container is not possible. Instead, you can hide the existing properties pane and create your own pane using public API's.

Hide the toolbar

You can use [enableToolbar](#) API in [DocumentEditorContainer](#) to hide the existing toolbar.

The following example code illustrates how to hide the existing toolbar.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent
} from '@syncfusion/ej2-react-documenteditor';
function App() {
  return (
    <DocumentEditorContainerComponent
      id="container"
      height={'590px'}
      serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
      enableToolbar={false}
    />
  );
}
export default App
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [How to customize the toolbar](#)

Insert text or image in table programmatically in React Document editor component

Using Document editor API's, you can insert [text](#) or [image](#) in [table](#) programmatically based on your requirement.

Use [selection](#) API's to navigate between rows and cells.

The following example illustrates how to create 2*2 table and then add text and image programmatically.

```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
```

```

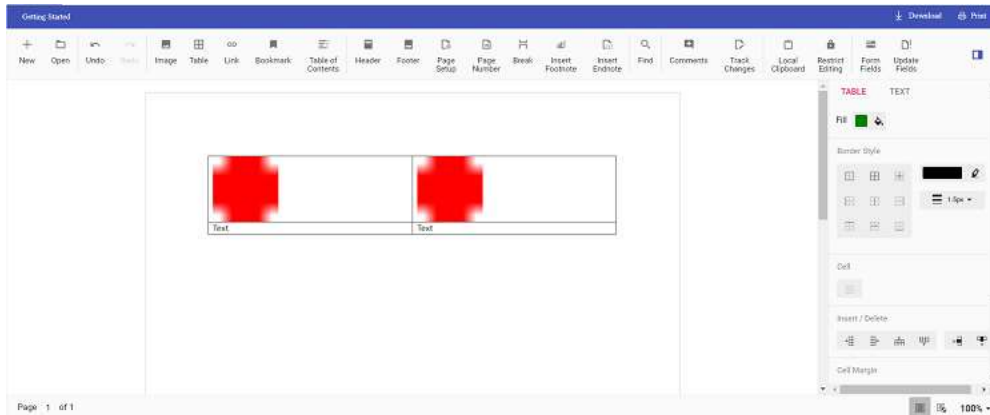
Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
let container: DocumentEditorContainerComponent;
function onCreated() {
// To insert the table in cursor position
container.documentEditor.editor.insertTable(2, 2);
// To insert the image at table first cell
container.documentEditor.editor.insertImage(
'data:image/png;base64,iVBORw0KGgoAAAANSUheUgAAAAUAAAAFCAYAAACNbybIAAAAHIEQVQI12P4
//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg=='
);
// To move the cursor to next cell
moveCursorToNextCell();
// To insert the image at table second cell
container.documentEditor.editor.insertImage(
'data:image/png;base64,iVBORw0KGgoAAAANSUheUgAAAAUAAAAFCAYAAACNbybIAAAAHIEQVQI12P4
//8/w38GIAXDIBKE0DHxgljNBAAO9TXL0Y4OHwAAAABJRU5ErkJggg=='
);
// To move the cursor to next row
moveCursorToNextRow();
// To insert text in cursor position
container.documentEditor.editor.insertText('Text');
// To move the cursor to next cell
moveCursorToNextCell();
// To insert text in cursor position
container.documentEditor.editor.insertText('Text');
}
function moveCursorToNextCell() {
// To get current selection start offset
var startOffset = container.documentEditor.selection.startOffset;
// Increasing cell index to consider next cell
var startOffsetArray = startOffset.split(';');

```

```
startOffsetArray[3] = parseInt(startOffsetArray[3]) + 1;
// Changing start offset
startOffset = startOffsetArray.join(';');
// Navigating selection using select method
container.documentEditor.selection.select(startOffset, startOffset);
}
function moveCursorToNextRow() {
// To get current selection start offset
var startOffset = container.documentEditor.selection.startOffset;
// Increasing row index to consider next row
var startOffsetArray = startOffset.split(';');
startOffsetArray[2] = parseInt(startOffsetArray[2]) + 1;
// Going back to first cell
startOffsetArray[3] = 0;
// Changing start offset
startOffset = startOffsetArray.join(';');
// Navigating selection using select method
container.documentEditor.selection.select(startOffset, startOffset);
}
return (
<DocumentEditorContainerComponent
id="container"
ref={({scope}) => {
container = scope;
}}
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
created={onCreated}
/>
);
}
export default App;
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

The output will be like below.



Change the default search highlight color in React Document editor component. Document editor provides an options to change the default search highlight color using [searchHighlightColor](#) in Document editor settings. The highlight color which is given in [documentEditorSettings](#) will be highlighted on the searched text. By default, search highlight color is yellow.

Similarly, you can use [documentEditorSettings](#) property for DocumentEditor also.

The following example code illustrates how to change the default search highlight color.

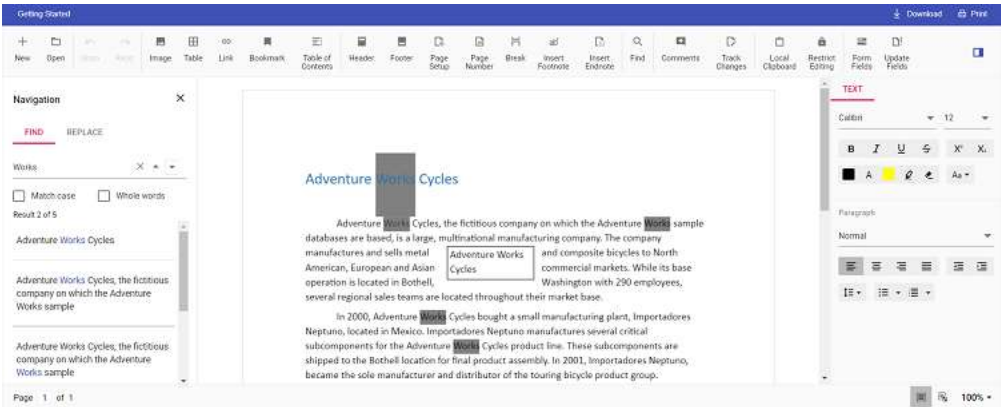
```
`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  // Add required color to change the default search highlight color
  let searchHighlightColor = {
    searchHighlightColor: 'Grey'
  };
  return (
    <DocumentEditorContainerComponent
      id="container"
```

```
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
documentEditorSettings={searchHighlightColor}
/>
);
}

export default App;

ReactDOM.render(<App />, document.getElementById('root'));
```

Output will be like below:



How to optimize the SFDT file

Starting from version v21.1.x, the SFDT file generated in Word Processor component is optimized by default to reduce the file size. All static keys are minified, and the final JSON string is compressed. This helps reduce the SFDT file size relative to a DOCX file and provides the following benefits,

- File transfer between client and server through the internet gets faster.
- The new optimized SFDT files require less storage space than the old SFDT files.

Hence, the optimized SFDT file can't be directly manipulated as JSON string.

This feature comes with a public API to switch between the old and new optimized SFDT format, allowing backward compatibility.

As a backward compatibility to create older format SFDT files, refer the following code changes,

Client/Server	Old Code	New Code from v21.1.x
Client-side		function App() { // Set optimize sfdt as false let settings = { optimizeSfdt: false }; return ();}

Server-side C#	WordDocument sfdtDocument = WordDocument.Load(stream, formatType);string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);	WordDocument sfdtDocument = WordDocument.Load(stream, formatType);sfdtDocument.OptimizeSfdt = false;string sfdt = Newtonsoft.Json.JsonConvert.SerializeObject(sfdtDocument);
Server-side Java	String sfdtDocument = WordProcessorHelper.load(stream, formatType);	String sfdtDocument = WordProcessorHelper.load(stream, formatType, false);

To convert from older format SFDT from a new optimized SFDT file, refer the following code example,

Client/Server	Code example
Client-side	function App() { // Set optimize sfdt as false let settings = { optimizeSfdt: false }; return ();}
Server-side C#	using(Syncfusion.DocIO.DLS.WordDocument docIODocument = WordDocument.Save(optimizedSfdt)) { sfdtDocument = WordDocument.Load(docIODocument); sfdtDocument.OptimizeSfdt = false; string oldSfdt = JsonSerializer.Serialize(sfdtDocument);}
Server-side Java	WordDocument docIODocument = WordProcessorHelper.save(optimizedSfdt);String oldSfdt = WordProcessorHelper.load(docIODocument, false);

How to disable auto focus in Syncfusion React Document Editor component

Document Editor gets focused automatically when the page loads. If you want the Document editor not to be focused automatically it can be customized.

The following example illustrates to disable the auto focus in DocumentEditorContainer.

```
`typescript
<DocumentEditorContainerComponent id="container" height={'590px'} enableAutoFocus={false} />
`
```

Note: Default value of [enableAutoFocus](#) property is `true`.

The following example illustrates to disable the auto focus in DocumentEditor.

```
`typescript
<DocumentEditorComponent id="container" height={'590px'} enableAutoFocus={false}/>
`
```

Note: Default value of [enableAutoFocus](#) property is `true`.

How to disable drag and drop in document editor in React Document editor component

Document Editor provides support to drag and drop contents within the component and it can be customized(enable and disable) using [allowDragAndDrop](#) property in Document editor settings.

The following example illustrates to customize the drag and drop option.

```
`typescript
var settings = { allowDragAndDrop: false };
var hostUrl = 'https://services.syncfusion.com/react/production/api/documenteditor/';
<DocumentEditorContainerComponent id="container" height={'590px'} serviceUrl={hostUrl}
documentEditorSettings={settings}/>
`
```

Note: Default value of [allowDragAndDrop](#) property is `true`.

The following example illustrates to disable the drag and drop option in DocumentEditor.

```
`typescript
var settings = { allowDragAndDrop: false };
var hostUrl = 'https://services.syncfusion.com/react/production/api/documenteditor/';
<DocumentEditorComponent id="container" height={'590px'} documentEditorSettings={settings}/>
`
```

Note: Default value of [allowDragAndDrop](#) property is `true`.

Enable ruler

How to enable ruler in Document Editor component

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor.

The following example illustrates how to enable ruler in Document Editor

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
  let container;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function created() {
    container.documentEditorSettings.showRuler = true;
    container.enableAllModules();
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }
  function onClick() {
```

```

        container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
    }
    return (<div>
        <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
        <DocumentEditorComponent id="container" height={'590px'}
isReadOnly={false} ref={ (scope) => {
            container = scope;
            created();
        }}
        serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/"
        />
    </div>

    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorComponent, Editor } from '@syncfusion/ej2-react-
documenteditor';
DocumentEditorComponent.Inject(Editor);
function App() {
    let container: DocumentEditorComponent;
    React.useEffect(() => {
        componentDidMount()
    }, []);
    function created() {
        container.documentEditorSettings.showRuler = true;
        container.enableAllModules();
    }
    function componentDidMount() {
        setTimeout(() => {
            created();
        });
    }
    function onClick() {
        container.documentEditorSettings.showRuler =
!container.documentEditorSettings.showRuler;
    }
    return ( <div>
        <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
        <DocumentEditorComponent id="container" height={'590px'}
isReadOnly={false} ref={ (scope) => {
            container = scope;
            created();
        }}
    </div>

```



```

        serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/"
    />
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'));
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Syncfusion React Button</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Essential JS 2 for React Components" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
    <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdn.jsdelivr.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
    <script src="systemjs.config.js"></script>
</head>
<body>
    <div id='root'>
        <div id='loader'>Loading....</div>
    </div>
</body>
</html>

```

How to enable ruler in Document Editor Container component

Using ruler we can refer to setting specific margins, tab stops, or indentations within a document to ensure consistent formatting in Document Editor Container.

The following example illustrates how to enable ruler in Document Editor Container.

INDEX.JSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let documenteditorcontainer;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onClick() {
    documenteditorcontainer.documentEditorSettings.showRuler =
!documenteditorcontainer.documentEditorSettings.showRuler;
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }
  function created() {
    documenteditorcontainer.documentEditorSettings.showRuler = true;
  }
  return (
    <div>
      <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
      <DocumentEditorContainerComponent height={'590px'} id="container"
ref={ (scope) => {
        documenteditorcontainer = scope;
        created();
      }}
      serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true} />
    </div>

  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'))
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import { DocumentEditorContainerComponent, Toolbar } from '@syncfusion/ej2-
react-documenteditor';
```

```

DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  let documenteditorcontainer: DocumentEditorContainerComponent;
  React.useEffect(() => {
    componentDidMount()
  }, []);
  function onClick() {
    documenteditorcontainer.documentEditorSettings.showRuler =
!documenteditorcontainer.documentEditorSettings.showRuler;
  }
  function componentDidMount() {
    setTimeout(() => {
      created();
    });
  }
  function created() {
    documenteditorcontainer.documentEditorSettings.showRuler = true;
  }
  return (
    <div>
      <button id='container_ruler_button' onClick={onClick}>Show/Hide
Ruler</button>
      <DocumentEditorContainerComponent height={'590px'} id="container"
ref={(scope) => {
        documenteditorcontainer = scope;
        created();
      }}
      serviceUrl="https://ej2services.syncfusion.com/production/web-
services/api/documenteditor/" enableToolbar={true} />
    </div>
  )
}
export default App;
ReactDOM.render(<App />, document.getElementById('root'))
{% endraw %}

```

INDEX.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <title>Syncfusion React Button</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta name="description" content="Essential JS 2 for React Components" />
  <meta name="author" content="Syncfusion" />
  <link href="index.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-react-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
documenteditor/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
buttons/styles/fabric.css" rel="stylesheet" />
  <link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
base/styles/fabric.css" rel="stylesheet" />

```

```

<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
dropdowns/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
inputs/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
splitbuttons/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
lists/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
navigations/styles/fabric.css" rel="stylesheet" />
<link href="https://cdn.syncfusion.com/ej2/20.3.56/ej2-
popups/styles/fabric.css" rel="stylesheet" />
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scr
ipt>
<script src="systemjs.config.js"></script>
</head>
<body>
  <div id='root'>
    <div id='loader'>Loading....</div>
  </div>
</body>
</html>

```

Customize color picker in React Document editor control

Document editor provides an options to customize the color picker using [colorPickerSettings](#) in Document editor settings. Color picker offers customization options for default appearance, by allowing selection between Picker or Palette mode, for font and border colors."

Similarly, you can use [documentEditorSettings](#) property for DocumentEditor also.

The following example code illustrates how to Customize the color picker in Document editor container.

```

`ts
import * as ReactDOM from 'react-dom';
import * as React from 'react';
import {
  DocumentEditorContainerComponent,
  Toolbar,
} from '@syncfusion/ej2-react-documenteditor';
DocumentEditorContainerComponent.Inject(Toolbar);
function App() {
  // Add required color picker settings to the document editor settings
  let settings = {colorPickerSettings: { mode: 'Palette', modeSwitcher: true, showButtons: true }};
  return (

```

```

<DocumentEditorContainerComponent
id="container"
height={'590px'}
serviceUrl="https://ej2services.syncfusion.com/production/web-services/api/documenteditor/"
enableToolbar={true}
documentEditorSettings={settings}
/>
);
}

export default App;
ReactDOM.render(<App />, document.getElementById('root'));
`

```

The following table illustrates all the possible properties for the color picker.

Property	Behaviour
columns	It is used to render the ColorPicker palette with specified columns. Defaults to 10
disabled	It is used to enable / disable ColorPicker component. If it is disabled the ColorPicker popup won't open. Defaults to false
mode	It is used to render the ColorPicker with the specified mode. Defaults to 'Picker'
modeSwitcher	It is used to show / hide the mode switcher button of ColorPicker component. Defaults to true
showButtons	It is used to show / hide the control buttons (apply / cancel) of ColorPicker component. Defaults to true

DropDownButton

Getting Started

This section briefly explains how to create a simple **DropDownButton** component and configure its available functionalities in React.

Dependencies

The following list of dependencies are required to use the **DropDownButton** component in your application.

```

`js
|-- @syncfusion/ej2-react-splitbuttons
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-splitbuttons

```

```
|-- @syncfusion/ej2-popups  
|-- @syncfusion/ej2-buttons  
,
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash  
npm install -g create-react-app  
,
```

To set-up a React application in TypeScript environment, run the following command.

```
`bash  
npx create-react-app my-app --template typescript  
cd my-app  
npm start  
,
```

To set-up a React application in JavaScript environment, run the following command.

```
`bash  
npx create-react-app my-app  
cd my-app  
npm start  
,
```

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install DropDownButton component, use the following command

```
`bash  
npm install @syncfusion/ej2-react-splitbuttons --save  
,
```

Adding CSS reference

Import the DropDownButton component required CSS references as follows in `src/App.css`.

```
`css  
  
/ import the DropDownButton dependency styles /  
@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";
`
```

Adding DropDownButton component

Now, you can start adding DropDownButton component in the application. For getting started, add the DropDownButton component in `src/App.tsx` file using following code.

Add the below code in the `src/App.tsx` to initialize the DropDownButton.

```
`ts
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import './App.css';
enableRipple(true);
// To render DropDownButton.
function App() {
  return (
    <div style={{marginTop: '150px'}}>
      <DropDownButtonComponent id="element" />
    </div>
  );
}
export default App;
`
```

Binding data source

After initialization, populate the DropDownButton with data using the [items](#) property. Here, an array of string values is passed to the DropDownButton component.

```
`ts
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import './App.css';
enableRipple(true);
// To render DropDownButton.
function App() {
```

```
let items: ItemModel[] = [
{
text: 'Cut',
},
{
text: 'Copy',
},
{
text: 'Paste',
}];
return (
<div style={{marginTop: '150px'}}>
<DropDownButtonComponent id="element" items={items}> Clipboard </DropDownButtonComponent>
</div>
);
}
export default App;
`
```

Run the application

After completing the configuration required to render a basic DropDownButton, run the following command to display the output in your default browser.

```
`
npm start
`
```

The following example shows a basic Drop Button component.

APP.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items = [
        {
            text: 'Cut',
        },
        {
            text: 'Copy',
        },
    ];
    return (
        <div>
            <DropDownButtonComponent id="element" items={items}> Clipboard </DropDownButtonComponent>
        </div>
    );
}
```



```

        },
        {
            text: 'Paste',
        }
    ];
    return (<div>
        <DropDownButtonComponent id="element" items={items}> Clipboard
    </DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            text: 'Cut',
        },
        {
            text: 'Copy',
        },
        {
            text: 'Paste',
        }
    ];
    return (
        <div>
            <DropDownButtonComponent id="element" items = {items}> Clipboard
        </DropDownButtonComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

See Also

- [DropDownButton with icons](#)
- [How to hide dropdown arrow](#)
- [Dropdown popup with separator](#)

Icons in React Drop down button component

DropDownButton icons

DropDownButton can have an icon to provide the visual representation of the action. To place the icon on a DropDownButton, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the DropDownButton. You

can customize the icon's position by using the [iconPosition](#) property.

In the following example, the DropDownButton with default iconPosition and iconPosition as **Top** is showcased.

APP.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items = [
        {
            text: 'Edit',
        },
        {
            text: 'Delete',
        },
        {
            text: 'Mark as Read',
        },
        {
            text: 'Like Message',
        }
    ];
    return (
        <div>
            <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'>
                Message </DropDownButtonComponent>
            <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'
                iconPosition='Top'> Message </DropDownButtonComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {

```

```

        text: 'Edit',
      },
      {
        text: 'Delete',
      },
      {
        text: 'Mark as Read',
      },
      {
        text: 'Like Message',
      }
    ]];
    return (
      <div>
        <DropDownButtonComponent items = {items} iconCss='ddb-icons e-
message'> Message </DropDownButtonComponent>
        <DropDownButtonComponent items = {items} iconCss='ddb-icons e-
message' iconPosition = 'Top'> Message </DropDownButtonComponent>
      </div>
    );
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Icon only DropDownButton

Icon only DropDownButton can be achieved by using [iconCss](#) property and to hide drop down arrow

`e-caret-hide` class is added using [cssClass](#) property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'New tab'
    },
    {
      text: 'New window'
    },
    {
      text: 'New incognito window'
    },
    {
      separator: true
    },
    {
      text: 'Print'
    },
    {
      text: 'Cast'
    },
  ],

```

```

        {
            text: 'Find'
        }
    ];
    return (<div>
        <DropDownButtonComponent items={items} iconCss='e-icons e-menu'
        cssClass='e-caret-hide'></DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            text: 'New tab'
        },
        {
            text: 'New window'
        },
        {
            text: 'New incognito window'
        },
        {
            separator: true
        },
        {
            text: 'Print'
        },
        {
            text: 'Cast'
        },
        {
            text: 'Find'
        }
    ];
    return (
        <div>
            <DropDownButtonComponent items = {items} iconCss='e-icons e-menu'
            cssClass= 'e-caret-hide'></DropDownButtonComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

DropDownButton with sprite image

Sprite images can be loaded in DropDownButton instead of font icons using [iconCss](#) property.

In this following example, `e-image` class is added with background url of the sprite image along with X and Y positions. The `width` and `height` of the element set as `32px`.

APP.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items = [
        {
            text: 'Display Settings'
        },
        {
            text: 'System Settings'
        },
        {
            text: 'Additional Settings'
        }
    ];
    return (
        <DropDownButtonComponent items={items} iconCss='e-icons e-image'
        cssClass='e-caret-hide' />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            text: 'Display Settings'
        },
        {
            text: 'System Settings'
        },
        {
            text: 'Additional Settings'
        }
    ];
    return (
        <div>
```

```

        <DropDownButtonComponent items = {items} iconCss='e-icons e-image'
        cssClass= 'e-caret-hide' />
      </div>
    );
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('button'));

```

The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element. You can also use third party icons on the DropDownButton using the [iconCss](#) property.

Vertical button

Vertical Button in DropDownButton can be achieved by adding **e-vertical** class using **cssClass** property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    },
    {
      text: 'Mark as Read'
    },
    {
      text: 'Like Message'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'
      cssClass='e-vertical' iconPosition='Top'> Message </DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {

```

```

let items: ItemModel[] = [
  {
    text: 'Edit'
  },
  {
    text: 'Delete'
  },
  {
    text: 'Mark as Read'
  },
  {
    text: 'Like Message'
  }
];
return (
  <div>
    <DropDownButtonComponent items = {items} iconCss='ddb-icons e-message'
    cssClass= 'e-vertical' iconPosition='Top'> Message </DropDownButtonComponent>
  </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element. You can also use third party icons on the DropDownButton using the [iconCss](#) property.

See Also

- [Dropdown popup with icons](#)
- [Customized icon size](#)

Popup items in React Drop down button component

Icons

The Popup action item have an icon/image in it to provide visual representation of the action. To place the icon on a popup item,

set the [iconCss](#) property to **e-icons** with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for Edit, Delete, Mark As Read and Like Message menu items are added using the iconCss property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {

```

```

        text: 'Edit',
        iconCss: 'ddb-icons e-edit'
    },
    {
        text: 'Delete',
        iconCss: 'ddb-icons e-delete'
    },
    {
        text: 'Mark As Read',
        iconCss: 'ddb-icons e-read'
    },
    {
        text: 'Like Message',
        iconCss: 'ddb-icons e-like'
    }
];
return (<div>
    <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'>
Message </DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { ButtonComponent } from '@syncfusion/ej2-react-buttons';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            text: 'Edit',
            iconCss: 'ddb-icons e-edit'
        },
        {
            text: 'Delete',
            iconCss: 'ddb-icons e-delete'
        },
        {
            text: 'Mark As Read',
            iconCss: 'ddb-icons e-read'
        },
        {
            text: 'Like Message',
            iconCss: 'ddb-icons e-like'
        }
    ];
    return (
        <div>
            <DropDownButtonComponent items = {items} iconCss= 'ddb-icons e-
message'> Message </DropDownButtonComponent>

```



```

    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Navigations

Actions in DropDownButton is usage to navigate to the other web page when action item is clicked. This can be achieved by providing link to the action item using the `url` property. In the following sample, Navigation URL for Flipkart, Amazon, and Snapdeal action items are added using the `url` property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Flipkart',
      url: 'https://www.google.co.in/search?q=flipkart'
    },
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Amazon',
      url: 'https://www.google.co.in/search?q=amazon'
    },
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Snapdeal',
      url: 'https://www.google.co.in/search?q=snapdeal'
    }
  ];
  function itemBeforeEvent(args) {
    args.element.getElementsByTagName('a')[0].setAttribute('target',
    '_blank');
  }
  return (
    <div>
      <DropDownButtonComponent items={items} iconCss='e-cart-icon e-
shopping' beforeItemRender={itemBeforeEvent}> Shop By
    </DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel, MenuEventArgs } from
 '@syncfusion/ej2-react-splitbuttons';

```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Flipkart',
      url: 'https://www.google.co.in/search?q=flipkart'
    },
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Amazon',
      url: 'https://www.google.co.in/search?q=amazon'
    },
    {
      iconCss: 'e-cart-icon e-link',
      text: 'Snapdeal',
      url: 'https://www.google.co.in/search?q=snapdeal'
    }
  ];
  function itemBeforeEvent(args: MenuEventArgs) {
    args.element.getElementsByTagName('a')[0].setAttribute('target',
    '_blank');
  }
  return (
    <div>
      <DropDownButtonComponent items = {items} iconCss= 'e-cart-icon e-
shopping' beforeItemRender={itemBeforeEvent} > Shop By
    </DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

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.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Edit'
    },
  ],

```

```

        {
            text: 'Cut'
        }
    ];
    function itemBeforeEvent(args) {
        if (args.item.text === 'Edit') {
            args.element.innerHTML = '<span></span><div><b>Paste
Text</b><div>Provides option to paste only the<br>selected
text.</div></div>';
            args.element.style.height = '80px';
            const span = args.element.children[0];
            span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
            const div = args.element.children[1];
            div.setAttribute('class', 'e-div-align');
        }
        else {
            args.element.innerHTML = '<span></span><div><b>Paste
Special</b><div>Provides options to paste formulas,<br> values, comments,
validations etc...</div></div>';
            args.element.style.height = '80px';
            const span = args.element.children[0];
            span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
            const div = args.element.children[1];
            div.setAttribute('class', 'e-div-align');
        }
    }
    return (<div>
        <DropDownButtonComponent items={items} iconCss='e-ddb-icons e-
paste' cssClass='e-vertical' iconPosition='Top'
beforeItemRender={itemBeforeEvent}>Paste</DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel, MenuEventArgs } from
 '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items:ItemModel[] = [
        {
            text: 'Edit'
        },
        {
            text: 'Cut'
        }
    ];

    function itemBeforeEvent (args: MenuEventArgs) {
        if (args.item.text === 'Edit') {

```

```

        args.element.innerHTML = '<span></span><div><b>Paste
Text</b><div>Provides option to paste only the<br>selected
text.</div></div>';
        args.element.style.height = '80px';
        const span = args.element.children[0];
        span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
        const div = args.element.children[1];
        div.setAttribute('class', 'e-div-align');
    } else {
        args.element.innerHTML = '<span></span><div><b>Paste
Special</b><div>Provides options to paste formulas,<br> values, comments,
validations etc...</div></div>';
        args.element.style.height = '80px';
        const span = args.element.children[0];
        span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
        const div = args.element.children[1];
        div.setAttribute('class', 'e-div-align');
    }
}
return (
    <div>
        <DropDownButtonComponent items = {items} iconCss= 'e-ddb-icons
e-paste' cssClass='e-vertical' iconPosition='Top'
beforeItemRender={itemBeforeEvent} >Paste</DropDownButtonComponent>
    </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Popup Templating

The whole popup can be customized as per the requirement and it can be customized by handling it in [target](#) property.

In the following sample, the whole popup item is customized as table template by giving `div` as target and it can be achieved

using `target` property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    function itemBeforeEvent(args) {
        args.element.style.height = '105px';
    }
    return (<div>
        <div id="target">
            <table>
                <caption><b>Insert Table</b></caption>
                <tbody>

```

```

        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
        <tr className='e-row'>
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
          <td className='e-data' />
        </tr>
      </tbody>
    </table>
  </div>
  <DropDownButtonComponent target='#target' iconCss='e-icons e-table'
    iconPosition='Top' cssClass='e-vertical'
    beforeItemRender={itemBeforeEvent}>Table</DropDownButtonComponent>
  </div>);
}
export default App;

```

```
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, MenuEventArgs } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  function itemBeforeEvent(args: MenuEventArgs) {
    args.element.style.height = '105px';
  }
  return (
    <div>
      <div id="target">
        <table>
          <caption><b>Insert Table</b></caption>
          <tbody>
            <tr className='e-row'>
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
            </tr>
            <tr className='e-row'>
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
            </tr>
            <tr className='e-row'>
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
            </tr>
            <tr className='e-row'>
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
            </tr>
            <tr className='e-row'>
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
              <td className='e-data' />
            </tr>
          </tbody>
        </table>
      </div>
    </div>
  );
}
```

```

        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
      </tr>
      <tr className='e-row'>
        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
        <td className='e-data' />
      </tr>
    </tbody>
  </table>
</div>
  <DropDownButtonComponent target='#target' iconCss='e-icons e-
table' iconPosition='Top' cssClass='e-vertical'
beforeItemRender={itemBeforeEvent} >Table</DropDownButtonComponent>
</div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

Separator

The Separators are the horizontal lines that are used to separate the popup items. You cannot select the separators. You can enable separators to group the popup items using the `separator` property.

In the following sample, cut, copy, and paste popup items are grouped using the separator property:

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      iconCss: 'e-db-icons e-cut',
      text: 'Cut'
    },
    {
      iconCss: 'e-icons e-copy',
      text: 'Copy'
    },
    {
      iconCss: 'e-db-icons e-paste',
      text: 'Paste'
    },
    {
      separator: true
    },
  ],
```

```

        {
            iconCss: 'e-db-icons e-font',
            text: 'Font'
        },
        {
            iconCss: 'e-db-icons e-paragraph',
            text: 'Paragraph'
        }
    ];
    return (<div>
        <DropDownButtonComponent items={items} iconCss='e-icons e-edit'>
Clipboard </DropDownButtonComponent>
        </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            iconCss: 'e-db-icons e-cut',
            text: 'Cut'
        },
        {
            iconCss: 'e-icons e-copy',
            text: 'Copy'
        },
        {
            iconCss: 'e-db-icons e-paste',
            text: 'Paste'
        },
        {
            separator: true
        },
        {
            iconCss: 'e-db-icons e-font',
            text: 'Font'
        },
        {
            iconCss: 'e-db-icons e-paragraph',
            text: 'Paragraph'
        }
    ];
    return (
        <div>
            <DropDownButtonComponent items = {items} iconCss= 'e-icons e-edit'>
Clipboard </DropDownButtonComponent>
        </div>
    );
}

```



```
);  
}  
export default App;  
ReactDOM.render(<App />, document.getElementById('button'));
```

See Also

- [Integration with ListView component](#)

Accessibility in React DropDownButton component

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

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the DropDownButton component as **button**, DropDownButton popup as **menu**, and the dropdown popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability of the popup element. |

| **aria-expanded** | Indicates whether the popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

Keyboard interaction

The Dropdown button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the DropDownButton component.

| Press | To do this |

| --- | --- |

| **Esc** | Closes the popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Space** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| **Alt + Up Arrow** | Closes the popup. |

| **Alt + Down Arrow** | Opens the popup. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

Style and appearance in React Drop down button component

To modify the DropDownButton appearance, you need to override the default CSS of DropDownButton component. Please find the list of CSS classes and its corresponding section in DropDownButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

|.e-dropdown-btn|To customize the dropdown button
|.e-dropdown-btn:hover|To customize the dropdown button on hover
|.e-dropdown-btn.e-active|To customize the dropdown button on active
|.e-dropdown-popup|To customize the dropdown button pop up
|.e-dropdown-popup ul .e-item:hover|To customize the dropdown button pop up items on hover
|.e-dropdown-popup ul .e-item:active|To customize the dropdown button pop up items on active

How To

Change caret icon in React Drop down button component

Dropdown arrow can be customized on popup open and close. It can be handled in [beforeOpen](#) and [beforeClose](#) event.

In the following example, the up arrow is updated on popup close and down arrow is updated on popup open using `beforeOpen` and `beforeClose` event by adding and removing `e-caret-up` class.

APP.JSX

```
{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let ddb;
  let items = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
};
```

```

// To update up arrow with `e-caret-up` class.
function beforeOpen() {
    ddb.cssClass = 'e-caret-up';
}
// To remove `e-caret-up` class.
function beforeClose() {
    ddb.cssClass = '';
}
return (<div>
    <DropDownButtonComponent ref={ (scope) => { ddb = scope; }}
    items={items} beforeOpen={beforeOpen} beforeClose={beforeClose}>
    Clipboard</DropDownButtonComponent>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
{% endraw %}

```

APP.TSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let ddb: DropDownButtonComponent;
    let items: ItemModel[] = [
        {
            text: 'Cut'
        },
        {
            text: 'Copy'
        },
        {
            text: 'Paste'
        }
    ];
    // To update up arrow with `e-caret-up` class.
    function beforeOpen () {
        ddb.cssClass = 'e-caret-up';
    }
    // To remove `e-caret-up` class.
    function beforeClose () {
        ddb.cssClass = '';
    }

    return (
        <div>
            <DropDownButtonComponent ref={ (scope) => { ddb = scope as
            DropDownButtonComponent; }} items = {items} beforeOpen={beforeOpen}
            beforeClose={beforeClose}> Clipboard</DropDownButtonComponent>
        </div>
    );
}

```

```

    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('button'));
    {% endraw %}

```

Create dropdownbutton with rounded corner in React Drop down button component

DropDownButton with rounded corner can be achieved by adding `border-radius` CSS property to button element.

In the following example, `e-round-corner` class is defined with `5px border-radius` property and added that class to button element using `cssClass` property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items = [
        {
            text: 'Cut'
        },
        {
            text: 'Copy'
        },
        {
            text: 'Paste'
        }
    ];
    return (<div>
        <DropDownButtonComponent items={items} cssClass='e-round-
corner'>Clipboard</DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-
splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items: ItemModel[] = [
        {
            text: 'Cut'
        },
        {

```

```

        text: 'Copy'
      },
      {
        text: 'Paste'
      }
    ];
    return (
      <div>
        <DropDownButtonComponent items = {items} cssClass='e-round-
corner'>Clipboard</DropDownButtonComponent>
      </div>
    );
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

Create right to left dropdownbutton in React Drop down button component

DropDownButton component has RTL support. This can be achieved by setting [enableRtl](#) as true.

The following example illustrates how to enable right-to-left support in DropDownButton component.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    },
    {
      text: 'Mark as Read'
    },
    {
      text: 'Like Message'
    }
  ];
  return (<div>
    <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'
enableRtl={true}> Message </DropDownButtonComponent>
  </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-
splitbuttons';
```

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    },
    {
      text: 'Mark as Read'
    },
    {
      text: 'Like Message'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items = {items} iconCss= 'ddb-icons e-message'
enableRtl={true}> Message </DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

Customize icon and width in React Drop down button component

Width of the DropDownButton can be customized by setting required width to the dropdown element.

The following UI can be achieved by setting [iconPosition](#) as **Top**, width as **85px** and size of the font icon as **40px** by adding **e-custom** class.

APP.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Find'
    },
    {
      text: 'Replace'
    },
    {
      text: 'Go To'
    },
    {
      text: 'Go To Special'
    }
  ]
```

```

];
return (<div>
  <DropDownButtonComponent items={items} iconCss='e-icons e-search'
  cssClass='e-custom e-vertical' iconPosition='Top'>Find &
  Select</DropDownButtonComponent>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      text: 'Find'
    },
    {
      text: 'Replace'
    },
    {
      text: 'Go To'
    },
    {
      text: 'Go To Special'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items = {items} iconCss='e-icons e-search'
      cssClass='e-custom e-vertical' iconPosition='Top'>Find &
      Select</DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Disable a dropdownbutton in React Drop down button component

DropDownButton component can be enabled/disabled by giving [disabled](#) property. To disable DropDownButton component, the disabled property can be set as `true`.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);

```



```
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    },
    {
      text: 'Mark as Read'
    },
    {
      text: 'Like Message'
    }
  ];
  return (<div>
    <DropDownButtonComponent items={items} iconCss='ddb-icons e-message'
disabled={true}> Message </DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      text: 'Edit'
    },
    {
      text: 'Delete'
    },
    {
      text: 'Mark as Read'
    },
    {
      text: 'Like Message'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items = {items} iconCss= 'ddb-icons e-message'
disabled={true}> Message </DropDownButtonComponent>
      </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

Group popup items with listview component in React Drop down button component

Header in popup items is possible in DropDownButton by templating entire popup with ListView. Create ListView with id `#listview` and provide it as a `target` for DropDownButton.

In the following example, ListView element is given as `target` to DropDownButton and header can be achieved by `groupBy` property.

APP.JSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let data = [
        { class: 'data', text: 'Print', id: 'data1', category: 'Customize Quick Access Toolbar' },
        { class: 'data', text: 'Save As', id: 'data2', category: 'Customize Quick Access Toolbar' },
        { class: 'data', text: 'Update Folder', id: 'data3', category: 'Customize Quick Access Toolbar' },
        { class: 'data', text: 'Reply', id: 'data4', category: 'Customize Quick Access Toolbar' }
    ];
    let field = { text: 'text', groupBy: 'category' };
    return (
        <div>
            <ListViewComponent id="listview" dataSource={data} fields={field} showCheckBox={true}/>
            <DropDownButtonComponent target="#listview" iconCss='e-icons e-down' cssClass='e-caret-hide'/>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { ListViewComponent } from '@syncfusion/ej2-react-lists';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let data: Array<{ [key: string]: string; }> = [
        { class: 'data', text: 'Print', id: 'data1', category: 'Customize Quick Access Toolbar' },
        { class: 'data', text: 'Save As', id: 'data2', category: 'Customize Quick Access Toolbar' },
    ];
}
```

```

    { class: 'data', text: 'Update Folder', id: 'data3', category: 'Customize Quick Access Toolbar' },
    { class: 'data', text: 'Reply', id: 'data4', category: 'Customize Quick Access Toolbar' }
  ];
  let field: { [key: string]: string; } = { text: 'text', groupBy: 'category' };

  return (
    <div>
      <ListViewComponent id="listview" dataSource={data} fields={field} showCheckBox={true}/>
      <DropDownButtonComponent target="#listview" iconCss='e-icons e-down' cssClass='e-caret-hide' />
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Hide dropdown arrow in React Drop down button component

You can hide the dropdown arrow from the DropDownButton by adding class `e-caret-hide` to DropDownButton element using `cssClass` property.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items={items} cssClass='e-caret-hide'>
        Clipboard
      </DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

APP.TSX

```

import { enableRipple } from '@syncfusion/ej2-base';

```

```
import { DropDownButtonComponent, ItemModel } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
  return (
    <div>
      <DropDownButtonComponent items = {items} cssClass='e-caret-hide'>
        Clipboard</DropDownButtonComponent>
      </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

Open a dialog on popup item click in React Drop down button component

This section explains about how to open a dialog on DropdownButton popup item click. This can be achieved by handling dialog open in [select](#) event of the DropdownButton.

In the following example, Dialog will open while selecting **Other Folder...** item.

APP.JSX

```
{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let position = { X: 100, Y: 100 };
  let dialog;
  let alertDlgButtons = [{
    buttonModel: {
      content: 'Submit',
      cssClass: 'e-flat',
      isPrimary: true
    },
    'click': () => {
      dialog.hide();
    }
  }];
}
```

```

let items = [
  {
    text: 'Archive'
  },
  {
    text: 'Inbox'
  },
  {
    text: 'HR Portal'
  },
  {
    separator: true
  },
  {
    text: 'Other Folder...'
  },
  {
    text: 'Copy to Folder'
  }
];
// To open dialog on selecting 'Other Folder...' item.
function onSelect(args) {
  if (args.item.text === 'Other Folder...') {
    dialog.show();
  }
}
return (<div>
  <DialogComponent ref={(scope) => { dialog = scope; }} width='250px'
id='dialog' content='Move Items To "Web Team"' header='Move Items'
buttons={alertDlgButtons} position={position} visible={false}/>
  <DropDownButtonComponent items={items} select={onSelect} iconCss='ddb-
icons e-folder' cssClass='e-vertical'
iconPosition='Top'>Move</DropDownButtonComponent>
</div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
{% endraw %}

```

APP.TSX

```

{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { DialogComponent } from '@syncfusion/ej2-react-popups';
import { DropDownButtonComponent, ItemModel, MenuEventArgs } from
 '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let position: any = {X: 100, Y: 100};
  let dialog: DialogComponent;
  let alertDlgButtons: any = [{
    buttonModel: {
      content: 'Submit',

```

```

        cssClass: 'e-flat',
        isPrimary: true
    },
    'click' : () => {
        dialog.hide()
    }
}];
let items: ItemModel[] = [
    {
        text: 'Archive'
    },
    {
        text: 'Inbox'
    },
    {
        text: 'HR Portal'
    },
    {
        separator: true
    },
    {
        text: 'Other Folder...'
    },
    {
        text: 'Copy to Folder'
    }
];
// To open dialog on selecting `Other Folder...` item.
function onSelect (args: MenuEventArgs) {
    if (args.item.text === 'Other Folder...') {
        dialog.show();
    }
}

return (
    <div>
        <DialogComponent ref={ (scope) => { dialog = scope as DialogComponent;
    }} width='250px' id='dialog' content='Move Items To "Web Team"' header='Move
Items' buttons={alertDlgButtons} position={position} visible={false}/>
        <DropDownButtonComponent items = {items} select={onSelect}
iconCss='ddb-icons e-folder' cssClass='e-vertical'
iconPosition='Top'>Move</DropDownButtonComponent>
    </div>
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
{% endraw %}

```

Position popup open in React Drop down button component

Popup open position can be changed according to the requirement. Popup open position can be changed in [open](#) event by setting **top** and **left** for the popup element.

In the following example, the **top** position of the popup element is changed in **open** event.

APP.JSX

```
{% raw %}
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let ddb;
    let items = [
        {
            text: 'Cut'
        },
        {
            text: 'Copy'
        },
        {
            text: 'Paste'
        }
    ];
    // To open popup in particular position.
    function onOpen(args) {
        const elem = args.element.parentElement;
        elem.style.top = ddb.element.getBoundingClientRect().top -
        elem.offsetHeight + 'px';
    }
    return (<div>
        <DropDownButtonComponent ref={ (scope) => { ddb = scope; }}
        items={items} open={onOpen} cssClass='e-caret-
        up'>Clipboard</DropDownButtonComponent>
    </div>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
{% endraw %}
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel, OpenCloseMenuEventArgs } from
 '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let ddb: DropDownButtonComponent;
    let items: ItemModel[] = [
        {
            text: 'Cut'
        },
        {
            text: 'Copy'
        },
        {
            text: 'Paste'
        }
    ];
}
```

```

    }];
    // To open popup in particular position.
    function onOpen (args: OpenCloseMenuEventArgs) {
        const elem = (args.element.parentElement as HTMLElement);
        elem.style.top = ddb.element.getBoundingClientRect().top -
        elem.offsetHeight + 'px';
    }

    return (
        <div>
            <DropDownButtonComponent ref= {(scope) => { ddb = scope as
DropDownButtonComponent; } } items = {items} open={onOpen} cssClass='e-caret-
up'>Clipboard</DropDownButtonComponent>
        </div>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));

```

Underline a character in the item text in React Drop down button component

Underline a particular character in a text can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

In the following example, **C** is underlined in the text **Copy**.

APP.JSX

```

import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent } from '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
    let items = [
        {
            text: 'Cut'
        },
        {
            text: 'Copy'
        },
        {
            text: 'Paste'
        }
    ];
    function itemRender(args) {
        if (args.item.text === 'Copy') {
            // To underline a particular text.
            args.element.innerHTML = '<u>C</u>opy';
        }
    }
    return (<div>
        <DropDownButtonComponent items={items}
beforeItemRender={itemRender}>Clipboard</DropDownButtonComponent>
    </div>);
}

```



```
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

APP.TSX

```
import { enableRipple } from '@syncfusion/ej2-base';
import { DropDownButtonComponent, ItemModel, MenuEventArgs } from
 '@syncfusion/ej2-react-splitbuttons';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
enableRipple(true);
// To render DropDownButton.
function App() {
  let items: ItemModel[] = [
    {
      text: 'Cut'
    },
    {
      text: 'Copy'
    },
    {
      text: 'Paste'
    }
  ];
  function renderItem(args: MenuEventArgs) {
    if (args.item.text === 'Copy') {
      // To underline a particular text.
      args.element.innerHTML = '<u>C</u>opy';
    }
  }
  return (
    <div>
      <DropDownButtonComponent items = {items}
beforeItemRender={itemRender}>Clipboard</DropDownButtonComponent>
    </div>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('button'));
```

DropDownList

Getting Started

Getting Started

This section briefly explains how to create a simple [Link to the Video](#) component and configure its available functionalities in React.

To get start quickly with React DropDownList, you can check on this video:

Dependencies

The following list of dependencies are required to use the **DropDownList** component in your application.

```
`javascript`
```

```
|-- @syncfusion/ej2-react-dropdowns
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
|-- @syncfusion/ej2-buttons
\
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash
npm install -g create-react-app
\
```

Start a new project using create-react-app command as follows

```
create-react-app quickstart --scripts-version=react-scripts-ts
cd quickstart
\
```

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install DropDownList component, use the following command

```
`bash
npm install @syncfusion/ej2-react-dropdowns --save
\
```

Adding DropDownList component

Now, you can start adding DropDownList component in the application. For getting started, add the DropDownList component in `src/App.tsx` file using following code.

Add the below code in the `src/App.tsx` to initialize the DropDownList.

```
[Class-component]
```

```
`ts
```

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
export default class App extends React.Component<{}, {}> {
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id='ddlelement'/>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

[Functional-component]

```
`ts
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id='ddlelement'/>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

[Adding CSS reference](#)

Import the DropDownList component required CSS references as follows in `src/App.css`.

```
`css
/ import the DropDownList dependency styles /
@import "../node_modules/@syncfusion/ej2-base/styles/material.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-dropdowns/styles/material.css";
```

Binding data source

After initialization, populate the DropDownList with data using the [dataSource](#) property. Here, an array of string values is passed to the DropDownList component.

[Class-component]

```
`ts
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the array of data
  private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" dataSource={this.sportsData} />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

```
`ts
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the array of data
  const sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData} />
  );
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Run the application

After completing the configuration required to render a basic DropDownList, run the following command to display the output in your default browser.

```
npm start
```

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
placeholder="Select a game"/>);
        }
    }
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
```

```
import * as ReactDOM from 'react-dom';
function App() {
  // define the array of data
  const sportsData = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData}
placeholder="Select a game"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the array of data
  const sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData}
placeholder="Select a game" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Configure the Popup List

By default, the width of the popup list automatically adjusts according to the DropDownList input element's width, and the height of the popup list has '300px'.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) properties respectively.

In the following sample, popup list's width and height are configured.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of data
  sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
'Hockey', 'Rugby', 'Snooker', 'Tennis'];
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
popupHeight="200px" popupWidth="250px" placeholder="select a game"/>);
  }
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
    'Football', 'Golf', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
            dataSource={this.sportsData} popupHeight="200px" popupWidth="250px"
            placeholder="select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football',
    'Golf', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={sportsData}
        popupHeight="200px" popupWidth="250px" placeholder="select a game"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData: string[] = ['Badminton', 'Basketball', 'Cricket',
    'Football', 'Golf', 'Hockey', 'Rugby', 'Snooker', 'Tennis'];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={sportsData}
        popupHeight="200px" popupWidth="250px" placeholder="select a game" />
    );
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [How to bind the data](#)

Getting Started with the React DropDownList Component in the Preact

This article provides a step-by-step guide for setting up a [Preact](#) project and integrating the Syncfusion React DropDownList component.

Preact is a fast and lightweight JavaScript library for building user interfaces. It's often used as an alternative to larger frameworks like React. The key difference is that Preact is designed to be smaller in size and faster in performance, making it a good choice for projects where file size and load times are critical factors.

Prerequisites

[System requirements for Syncfusion React UI components](#)

Set up the Preact project

To create a new **Preact** project, use one of the commands that are specific to either NPM or Yarn.

```
`bash
```

```
npm init preact
```

```
,
```

or

```
`bash
```

```
yarn init preact
```

```
,
```

Using one of the above commands will lead you to set up additional configurations for the project, as below:

1\ Define the project name: We can specify the name of the project directly. Let's specify the name of the project as **my-project** for this article.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project directory:

```
| my-project
```

```
—
```

```
,
```

2\ Choose **JavaScript** as the framework variant to build this Preact project using JavaScript and React.


```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Project language:

```
| > JavaScript
```

```
| TypeScript
```

```
—
```

```
`
```

3\). Then configure the project as below for this article.

```
`bash
```

```
T Preact - Fast 3kB alternative to React with the same modern API
```

```
|
```

- Use router?

```
| Yes / > No
```

```
—
```

```
|
```

- Prerender app (SSG)?

```
| Yes / > No
```

```
—
```

```
|
```

- Use ESLint?

```
| Yes / > No
```

```
—
```

```
`
```

5\). Upon completing the aforementioned steps to create `my-project`, run the following command to jump into the project directory:

```
`bash
```

```
cd my-project
```

```
`
```

Now that `my-project` is ready to run with default settings, let's add Syncfusion components to the project.

Add the Syncfusion React packages

Syncfusion React component packages are available at [npmjs.com](https://www.npmjs.com). To use Syncfusion React components in the project, install the corresponding npm package.

This article uses the [React DropDownList component](#) as an example. To use the React DropDownList component in the project, the `@syncfusion/ej2-react-dropdowns` package needs to be installed using the following command:

```
`bash
npm install @syncfusion/ej2-react-dropdowns --save
`
```

or

```
`bash
yarn add @syncfusion/ej2-react-dropdowns
`
```

Import Syncfusion CSS styles

You can import themes for the Syncfusion React component in various ways, such as using CSS or SASS styles from npm packages, CDN, CRG and [Theme Studio](#). Refer to [themes topic](#) to know more about built-in themes and different ways to refer to theme's in a React project.

In this article, the `Material 3` theme is applied using CSS styles, which are available in installed packages. The necessary `Material 3` CSS styles for the DropDownList component and its dependents were imported into the `src/style.css` file.

~/SRC/STYLE.CSS

```
@import "../node_modules/@syncfusion/ej2-base/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-inputs/styles/material3.css";
@import "../node_modules/@syncfusion/ej2-react-
dropdowns/styles/material3.css";
```

The order of importing CSS styles should be in line with its dependency graph.

Add the Syncfusion React component

Follow the below steps to add the React DropDownList component to the Vite project:

1\ Before adding the DropDownList component to your markup, import the DropDownList component in the `src/index.jsx` file.

~/SRC/INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
```

2\ Then, define the DropDownList component with the [dataSource](#) property. Declare the values for the `dataSource` property.

~/SRC/INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import { render } from 'preact';
import './style.css';
function App() {
  // define the array of data
  const sportsData = ['Badminton', 'Basketball', 'Cricket', 'Football', 'Golf',
    'Hockey', 'Rugby', 'Snooker', 'Tennis'];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData}
    placeholder="Select a game"/>);
}
render(<App />, document.querySelector('#app'));
```

Run the project

To run the project, use the following command:

```
`bash
```

```
npm run dev
```

```
`
```

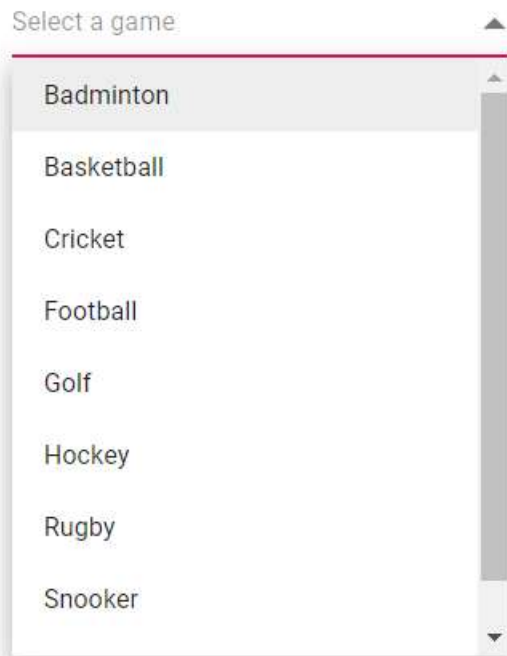
or

```
`bash
```

```
yarn run dev
```

```
`
```

The output will appear as follows:



See also

[Getting Started with the Syncfusion React UI Component](#)

Data binding in React Drop down list component

The DropDownList loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of `array` or `DataManager`.

The DropDownList also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

	Fields	Type	Description
--	--------	------	-------------

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

text	<code>string</code>	Specifies the display text of each list item.
------	---------------------	---

value	<code>number or string</code>	Specifies the hidden data value mapped to each list item that should contain a unique value.
-------	-------------------------------	--

groupBy	<code>string</code>	Specifies the category under which the list item has to be grouped.
---------	---------------------	---

iconCss	<code>string</code>	Specifies the icon class of each list item.
---------	---------------------	---

When binding complex data to the DropDownList, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The DropDownList has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
placeholder="Select a game"/>);
        }
    }
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of string
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football',
'Golf', 'Tennis'];
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
dataSource={this.sportsData} placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of string
    const sportsData = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    return (
        // specifies the tag for render the DropDownList component
```

```

    <DropDownListComponent id="ddlelement" dataSource={sportsData}
    placeholder="Select a game"/>;
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the array of string
  const sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
  'Tennis'];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData}
    placeholder="Select a game" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

2. Array of JSON data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Id** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
      fields={this.fields} placeholder="Select a game"/>;
    )
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // define the JSON of data
    private sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'Game', value: 'Id' };
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
            dataSource={this.sportsData} fields={this.fields} placeholder="Select a game"
            />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Game', value: 'Id' };
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={sportsData}
        fields={fields} placeholder="Select a game"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
```

```

    { Id: 'game3', Game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  const fields: object = { text: 'Game', value: 'Id' };
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" dataSource={sportsData}
    fields={fields} placeholder="Select a game" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

3. Array of Complex data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Code.Id` column and `Country.Name` column from complex data have been mapped to the `value` field and `text` field, respectively.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  countriesData = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Country.Name', value: 'Code.Id' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement"
      dataSource={this.countriesData} fields={this.fields} placeholder="Select a
      country"/>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // define the JSON of data

```



```

private countriesData: { [key: string]: Object }[] = [
    { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
    { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
    { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
    { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
    { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
    { Country: { Name: 'France' }, Code: { Id: 'FR' } }
];
// maps the appropriate column to fields property
private fields: object = { text: 'Country.Name', value: 'Code.Id' };
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement"
dataSource={this.countriesData} fields={this.fields} placeholder="Select a
country" />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

[Functional-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data
    const countriesData = [
        { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
        { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
        { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
        { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
        { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
        { Country: { Name: 'France' }, Code: { Id: 'FR' } }
    ];
    // maps the appropriate column to fields property
    const fields = { text: 'Country.Name', value: 'Code.Id' };
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={countriesData}
fields={fields} placeholder="Select a country"/>
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the JSON of data

```

```

const countriesData: { [key: string]: Object }[] = [
  { Country: { Name: 'Australia' }, Code: { Id: 'AU' } },
  { Country: { Name: 'Bermuda' }, Code: { Id: 'BM' } },
  { Country: { Name: 'Canada' }, Code: { Id: 'CA' } },
  { Country: { Name: 'Cameroon' }, Code: { Id: 'CM' } },
  { Country: { Name: 'Denmark' }, Code: { Id: 'DK' } },
  { Country: { Name: 'France' }, Code: { Id: 'FR' } }
];
// maps the appropriate column to fields property
const fields: object = { text: 'Country.Name', value: 'Code.Id' };
return (
  // specifies the tag for render the DropDownList component
  <DropDownListComponent id="ddlelement" dataSource={countriesData}
fields={fields} placeholder="Select a country" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
[% endraw %]

```

Binding remote data

The DropDownList supports retrieval of data from remote data services with the help of **DataManager** component. The [Query](#) property

is used to fetch data from the database and bind it to the DropDownList.

The following sample displays the first 6 contacts from “Customers” table of the **Northwind** Data Service.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // bind the Query instance to query property
  query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'ContactName', value: 'CustomerID' };
  // sort the resulted items
  sortOrder = 'Ascending';
  render() {
    return (
      // specifies the tag for render the DropDownList component

```

```

        <DropDownListComponent id="ddlelement" query={this.query}
        dataSource={this.customerData} fields={this.fields}
        sortOrder={this.sortOrder} placeholder="Select a customer"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new
    Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
            dataSource={this.customerData}
            fields={this.fields} sortOrder={this.sortOrder}
            placeholder="Select a customer" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });

```

```

});
// bind the Query instance to query property
const query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
// maps the appropriate column to fields property
const fields = { text: 'ContactName', value: 'CustomerID' };
// sort the resulted items
const sortOrder = 'Ascending';
return (
// specifies the tag for render the DropDownList component
<DropDownListComponent id="ddlelement" query={query}
dataSource={customerData} fields={fields} sortOrder={sortOrder}
placeholder="Select a customer"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
// bind the DataManager instance to dataSource property
const customerData: DataManager = new DataManager({
adaptor: new ODataV4Adaptor,
crossDomain: true,
url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
});
// bind the Query instance to query property
const query: Query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
// maps the appropriate column to fields property
const fields: object = { text: 'ContactName', value: 'CustomerID' };
// sort the resulted items
const sortOrder: SortOrder = 'Ascending';
return (
// specifies the tag for render the DropDownList component
<DropDownListComponent id="ddlelement" query={query}
dataSource={customerData}
fields={fields} sortOrder={sortOrder} placeholder="Select a customer"
/>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)
- [How to get the count of the data when using remote data](#)

- [How to achieve cascading](#)
- [How to add item in between the options](#)
- [How to remove an item](#)
- [How to preselect the items in dropdownlist](#)

Value binding in DropDownList

Value binding in the DropDownList control allows you to associate data values with each list item. This facilitates managing and retrieving selected values efficiently. The DropDownList component provides flexibility in binding both primitive data types and complex objects.

Primitive Data Types

The DropDownList control provides flexible binding capabilities for primitive data types like strings and numbers. You can effortlessly bind local primitive data arrays, fetch and bind data from remote sources, and even custom data binding to suit specific requirements. Bind the value of primitive data to the [value](#) property of the DropDownList.

Primitive data types include:

- String
- Number
- Boolean
- Null

The following sample shows the example for preselect values for primitive data type

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    constructor(props) {
        super(props);
        this.records = ["item 1", "item 2", "item 3", "item 4", "item 5", "item 6", "item 7", "item 8", "item 9", "item 10"];
    }
    fields = { text: 'text', value: 'id' };
    value = "item 5";
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" value={this.value} allowFiltering={false}
popupHeight="200px" >
                </DropDownListComponent>;
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: string[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        this.records = ["item 1", "item 2", "item 3", "item 4", "item 5", "item 6", "item 7", "item 8", "item 9", "item 10"];
    }
    private value: string = "item 1";
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
            value={this.value} placeholder="e.g. Item 1" allowFiltering={false}
            popupHeight="200px" >
                </DropDownListComponent>
            );
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));
```

Object Data Types

In the DropDownList control, object binding allows you to bind to a dataset of objects. When [allowObjectBinding](#) is enabled, the value of the control will be an object of the same type as the selected item in the [value](#) property. This feature seamlessly binds arrays of objects, whether sourced locally, retrieved from remote endpoints, or customized to suit specific application needs.

The following sample shows the example for preselect values for object data type

[Link to the Video](#)

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    fields = { text: 'text', value: 'id' };
    value = { text: 'Item 5', value: 'id5' };
    render() {
```

```

        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" value={this.value} allowObjectBinding={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
                </DropDownListComponent>;
        )
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    private value: { [key: string]: Object } = { text: 'Item 1', id: 'id1' }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
value={this.value} placeholder="e.g. Item 1" allowObjectBinding={true}
allowFiltering={false} fields={this.fields} popupHeight="200px" >
                </DropDownListComponent>
            );
        }
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Templates in React Drop down list component

The DropDownList has been provided with several options to customize each list item, group title, selected value, header, and footer elements.

To get started with React DropDownList templates, you can check on this video:

Item template

The content of each list item within the DropDownList can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

[Class-component]**INDEX.JSX**

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    sortOrder = 'Ascending';
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
            itemTemplate={this.itemTemplate = this.itemTemplate.bind(this)}
            dataSource={this.employeeData} sortOrder={this.sortOrder}
            fields={this.fields} placeholder="Select an employee"/>);
        // set the value to itemTemplate property
        itemTemplate(data) {
            return (<span><span className='name'>{data.FirstName}</span><span
            className='city'>{data.City}</span></span>);
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
```



```

private fields: object = { text: 'FirstName', value: 'EmployeeID' };
// sort the resulted items
private sortOrder: string = 'Ascending';
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={this.query}
itemTemplate={this.itemTemplate = this.itemTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee" />
    );
}
// set the value to itemTemplate property
private itemTemplate(data: any): JSX.Element {
    return (
        <span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={query}
itemTemplate={itemTemplate = itemTemplate.bind(this)}
dataSource={employeeData} sortOrder={sortOrder} fields={fields}
placeholder="Select an employee"/>
        // set the value to itemTemplate property
        function itemTemplate(data) {
            return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
        }
    );
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  const employeeData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // bind the Query instance to query property
  const query: Query = new Query().from('Employees').select(['FirstName',
  'City', 'EmployeeID']).take(6);
  // maps the appropriate column to fields property
  const fields: object = { text: 'FirstName', value: 'EmployeeID' };
  // sort the resulted items
  const sortOrder: string = 'Ascending';
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" query={query}
    itemTemplate={itemTemplate = itemTemplate.bind(this)}
    dataSource={employeeData} sortOrder={sortOrder} fields={fields}
    placeholder="Select an employee" />
  );
  // set the value to itemTemplate property
  function itemTemplate(data: any): JSX.Element {
    return (
      <span><span className='name'>{data.FirstName}</span><span
      className='city'>{data.City}</span></span>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Value template

The currently selected value that is displayed by default on the DropDownList input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the DropDownList input, which is separated by a hyphen.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
```

```

import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    sortOrder = 'Ascending';
    // set the value to itemTemplate property
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
valueTemplate={this.valueTemplate = this.valueTemplate.bind(this)}
query={this.query} itemTemplate={this.itemTemplate =
this.itemTemplate.bind(this)} sortOrder={this.sortOrder}
dataSource={this.employeeData} fields={this.fields} placeholder="Select an
employee"/>);
        }
        itemTemplate(data) {
            return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
        }
        // set the value to valueTemplate property
        valueTemplate(data) {
            return (<span>{data.FirstName} - {data.City}</span>);
        }
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'FirstName', value: 'EmployeeID' };

```

```

// sort the resulted items
private sortOrder: string = 'Ascending';
// set the value to itemTemplate property
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement"
valueTemplate={this.valueTemplate = this.valueTemplate.bind(this)}
query={this.query} itemTemplate={this.itemTemplate =
this.itemTemplate.bind(this)} sortOrder={this.sortOrder}
dataSource={this.employeeData} fields={this.fields} placeholder="Select an
employee" />
    );
}
private itemTemplate(data: any): JSX.Element {
    return (
        <span><span className='name'>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
    );
}
// set the value to valueTemplate property
private valueTemplate(data: any): JSX.Element {
    return (
        <span>{data.FirstName} - {data.City}</span>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    // set the value to itemTemplate property
    return (
        // specifies the tag for render the DropDownList component

```

```

    <DropDownListComponent id="ddlelement" valueTemplate={valueTemplate =
valueTemplate.bind(this)} query={query} itemTemplate={itemTemplate =
itemTemplate.bind(this)} sortOrder={sortOrder} dataSource={employeeData}
fields={fields} placeholder="Select an employee"/>;
    function itemTemplate(data) {
        return (<span><span className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
    // set the value to valueTemplate property
    function valueTemplate(data) {
        return (<span>{data.FirstName} - {data.City}</span>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder: string = 'Ascending';
    // set the value to itemTemplate property
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" valueTemplate={valueTemplate =
valueTemplate.bind(this)} query={query} itemTemplate={itemTemplate =
itemTemplate.bind(this)} sortOrder={sortOrder} dataSource={employeeData}
fields={fields} placeholder="Select an employee" />
    );
    function itemTemplate(data: any): JSX.Element {
        return (
            <span><span className='name'>{data.FirstName}</span><span className
='city'>{data.City}</span></span>
        );
    }
    // set the value to valueTemplate property
    function valueTemplate(data: any): JSX.Element {
        return (
            <span>{data.FirstName} - {data.City}</span>
        );
    }
}

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
 '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  employeeData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // form predicate to fetch the grouped data
  groupPredicate = new Predicate('City', 'equal', 'london').or('City',
    'equal', 'seattle');
  // bind the Query instance to query property
  query = new Query().from('Employees').select(['FirstName', 'City',
    'EmployeeID']).take(6).where(this.groupPredicate);
  // maps the appropriate column to fields property
  fields = { text: 'FirstName', value: 'EmployeeID', groupBy: 'City' };
  // sort the resulted items
  sortOrder = 'Ascending';
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" query={this.query}
groupTemplate={this.groupTemplate = this.groupTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee"/>);
    // set the value to groupTemplate
    groupTemplate(data) {
      return (<strong>{data.City}</strong>);
    }
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
```

```

import { DataManager, ODataV4Adaptor, Predicate, Query } from
'@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    private groupPredicate = new Predicate('City', 'equal',
'london').or('City', 'equal', 'seattle');
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6).where(this.groupPredicate);
    // maps the appropriate column to fields property
    private fields: object = { text: 'FirstName', value: 'EmployeeID', groupBy:
'City' };
    // sort the resulted items
    private sortOrder: string = 'Ascending';
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
groupTemplate={this.groupTemplate} = this.groupTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
fields={this.fields} placeholder="Select an employee" />
        );
    }
    // set the value to groupTemplate
    private groupTemplate(data: any): JSX.Element {
        return (
            <strong>{data.City}</strong>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
'@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,

```

```

        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    const groupPredicate = new Predicate('City', 'equal',
'london').or('City', 'equal', 'seattle');
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6).where(groupPredicate);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID', groupBy: 'City'
};
    // sort the resulted items
    const sortOrder = 'Ascending';
    return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" query={query}
groupTemplate={groupTemplate = groupTemplate.bind(this)}
dataSource={employeeData} sortOrder={sortOrder} fields={fields}
placeholder="Select an employee"/>);
    // set the value to groupTemplate
    function groupTemplate(data) {
        return (<strong>{data.City}</strong>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Predicate, Query } from
'@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // form predicate to fetch the grouped data
    const groupPredicate = new Predicate('City', 'equal', 'london').or('City',
'equal', 'seattle');
    // bind the Query instance to query property
    const query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6).where(groupPredicate);
    // maps the appropriate column to fields property
    const fields: object = { text: 'FirstName', value: 'EmployeeID', groupBy:
'City' };
    // sort the resulted items
    const sortOrder: string = 'Ascending';
    return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" query={query}
groupTemplate={groupTemplate = groupTemplate.bind(this)}

```



```

dataSource={employeeData} sortOrder={sortOrder} fields={fields}
placeholder="Select an employee" />
);
// set the value to groupTemplate
function groupTemplate(data: any): JSX.Element {
  return (
    <strong>{data.City}</strong>
  );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Header template

The header element is shown statically at the top of the popup list items within the DropDownList, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  employeeData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  // bind the Query instance to query property
  query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'FirstName', value: 'EmployeeID' };
  // sort the resulted items
  sortOrder = 'Ascending';
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" query={this.query}
headerTemplate={this.headerTemplate = this.headerTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
itemTemplate={this.itemTemplate = this.itemTemplate.bind(this)}
fields={this.fields} placeholder="Select an employee"/>
    );
    // set the value to header template
    headerTemplate(data) {
      return (
        <span className='head'><span
className='name'>Name</span><span className='city'>City</span></span>
      );
    }
  }
}

```

```

    // set the value to item template
    itemTemplate(data) {
        return (<span className='item'><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    private sortOrder: string = 'Ascending';
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
headerTemplate={this.headerTemplate = this.headerTemplate.bind(this)}
dataSource={this.employeeData} sortOrder={this.sortOrder}
itemTemplate={this.itemTemplate = this.itemTemplate.bind(this)}
fields={this.fields} placeholder="Select an employee" />
        );
    }
    // set the value to header template
    private headerTemplate(data: any): JSX.Element {
        return (
            <span className='head'><span className='name'>Name</span><span
className='city'>City</span></span>
        );
    }
    // set the value to item template
    private itemTemplate(data: any): JSX.Element {
        return (
            <span className='item' ><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>
        );
    }
}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder = 'Ascending';
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={query}
        headerTemplate={headerTemplate = headerTemplate.bind(this)}
        dataSource={employeeData} sortOrder={sortOrder} itemTemplate={itemTemplate =
        itemTemplate.bind(this)} fields={fields} placeholder="Select an employee"/>);
    // set the value to header template
    function headerTemplate(data) {
        return (<span className='head'><span
        className='name'>Name</span><span className='city'>City</span></span>);
    }
    // set the value to item template
    function itemTemplate(data) {
        return (<span className='item'><span
        className='name'>{data.FirstName}</span><span
        className='city'>{data.City}</span></span>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const employeeData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
```

```

        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Employees').select(['FirstName',
'City', 'EmployeeID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'FirstName', value: 'EmployeeID' };
    // sort the resulted items
    const sortOrder: string = 'Ascending';
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={query}
headerTemplate={headerTemplate} = headerTemplate.bind(this)}
dataSource={employeeData} sortOrder={sortOrder} itemTemplate={itemTemplate =
itemTemplate.bind(this)} fields={fields} placeholder="Select an employee" />
    );
    // set the value to header template
    function headerTemplate(data: any): JSX.Element {
        return (
            <span className='head'><span className='name'>Name</span><span
className='city'>City</span></span>
        );
    }
    // set the value to item template
    function itemTemplate(data: any): JSX.Element {
        return (
            <span className='item' ><span
className='name'>{data.FirstName}</span><span
className='city'>{data.City}</span></span>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Footer template

The DropDownList has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the DropDownList.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ["BasketBall", "Cricket", "Football", "Golf"];
    render() {
        return (
            // specifies the tag for render the DropDownList component

```

```

    <DropDownListComponent id="ddlelement"
    footerTemplate={this.footerTemplate = this.footerTemplate.bind(this)}
    dataSource={this.sportsData} placeholder="Select a game"/>);
    }
    // set the value to footer template
    footerTemplate(data) {
        return (<span className='foot'/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ["BasketBall", "Cricket", "Football",
    "Golf"];
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
            footerTemplate={this.footerTemplate = this.footerTemplate.bind(this)}
            dataSource={this.sportsData} placeholder="Select a game" />
            );
        }
        // set the value to footer template
        private footerTemplate(data: any): JSX.Element {
            return (
                <span className='foot'/>
            );
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData = ["BasketBall", "Cricket", "Football", "Golf"];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" footerTemplate={footerTemplate =
        footerTemplate.bind(this)} dataSource={sportsData} placeholder="Select a
        game"/>);
        // set the value to footer template
        function footerTemplate(data) {
            return (<span className='foot'/>);
        }
    }

```

```
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData: string[] = ["BasketBall", "Cricket", "Football",
"Golf"];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" footerTemplate={footerTemplate
= footerTemplate.bind(this)} dataSource={sportsData} placeholder="Select a
game" />
    );
    // set the value to footer template
    function footerTemplate(data: any): JSX.Element {
        return (
            <span className='foot' />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

No records template

The DropDownList is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of

[noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    data = [];
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
noRecordsTemplate={this.noRecordsTemplate =
this.noRecordsTemplate.bind(this)} dataSource={this.data} placeholder="Select
an item"/>
        );
        // set the value to noRecords template
        noRecordsTemplate(data) {
            return (<span className='norecord'> NO DATA AVAILABLE</span>);
        }
    }
}
```

```

    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
  // define the array of data
  private data: { [key: string]: Object }[] = [];
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement"
noRecordsTemplate={this.noRecordsTemplate} =
this.noRecordsTemplate.bind(this)} dataSource={this.data} placeholder="Select
an item" />
    );
  }
  //set the value to noRecords template
  private noRecordsTemplate(data: any): JSX.Element {
    return (
      <span className='norecord'> NO DATA AVAILABLE</span>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // define the array of data
  const data = [];
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement"
noRecordsTemplate={noRecordsTemplate} = noRecordsTemplate.bind(this)}
dataSource={data} placeholder="Select an item"/>);
  // set the value to noRecords template
  function noRecordsTemplate(data) {
    return (<span className='norecord'> NO DATA AVAILABLE</span>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';

```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const data: { [key: string]: Object }[] = [];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement"
noRecordsTemplate={noRecordsTemplate = noRecordsTemplate.bind(this)}
dataSource={data} placeholder="Select an item" />
    );
    // set the value to noRecords template
    function noRecordsTemplate(data: any): JSX.Element {
        return (
            <span className='norecord'> NO DATA AVAILABLE</span>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the DropDownList displays the notification.

[Class-component]

INDEX.JSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
actionFailureTemplate={this.failureTemplate =
this.failureTemplate.bind(this)} dataSource={this.customerData}
fields={this.fields} placeholder="Select a customer"/>
        );
        // set the value to action failure template
    }
}

```



```

        failureTemplate(data) {
            return (<span className='action-failure'> Data fetch get
fails</span>);
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    private query: Query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
actionFailureTemplate={this.failureTemplate} =
this.failureTemplate.bind(this)} dataSource={this.customerData}
fields={this.fields} placeholder="Select a customer" />
        );
    }
    // set the value to action failure template
    private failureTemplate(data: any): JSX.Element {
        return (
            <span className='action-failure'> Data fetch get fails</span>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData = new DataManager({

```

```

        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    const query = new Query().from('Customers').select(['ContactName',
    'CustomerID']).take(6);
    // maps the appropriate column to fields property
    const fields = { text: 'ContactName', value: 'CustomerID' };
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={query}
    actionFailureTemplate={failureTemplate = failureTemplate.bind(this)}
    dataSource={customerData} fields={fields} placeholder="Select a customer"/>);
    // set the value to action failure template
    function failureTemplate(data) {
        return (<span className='action-failure'> Data fetch get fails</span>);
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs/'
    });
    // bind the Query instance to query property
    const query: Query = new Query().from('Customers').select(['ContactName',
    'CustomerID']).take(6);
    // maps the appropriate column to fields property
    const fields: object = { text: 'ContactName', value: 'CustomerID' };
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" query={query}
    actionFailureTemplate={failureTemplate = failureTemplate.bind(this)}
    dataSource={customerData} fields={fields} placeholder="Select a customer" />
    );
    // set the value to action failure template
    function failureTemplate(data: any): JSX.Element {
        return (
            <span className='action-failure'> Data fetch get fails</span>
        );
    }
    ReactDOM.render(<App />, document.getElementById('sample'));
}

```

See Also

- [How to achieve filtering](#)
- [How to group the data using header](#)
- [How to show the list items with icon](#)
- [How to render tooltip for the options](#)

Virtualization in DropDown List

Dropdown list virtualization is a technique used to efficiently render extensive lists of items while minimizing the impact on performance. This method is particularly advantageous when dealing with large datasets because it ensures that only a fixed number of DOM (Document Object Model) elements are created. When scrolling through the list, existing DOM elements are reused to display relevant data instead of generating new elements for each item. This recycling process is managed internally.

During virtual scrolling, the data retrieved from the data source depends on the popup height and the calculation of the list item height. Enabling the [enableVirtualization](#) option in a dropdown list activates this virtualization technique.

When fetching data from the data source, the [actionBegin](#) event is triggered before data retrieval begins. Then, the [actionComplete](#) event is triggered once the data is successfully fetched.

Furthermore, Incremental Search is supported with virtualization in the DropDownList component. When a key is typed, the focus is moved to the respective element, and the value is updated in the component in the open popup state. In the closed popup state, the respective value is updated in the component based on the typed key. The Incremental Search functionality is well-suited for scenarios involving remote data binding.

When the enableVirtualization property is enabled, the `skip` and `take` properties provided by the user within the Query class at the initial state or during the `actionBegin` or `actionComplete` events will not be considered, since it is internally managed and calculated based on certain dimensions with respect to the popup height.

Binding local data

The DropDownList can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property. When using virtual scrolling, the list updates based on the scroll offset value, triggering a request to fetch more data from the server. As the data is being fetched, the `actionBegin` event occurs before the data retrieval starts. Once the data retrieval is successful, the `actionComplete` event is triggered, indicating that the data fetch process is complete.

In the following example, `id` column and `text` column from complex data have been mapped to the `value` field and `text` field, respectively.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
```

```

        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    fields = { text: 'text', value: 'id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={false}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </DropDownListComponent>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        this.records = Array.from({ length: 150 }, (_, i) => ({
            id: 'id' + (i + 1),
            text: `Item ${i + 1}`,
        }));
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={false}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </DropDownListComponent>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Binding remote data

The DropDownList supports retrieval of data from remote data services with the help of **DataManager** component. When using remote data, it initially fetches all the data from the server, triggering the

`actionBegin` and `actionComplete` events, and then stores the data locally. During virtual scrolling, additional data is retrieved from the locally stored data, triggering the `actionBegin` and `actionComplete` events at that time as well.

The following sample displays the OrderId from the `Orders` Data Service.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import { DataManager, WebApiAdaptor , Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  customerData = new DataManager({
    url: 'https://services.syncfusion.com/react/production/api/Orders',
    adaptor: new WebApiAdaptor,
    crossDomain: true
  });
  customerField = { text: 'OrderID', value: 'OrderID' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="datas" dataSource={this.customerData}
placeholder="OrderID" enableVirtualization={true} allowFiltering={true}
fields={this.customerField} popupHeight="200px" >
        <Inject services={[VirtualScroll]} />
      </DropDownListComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import { DataManager, WebApiAdaptor , Query } from '@syncfusion/ej2-data';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  // maps the appropriate column to fields property
  private customerField: object = { text: 'OrderID', value: 'OrderID' };

  private customerData: DataManager = new DataManager({
    url: 'https://services.syncfusion.com/react/production/api/Orders',
    adaptor: new WebApiAdaptor,
    crossDomain: true
  });
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="datas" dataSource={this.customerData}
placeholder="OrderID" enableVirtualization={true} allowFiltering={true}
fields={this.customerField} popupHeight="200px" >
```

```

        <Inject services={[VirtualScroll]} />
      </DropDownListComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Grouping

The DropDownList component supports grouping with Virtualization. It allows you to organize elements into groups based on different categories. Each item in the list can be classified using the [groupBy](#) field in the data table. After grouping, virtualization works similarly to local data binding, providing a seamless user experience. When the data source is bound to remote data, an initial request is made to retrieve all data for the purpose of grouping. Subsequently, the grouped data works in the same way as local data binding on virtualization.

The following sample shows the example for Grouping with Virtualization.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  records = [];
  constructor(props) {
    super(props);
    for (let i = 1; i <= 150; i++) {
      let item = {};
      item.id = 'id' + i;
      item.text = `Item ${i}`;
      // Generate a random number between 1 and 4 to determine the
group
      const randomGroup = Math.floor(Math.random() * 4) + 1;
      switch (randomGroup) {
        case 1:
          item.group = 'Group A';
          break;
        case 2:
          item.group = 'Group B';
          break;
        case 3:
          item.group = 'Group C';
          break;
        case 4:
          item.group = 'Group D';
          break;
        default:
          break;
      }
      this.records.push(item);
    }
  }
}

```

```

    fields = { groupBy: 'group', text: 'text', value: 'id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
                <Inject services={[VirtualScroll]} />
            </DropDownListComponent>);
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent, Inject, VirtualScroll } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = {groupBy: 'group', text: 'text', value: 'id' };
    // define the array of string
    private records: { [key: string]: Object }[] = [];

    // define the array of string
    constructor(props) {
        super(props);
        for (let i = 1; i <= 150; i++) {
            const item: { id: string, text: string, group: string } = {
                id: 'id' + i,
                text: `Item ${i}`,
                group: ''
            };
            // Generate a random number between 1 and 4 to determine the group
            const randomGroup = Math.floor(Math.random() * 4) + 1;
            switch (randomGroup) {
                case 1:
                    item.group = 'Group A';
                    break;
                case 2:
                    item.group = 'Group B';
                    break;
                case 3:
                    item.group = 'Group C';
                    break;
                case 4:
                    item.group = 'Group D';
                    break;
                default:
                    break;
            }
            this.records.push(item);
        }
    }
    public render() {

```

```

    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll]} />
      </DropDownListComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Filtering with Virtualization

The DropDownList component supports Filtering with Virtualization. The DropDownList includes a built-in feature that enables data filtering when the [allowFiltering](#) option is enabled. In the context of Virtual Scrolling, the filtering process operates in response to the typed characters. Specifically, the DropDownList sends a request to the server, utilizing the full data source, to achieve filtering. Before initiating the request, an action event is triggered. Upon successful retrieval of data from the server, an action complete event is triggered. The initial data is loaded when the popup is opened. Whether the filter list has a selection or not, the popup closes.

The following sample shows the example for Filtering with Virtualization.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  fields = { text: 'text', value: 'id' };
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll]} />
      </DropDownListComponent>);
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX


```

import { DropDownListComponent, Inject, VirtualScroll } from
 '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
  // maps the appropriate column to fields property
  private fields: object = { text: 'text', value: 'id' };
  // define the array of string
  private records: { [key: string]: Object }[] = [];

  // define the array of string
  constructor(props) {
    super(props);
    this.records = Array.from({ length: 150 }, (_, i) => ({
      id: 'id' + (i + 1),
      text: `Item ${i + 1}`,
    }));
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="datas" dataSource={this.records}
placeholder="e.g. Item 1" enableVirtualization={true} allowFiltering={true}
fields={this.fields} popupHeight="200px" >
        <Inject services={[VirtualScroll]}>/>
      </DropDownListComponent>
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Grouping in React Drop down list component

The DropDownList supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content

is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using [groupBy](#) field.

[Class-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the data with category
  vegetableData = [
    { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
    { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
    { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
  ],
  { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
  { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },

```

```

        { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
        { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
        { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
        { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
        { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
        { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    fields = { groupBy: 'Category', text: 'Vegetable', value: 'Id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight='200px'
fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
vegetable"/>);
        }
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the data with category
    private vegetableData: { [key: string]: Object }[] = [
        { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
        { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
        { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
        { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
        { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
        { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
        { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
        { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
        { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
        { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
        { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    private fields: object = { groupBy: 'Category', text: 'Vegetable', value: 'Id' };
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight='200px'
fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
vegetable" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    const vegetableData = [
        { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
        { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
        { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
    ],
    { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
    { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
    { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
    { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
    { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
    { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
    { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
    { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    const fields = { groupBy: 'Category', text: 'Vegetable', value: 'Id' };
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" popupHeight='200px'
        fields={fields} dataSource={vegetableData} placeholder="Select a
        vegetable"/>);
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the data with category
    const vegetableData: { [key: string]: Object }[] = [
        { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
        { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
        { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
    ],
    { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
    { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
    { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
    { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
    { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
    { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
    { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
    { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    const fields: object = { groupBy: 'Category', text: 'Vegetable', value:
    'Id' };
    return (

```

```
// specifies the tag for render the DropDownList component
<DropDownListComponent id="ddlelement" popupHeight='200px'
fields={fields} dataSource={vegetableData} placeholder="Select a vegetable"
/>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Customization

The grouping header is also provided with customization option. This allows custom designing using the [Link to the Video](#) property for both inline and fixed headers as referred here: [Group Template support to DropDownList](#).

To get started with grouping, grouping templates and adding icons in React DropDownList, you can check on this video:

See Also

- [Group Template support to DropDownList](#).
- [How to disable the fixed group header](#)

Filtering in React Drop down list component

The DropDownList has built-in support to filter data items when `allowFiltering` is enabled. The filter operation starts as soon as you start typing characters in the search box.

To display filtered items in the popup, filter the required data and return it to the DropDownList via `updateData` method by using the `filtering` event.

The following sample illustrates how to query the data source and pass the data to the DropDownList through the `updateData` method in `filtering` event.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // maps the appropriate column to fields property
  fields = { text: "Country", value: "Index" };
  // define the filtering data
  searchData = [
    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"California" },
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
"Georgia" }
  ];
  // filtering event handler to filter a Country
  onFiltering(args) {
    let query = new Query();
    // frame the query based on search string with filter type.
```

```

        query = (args.text !== "") ? query.where("Country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight="250px"
fields={this.fields} filtering={this.onFiltering =
this.onFiltering.bind(this)} allowFiltering={true}
dataSource={this.searchData} placeholder="Select a country"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // maps the appropriate column to fields property
    private fields: object = { text: "Country", value: "Index" };
    // define the filtering data
    private searchData: { [key: string]: Object }[] = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"California" },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia"
    }
    ];
    // filtering event handler to filter a Country
    public onFiltering(args: FilteringEventArgs) {
        let query = new Query();
        // frame the query based on search string with filter type.
        query = (args.text !== "") ? query.where("Country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight="250px"
fields={this.fields} filtering={this.onFiltering =
this.onFiltering.bind(this)} allowFiltering={true}
dataSource={this.searchData} placeholder="Select a country" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // maps the appropriate column to fields property
    const fields = { text: "Country", value: "Index" };
    // define the filtering data
    const searchData = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"California" },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
"Georgia" }
    ];
    // filtering event handler to filter a Country
    function onFiltering(args) {
        let query = new Query();
        // frame the query based on search string with filter type.
        query = (args.text !== "") ? query.where("Country", "startswith",
args.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        args.updateData(this.searchData, query);
    }
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" popupHeight="250px"
fields={fields} filtering={onFiltering = onFiltering.bind(this)}
allowFiltering={true} dataSource={searchData} placeholder="Select a
country"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // maps the appropriate column to fields property
    const fields: object = { text: "Country", value: "Index" };
    // define the filtering data
    const searchData: { [key: string]: Object }[] = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"California" },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia"
}
    ];
    // filtering event handler to filter a Country
    function onFiltering(args: FilteringEventArgs) {
```

```

    let query = new Query();
    // frame the query based on search string with filter type.
    query = (args.text !== "") ? query.where("Country", "startswith",
args.text, true) : query;
    // pass the filter data source, filter query to updateData method.
    args.updateData(this.searchData, query);
  }
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" popupHeight="250px"
fields={fields} filtering={onFiltering = onFiltering.bind(this)}
allowFiltering={true} dataSource={searchData} placeholder="Select a country"
/>
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the DropDownList. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

[Class-component]

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  searchData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  fields = { text: 'ContactName', value: 'CustomerID' };
  // bind the Query instance to query property
  query = new Query().select(['ContactName', 'CustomerID']).take(7);
  // sort the resulted items
  sortOrder = 'Ascending';
  // filtering event handler to filter a customer
  onFiltering(e) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    }
    else {
      // restrict the remote request until search key contains 3
characters.

```

```

        if (e.text.length < 3) {
            return;
        }
        let query = new Query().select(['ContactName', 'CustomerID']);
        query = (e.text !== '') ? query.where('ContactName',
'startswith', e.text, true) : query;
        e.updateData(this.searchData, query);
    }
}
render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={this.onFiltering =
this.onFiltering.bind(this)} sortOrder={this.sortOrder} query={this.query}
dataSource={this.searchData} fields={this.fields} placeholder="Select a
customer"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        } else {
            // restrict the remote request until search key contains 3 characters.
            if (e.text.length < 3) { return; }
            let query: Query = new Query().select(['ContactName', 'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
}

```



```

    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
            popupHeight="250px" filtering={this.onFiltering} =
            this.onFiltering.bind(this)} sortOrder={this.sortOrder} query={this.query}
            dataSource={this.searchData} fields={this.fields} placeholder="Select a
            customer" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    const searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    const fields = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    const query = new Query().select(['ContactName', 'CustomerID']).take(7);
    // sort the resulted items
    const sortOrder = 'Ascending';
    // filtering event handler to filter a customer
    function onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        }
        else {
            // restrict the remote request until search key contains 3
            characters.
            if (e.text.length < 3) {
                return;
            }
            let query = new Query().select(['ContactName', 'CustomerID']);
            query = (e.text !== '') ? query.where('ContactName',
            'startswith', e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
    return (
        // specifies the tag for render the DropDownList component

```

```

    <DropDownListComponent id="ddlelement" allowFiltering={true}
    popupHeight="250px" filtering={onFiltering = onFiltering.bind(this)}
    sortOrder={sortOrder} query={query} dataSource={searchData} fields={fields}
    placeholder="Select a customer"/>;
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  const searchData: DataManager = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
  });
  // maps the appropriate column to fields property
  const fields: object = { text: 'ContactName', value: 'CustomerID' };
  // bind the Query instance to query property
  const query: Query = new Query().select(['ContactName',
  'CustomerID']).take(7);
  // sort the resulted items
  const sortOrder: SortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e: FilteringEventArgs) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    } else {
      // restrict the remote request until search key contains 3 characters.
      if (e.text.length < 3) { return; }
      let query: Query = new Query().select(['ContactName', 'CustomerID']);
      query = (e.text !== '') ? query.where('ContactName', 'startswith',
      e.text, true) : query;
      e.updateData(this.searchData, query);
    }
  }
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" allowFiltering={true}
    popupHeight="250px" filtering={onFiltering = onFiltering.bind(this)}
    sortOrder={sortOrder} query={query} dataSource={searchData} fields={fields}
    placeholder="Select a customer" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

[Class-component]

INDEX.JSX

```
// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    searchData = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"Californiag" },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
"Georgia" }
    ];
    // maps the appropriate column to fields property
    fields = { value: "Country" };
    // sort the resulted items
    sortOrder = 'Ascending';
    // filtering event handler to filter a customer
    onFiltering(e) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        }
        else {
            let query = new Query();
            // change the type of filtering
            query = (e.text !== '') ? query.where('Country', 'endsWith',
e.text, true) : query;
            e.updateData(this.searchData, query);
        }
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={this.onFiltering =
this.onFiltering.bind(this)} sortOrder={this.sortOrder}
dataSource={this.searchData} fields={this.fields} placeholder="Select a
customer"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
```

```

import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: { [key: string]: Object }[] = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country: "Californiag"
    },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia" }
    ]
    // maps the appropriate column to fields property
    private fields: object = { value: "Country" };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        } else {
            let query: Query = new Query();
            // change the type of filtering
            query = (e.text !== '') ? query.where('Country', 'endsWith', e.text,
true) : query;
            e.updateData(this.searchData, query);
        }
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={this.onFiltering} =
this.onFiltering.bind(this)} sortOrder={this.sortOrder}
dataSource={this.searchData} fields={this.fields} placeholder="Select a
customer" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind the DataManager instance to dataSource property
    let searchData = [

```

```

    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"Californiag" },
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
"Georgia" }
  ];
  // maps the appropriate column to fields property
  let fields = { value: "Country" };
  // sort the resulted items
  let sortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(searchData);
    }
    else {
      let query = new Query();
      // change the type of filtering
      query = (e.text !== '') ? query.where('Country', 'endsWith',
e.text, true) : query;
      e.updateData(searchData, query);
    }
  }
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={onFiltering = onFiltering.bind(this)}
sortOrder={sortOrder} dataSource={searchData} fields={fields}
placeholder="Select a customer"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import DataManager related classes
import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // bind the DataManager instance to dataSource property
  let searchData: { [key: string]: Object }[] = [
    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country: "Californiag"
},
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia" }
  ]
  // maps the appropriate column to fields property
  let fields: object = { value: "Country" };
  // sort the resulted items
  let sortOrder: SortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e: FilteringEventArgs) {
    // load overall data when search key empty.
    if (e.text === '') {

```

```

    e.updateData(searchData);
  } else {
    let query: Query = new Query();
    // change the type of filtering
    query = (e.text !== '') ? query.where('Country', 'endsWith', e.text,
true) : query;
    e.updateData(searchData, query);
  }
}

return (
  // specifies the tag for render the DropDownList component
  <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={onFiltering} = onFiltering.bind(this) }
sortOrder={sortOrder} dataSource={searchData} fields={fields}
placeholder="Select a customer" />
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

The following example shows how to perform case-sensitive filter.

[Class-component]

INDEX.JSX

```

import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  searchData = [
    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"Californiag" },
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
"Georgia" }
  ];
  // maps the appropriate column to fields property
  fields = { value: "Country" };
  // sort the resulted items
  sortOrder = 'Ascending';
  // filtering event handler to filter a customer
  onFiltering(e) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    }
    else {
      let query = new Query();
      // change the type of filtering
      query = (e.text !== '') ? query.where('Country', 'contains',
e.text, false) : query;
    }
  }
}

```

```

        e.updateData(this.searchData, query);
    }
}
render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" popupHeight="245px"
allowFiltering={true} filtering={this.onFiltering =
this.onFiltering.bind(this)} sortOrder={this.sortOrder}
dataSource={this.searchData} fields={this.fields} placeholder="Select a
customer"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: { [key: string]: Object }[] = [
        { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
"Californiag" },
        { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia"
    }
    ]
    // maps the appropriate column to fields property
    private fields: object = { value: "Country" };
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // load overall data when search key empty.
        if (e.text === '') {
            e.updateData(this.searchData);
        } else {
            let query: Query = new Query();
            // change the type of filtering
            query = (e.text !== '') ? query.where('Country', 'contains', e.text,
false) : query;
            e.updateData(this.searchData, query);
        }
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight="245px"
allowFiltering={true} filtering={this.onFiltering =
this.onFiltering.bind(this)} sortOrder={this.sortOrder}
dataSource={this.searchData} fields={this.fields} placeholder="Select a
customer" />

```

```

    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

function App() {
  // bind the DataManager instance to dataSource property
  const searchData = [
    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
    "Californiag" },
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country:
    "Georgia" }
  ];
  // maps the appropriate column to fields property
  const fields = { value: "Country" };
  // sort the resulted items
  const sortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    }
    else {
      let query = new Query();
      // change the type of filtering
      query = (e.text !== '') ? query.where('Country', 'contains',
e.text, false) : query;
      e.updateData(this.searchData, query);
    }
  }
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" popupHeight="245px"
allowFiltering={true} filtering={onFiltering = onFiltering.bind(this)}
sortOrder={sortOrder} dataSource={searchData} fields={fields}
placeholder="Select a customer"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-
react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```



```
function App(){
  // bind the DataManager instance to dataSource property
  const searchData: { [key: string]: Object }[] = [
    { Index: "s1", Country: "Alaska" }, { Index: "s2", Country:
    "Californiag" },
    { Index: "s3", Country: "Florida" }, { Index: "s4", Country: "Georgia"
  }
  ]
  // maps the appropriate column to fields property
  const fields: object = { value: "Country" };
  // sort the resulted items
  const sortOrder: SortOrder = 'Ascending';
  // filtering event handler to filter a customer
  function onFiltering(e: FilteringEventArgs) {
    // load overall data when search key empty.
    if (e.text === '') {
      e.updateData(this.searchData);
    } else {
      let query: Query = new Query();
      // change the type of filtering
      query = (e.text !== '') ? query.where('Country', 'contains', e.text,
      false) : query;
      e.updData(query);
    }
  }
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" popupHeight="245px"
    allowFiltering={true} filtering={onFiltering = onFiltering.bind(this)}
    sortOrder={sortOrder} dataSource={searchData} fields={fields}
    placeholder="Select a customer" />
  );
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Diacritics Filtering

DropDownList supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample,data with diacritics are bound as dataSource for DropDownList.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  diacriticsData = [
    'Aeróbics',
    'Aeróbics en Agua',
    'Aerografía',
    'Aerodelaje',
    'Águilas',
    'Ajedrez',
  ]
}
```

```

        'Ala Delta',
        'Álbumes de Música',
        'Alusivos',
        'Análisis de Escritura a Mano'
    ];
    render() {
        return (<DropDownListComponent id="diacritics" ignoreAccent={true}
        dataSource={this.diacriticsData} allowFiltering={true} placeholder="Select a
        value" filterBarPlaceholder="e.g: aero"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    private diacriticsData: string[] = [
        'Aeróbics',
        'Aeróbics en Agua',
        'Aerografía',
        'Aerodelaje',
        'Águilas',
        'Ajedrez',
        'Ala Delta',
        'Álbumes de Música',
        'Alusivos',
        'Análisis de Escritura a Mano'];
    public render() {
        return (
            <DropDownListComponent id="diacritics" ignoreAccent={true}
            dataSource={this.diacriticsData} allowFiltering={true} placeholder="Select a
            value" filterBarPlaceholder="e.g: aero" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const diacriticsData = [
        'Aeróbics',
        'Aeróbics en Agua',
        'Aerografía',
        'Aerodelaje',
        'Águilas',
        'Ajedrez',
        'Ala Delta',

```

```

        'Álbumes de Música',
        'Alusivos',
        'Análisis de Escritura a Mano'
    ];
    return (<DropDownListComponent id="diacritics" ignoreAccent={true}
dataSource={diacriticsData} allowFiltering={true} placeholder="Select a
value" filterBarPlaceholder="e.g: aero"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const diacriticsData: string[] = [
        'Aeróbics',
        'Aeróbics en Agua',
        'Aerografía',
        'Aeromodelaje',
        'Águilas',
        'Ajedrez',
        'Ala Delta',
        'Álbumes de Música',
        'Alusivos',
        'Análisis de Escritura a Mano'];
    return (
        <DropDownListComponent id="diacritics" ignoreAccent={true}
dataSource={diacriticsData} allowFiltering={true} placeholder="Select a
value" filterBarPlaceholder="e.g: aero" />
    );
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

You can also explore our [React DropDownList Filtering example](#) that shows how to render the DropDownList Filter in React.

See Also

- [How to limit the search result while filtering](#)
- [How to highlight the matched characters in filtering](#)
- [How to modify the result data using remote data source](#)

Localization in React Drop down list component

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the DropDownList.

| Locale key | en-US (default) |

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

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the DropDownList and no data is loaded. Hence, the **noRecordsTemplate** property displays its text in French culture initially, and if the sample is run offline, the **actionFailureTemplate** property displays its text appropriately.

[Class-component]

INDEX.JSX

```
// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind remotedata to showcase actionFailureTemplate in offline.
    customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property.
    query = new Query().select(['ContactName', 'CustomerID']).take(0);
    // set locale culture to DropDownList
    componentWillMount() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" fields={this.fields}
            locale="fr-BE" query={this.query} dataSource={this.customerData}
            placeholder="Sélectionnez un client"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
```

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind remotedata to showcase actionFailureTemplate in offline.
    private customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property.
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(0);
    // set locale culture to DropDownList
    public componentWillMount() {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" fields={this.fields} locale="fr-
BE" query={this.query} dataSource={this.customerData}
placeholder="Sélectionnez un client" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

[Functional-component]

INDEX.JSX

```
// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    const customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
```

```

    });
    const fields= { text: 'ContactName', value: 'CustomerID' };
    const query= new Query().select(['ContactName', 'CustomerID']).take(0);
    // set locale culture to DropDownList
    React.useEffect(() => {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }, []);
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" fields={fields} locale="fr-BE"
        query={query} dataSource={customerData} placeholder="Sélectionnez un
        client"/>);
    }
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

// import L10n class for load function
import { L10n } from '@syncfusion/ej2-base';
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // bind remotedata to showcase actionFailureTemplate in offline.
    const customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    const fields: object = { text: 'ContactName', value: 'CustomerID' };
    // take 0 item to showcase noRecordsTemplate property.
    const query: Query = new Query().select(['ContactName',
    'CustomerID']).take(0);
    // set locale culture to DropDownList
    React.useEffect(() => {
        L10n.load({
            'fr-BE': {
                'dropdowns': {
                    'actionFailureTemplate': "Modèle d'échec d'action",
                    'noRecordsTemplate': "Aucun enregistrement trouvé"
                }
            }
        });
    }, []);
    return (
        // specifies the tag for render the DropDownList component

```

```
<DropDownListComponent id="ddlelement" fields={fields} locale="fr-BE"
query={query} dataSource={customerData} placeholder="Sélectionnez un client"
/>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

Style in React Drop down list 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 wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
.e-ddl.e-input-group.e-control-wrapper .e-input {
font-size: 20px;
font-family: emoji;
color: #ab3243;
background: #32a5ab;
}
`
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
.e-ddl.e-input-group .e-input-group-icon,.e-ddl.e-input-group.e-control-wrapper .e-input-group-
icon:hover {
color: #bb233d;
font-size: 13px;
}
`
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css
.e-ddl.e-input-group.e-control-wrapper.e-input-focus::before, .e-ddl.e-input-group.e-control-wrapper.e-
input-focus::after {
```

```
background: #c000ff;
}
```

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css

.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {

border-color: #b1bd15;

box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;

}
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css

.e-input-group.e-control-wrapper .e-input[disabled] {

-webkit-text-fill-color: #0d9133;

}
```

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css

.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::after {

background-color: #2319b8;

}

.e-ddl.e-lib.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top {

color: #2319b8;

}
```


Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css
.e-ddl.e-input-group input.e-input::placeholder {
color: red;
}
`
```

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
`
```

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px;
}
`
```

Adding mandatory asterisk to placeholder and float label

You can add a mandatory asterisk(*) to placeholder and float label using `.e-input-group.e-control-wrapper.e-float-input .e-float-text::after` class.

[Class-component]

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the array of data
  sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
```

```

render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
placeholder="Select a game" floatLabelType="auto"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" dataSource={this.sportsData}
placeholder="Select a game" floatLabelType= "auto" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

[Functional-component]**INDEX.JSX**

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" dataSource={sportsData}
placeholder="Select a game" floatLabelType="auto"/>);
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // define the array of data
    const sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
'Tennis'];

```

```

    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" dataSource={sportsData}
        placeholder="Select a game" floatLabelType= "auto" />
    );
  }
  ReactDOM.render(<App />, document.getElementById('sample'));

```

Accessibility in React Drop down list component

The DropDownList component has been designed, keeping in mind the **WAI-ARIA** specifications, and applies the WAI-ARIA roles, states, and properties along with **keyboard support**. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who

use assistive technologies (AT) or those who completely rely on keyboard navigation.

The DropDownList component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the DropDownList component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All
features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

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

| Properties | Functionalities |

```
| --- | --- |
```

```
| aria-haspopup | Indicates whether the DropDownList input element has a popup list or not. |
```

```
| aria-expanded | Indicates whether the popup list has expanded or not. |
```

```
| aria-selected | Indicates the selected option. |
```

```
| aria-readonly | Indicates the readonly state of the DropDownList element. |
```

```
| aria-disabled | Indicates whether the DropDownList component is in a disabled state or not. |
```

```
| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child
element. |
```

```
| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |
```

Keyboard interaction

You can use the following key shortcuts to access the DropDownList without interruptions.

| Keyboard shortcuts | Actions |

```
| --- | --- |
```

```
| Arrow Down | Selects the first item in the DropDownList when no item selected. Otherwise,
selects the item next to the currently selected item. |
```

```
| Arrow Up | Selects the item previous to the currently selected one. |
```

```
| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |
```

```
| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |
```

```
| Enter | Selects the focused item, and when it is in an open state the popup list closes.
Otherwise, toggles the popup list. |
```

```
| Tab | Focuses on the next TabIndex element on the page when the popup is closed.
Otherwise, closes the popup list and remains the focus of the component. |
```

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Selects the first item. |

| End | Selects the last item. |

In the below sample, alt+t keys are used to focus the DropDownList component.

[Class-component]

INDEX.JSX

```
{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // defined the array of data
  gameList = [
    { Id: 'Game1', Game: 'Badminton' },
    { Id: 'Game2', Game: 'Basketball' },
    { Id: 'Game3', Game: 'Cricket' },
    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' },
    { Id: 'Game6', Game: 'Hockey' },
    { Id: 'Game7', Game: 'Rugby' },
    { Id: 'Game8', Game: 'Snooker' },
    { Id: 'Game9', Game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  // instance of DropDownList component
  dropDownListObj;
  componentDidMount() {
    const proxy = this;
    document.onkeyup = (e) => {
      if (e.altKey && e.keyCode === 84 /* t */) {
        // press alt+t to focus the control.
        proxy.dropDownListObj.inputElement.focus();
      }
    };
  }
  render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" ref={(scope) => {
        this.dropDownListObj = scope; }} popupHeight='200px' fields={this.fields}
        dataSource={this.gameList} placeholder="Select a game"/>);
  }
}
```

```
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // defined the array of data
    private gameList: { [key: string]: Object }[] = [
        { Id: 'Game1', Game: 'Badminton' },
        { Id: 'Game2', Game: 'Basketball' },
        { Id: 'Game3', Game: 'Cricket' },
        { Id: 'Game4', Game: 'Football' },
        { Id: 'Game5', Game: 'Golf' },
        { Id: 'Game6', Game: 'Hockey' },
        { Id: 'Game7', Game: 'Rugby' },
        { Id: 'Game8', Game: 'Snooker' },
        { Id: 'Game9', Game: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'Game', value: 'Id' };
    // instance of DropDownList component
    private dropDownListObj: any;
    public componentDidMount() {
        const proxy = this;
        document.onkeyup = (e) => {
            if (e.altKey && e.keyCode === 84 /* t */) {
                // press alt+t to focus the control.
                proxy.dropDownListObj.inputElement.focus();
            }
        };
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddelement" ref={(scope) => {
                this.dropDownListObj = scope; }} popupHeight='200px' fields={this.fields}
                dataSource={this.gameList} placeholder="Select a game" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

[Functional-component]

INDEX.JSX

```
{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
```

```

// defined the array of data
const gameList = [
  { Id: 'Game1', Game: 'Badminton' },
  { Id: 'Game2', Game: 'Basketball' },
  { Id: 'Game3', Game: 'Cricket' },
  { Id: 'Game4', Game: 'Football' },
  { Id: 'Game5', Game: 'Golf' },
  { Id: 'Game6', Game: 'Hockey' },
  { Id: 'Game7', Game: 'Rugby' },
  { Id: 'Game8', Game: 'Snooker' },
  { Id: 'Game9', Game: 'Tennis' }
];
// maps the appropriate column to fields property
const fields = { text: 'Game', value: 'Id' };
// instance of DropDownList component
let dropDownListObj;
React.useEffect(() => {
  document.onkeyup = (e) => {
    if (e.altKey && e.keyCode === 84 /* t */) {
      // press alt+t to focus the control.
      dropDownListObj.inputElement.focus();
    }
  };
}, []);
return (
  // specifies the tag for render the DropDownList component
  <DropDownListComponent id="ddlelement" ref={(scope) => { dropDownListObj
= scope; }} popupHeight='200px' fields={fields} dataSource={gameList}
placeholder="Select a game"/>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defined the array of data
  const gameList: { [key: string]: Object }[] = [
    { Id: 'Game1', Game: 'Badminton' },
    { Id: 'Game2', Game: 'Basketball' },
    { Id: 'Game3', Game: 'Cricket' },
    { Id: 'Game4', Game: 'Football' },
    { Id: 'Game5', Game: 'Golf' },
    { Id: 'Game6', Game: 'Hockey' },
    { Id: 'Game7', Game: 'Rugby' },
    { Id: 'Game8', Game: 'Snooker' },
    { Id: 'Game9', Game: 'Tennis' }
  ];
  // maps the appropriate column to fields property
  const fields: object = { text: 'Game', value: 'Id' };
  // instance of DropDownList component
  let dropDownListObj: any;

```

```

    React.useEffect(() => {
      document.onkeyup = (e) => {
        if (e.altKey && e.keyCode === 84 /* t */) {
          // press alt+t to focus the control.
          dropDownListObj.inputElement.focus();
        }
      };
    }, []);
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" ref={(scope) => {
        dropDownListObj = scope; }} popupHeight='200px' fields={fields}
        dataSource={gameList} placeholder="Select a game" />
    );
  }
  ReactDOM.render(<App />, document.getElementById('sample'));
  {% endraw %}

```

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)

How To

Add item in React Drop down list component

You can add item in between based on item [index](#). If you add new item without item index, item will be added as last item in list.

The following example demonstrate how to add item in between in DropDownList.

INDEX.JSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
  ];
  dropDownListObject;
  // maps the appropriate column to fields property
  fields = { text: 'Game', value: 'Id' };
  onclickfirst() {

```



```

        this.dropDownListObject.addItem({ Id: 'game0', Game: 'Hockey' }, 0);
    }
    onclick() {
        this.dropDownListObject.addItem({ Id: 'game4', Game: 'Golf' }, 2);
    }
    onclicklast() {
        this.dropDownListObject.addItem({ Id: 'game5', Game: 'Cricket' });
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent id="ddlelement" ref={(scope) => {
                    this.dropDownListObject = scope; }} dataSource={this.sportsData}
                    fields={this.fields} placeholder="Select a game"/>
                <div id="btn1Div">
                    <button id='btn' className="e-control e-btn"
                        onClick={this.onclickfirst = this.onclickfirst.bind(this)}>
                        add item (Hockey) in first</button>
                </div>
                <div id="btn2Div">
                    <button id='btn2' className="e-control e-btn"
                        onClick={this.onclick = this.onclick.bind(this)}> add item (Golf) in
                    between</button>
                </div>
                <div id="btn3Div">
                    <button id='btn3' className="e-control e-btn"
                        onClick={this.onclicklast = this.onclicklast.bind(this)}>
                        add item (Cricket) in last</button>
                </div>
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the JSON of data
    private sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },
        { Id: 'game3', Game: 'Tennis' }
    ];
    private dropDownListObject: any;
    // maps the appropriate column to fields property
    private fields: object = { text: 'Game', value: 'Id' };
    public onclickfirst() {
        this.dropDownListObject.addItem({ Id: 'game0', Game: 'Hockey' }, 0);
    }
    public onclick() {

```

```

    this.dropDownListObject.addItem({Id: 'game4', Game: 'Golf'}, 2);
  }
  public onclicklast() {
    this.dropDownListObject.addItem({Id: 'game5', Game: 'Cricket'});
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <div>
        <DropDownListComponent id="ddlelement" ref={ (scope) => {
          this.dropDownListObject = scope; }}dataSource={this.sportsData}
          fields={this.fields} placeholder="Select a game" />
          <div id="btn1Div">
            <button id='btn' className="e-control e-btn"
              onClick={this.onclickfirst = this.onclickfirst.bind(this)}>
              add item (Hockey) in first</button>
            </div>
            <div id="btn2Div">
              <button id='btn2' className="e-control e-btn" onClick={
                this.onclick = this.onclick.bind(this)}> add item (Golf) in between</button>
            </div>
            <div id="btn3Div">
              <button id='btn3' className="e-control e-btn"
                onClick={this.onclicklast = this.onclicklast.bind(this)}>
                add item (Cricket) in last</button>
            </div>
          </div>
        );
      );
    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));
  {% endraw %}

```

Cascading in React Drop down list component

The cascading DropDownList is a series of DropDownList, where the value of one DropDownList depends upon another's value. This can be configured by using the [change](#) event of the parent DropDownList. Within that change event handler, data has to be loaded to the child DropDownList based on the selected value of the parent DropDownList.

The following example, shows the cascade behavior of country, state, and city DropDownList. Here, the `dataBind` method is used to reflect the property changes immediately to the DropDownList.

INDEX.JSX

```

{% raw %}
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // country DropDownList instance
  countryObj;
  // state DropDownList instance
  stateObj;
  // city DropDownList instance
  cityObj;

```

```

// define the country DropDownList data
countryData = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
];
// define the state DropDownList data
stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
// define the city DropDownList data
cityData = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
// maps the country column to fields property
countryField = { value: 'CountryId', text: 'CountryName' };
// maps the state column to fields property
stateField = { value: 'StateId', text: 'StateName' };
// maps the city column to fields property
cityField = { text: 'CityName', value: 'CityId' };
onCountryChange() {
    // query the data source based on country DropDownList selected value
    this.stateObj.query = new Query().where('CountryId', 'equal',
this.countryObj.value);
    // enable the state DropDownList
    this.stateObj.enabled = true;
    // clear the existing selection.
    this.stateObj.text = null;
    // bind the property changes to state DropDownList
    this.stateObj.dataBind();
    // clear the existing selection in city DropDownList
    this.cityObj.text = null;
    // disable the city DropDownList
    this.cityObj.enabled = false;
    // bind the property change to City DropDownList
    this.cityObj.dataBind();
}
onStateChange() {
    // query the data source based on state DropDownList selected value
    this.cityObj.query = new Query().where('StateId', 'equal',
this.stateObj.value);
    // enable the city DropDownList
    this.cityObj.enabled = true;
    // clear the existing selection
    this.cityObj.text = null;
    // bind the property change to city DropDownList
    this.cityObj.dataBind();
}
render() {
    return (<div>

```

```

        /* specifies the tag for render the country DropDownList
component */
        <DropDownListComponent id="country-ddl" ref={(scope) => {
this.countryObj = scope; }} fields={this.countryField}
dataSource={this.countryData} placeholder='Select a country'
change={this.onCountryChange = this.onCountryChange.bind(this)} />
        <br />
        /* specifies the tag for render the state DropDownList
component */
        <DropDownListComponent id="state-ddl" ref={(scope) => {
this.stateObj = scope; }} enabled={false} fields={this.stateField}
dataSource={this.stateData} placeholder='Select a state'
change={this.onStateChange = this.onStateChange.bind(this)} />
        <br />
        /* specifies the tag for render the city DropDownList
component */
        <DropDownListComponent id="city-ddl" ref={(scope) => {
this.cityObj = scope; }} enabled={false} fields={this.cityField}
dataSource={this.cityData} placeholder='Select a city' />
    </div>;
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // country DropDownList instance
    private countryObj: any;
    // state DropDownList instance
    private stateObj: any;
    // city DropDownList instance
    private cityObj: any;
    // define the country DropDownList data
    private countryData: { [key: string]: Object }[] = [
        { CountryName: 'Australia', CountryId: '2' },
        { CountryName: 'United States', CountryId: '1' }
    ];
    // define the state DropDownList data
    private stateData: { [key: string]: Object }[] = [
        { StateName: 'New York', CountryId: '1', StateId: '101' },
        { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
        { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
    ];
    // define the city DropDownList data
    private cityData: { [key: string]: Object }[] = [
        { CityName: 'Albany', StateId: '101', CityId: 201 },
        { CityName: 'Beacon ', StateId: '101', CityId: 202 },
        { CityName: 'Emporia', StateId: '102', CityId: 206 },
        { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    ]
}

```

```

        { CityName: 'Hobart', StateId: '105', CityId: 213 },
        { CityName: 'Launceston ', StateId: '105', CityId: 214 }
    ];
    // maps the country column to fields property
    private countryField: object = { value: 'CountryId', text: 'CountryName'
};
    // maps the state column to fields property
    private stateField: object = { value: 'StateId', text: 'StateName' };
    // maps the city column to fields property
    private cityField: object = { text: 'CityName', value: 'CityId' };
    public onCountryChange() {
        // query the data source based on country DropDownList selected value
        this.stateObj.query = new Query().where('CountryId', 'equal',
this.countryObj.value);
        // enable the state DropDownList
        this.stateObj.enabled = true;
        // clear the existing selection.
        this.stateObj.text = null;
        // bind the property changes to state DropDownList
        this.stateObj.dataBind();
        // clear the existing selection in city DropDownList
        this.cityObj.text = null;
        // disable the city DropDownList
        this.cityObj.enabled = false;
        // bind the property change to City DropDownList
        this.cityObj.dataBind();
    }
    public onStateChange() {
        // query the data source based on state DropDownList selected value
        this.cityObj.query = new Query().where('StateId', 'equal',
this.stateObj.value);
        // enable the city DropDownList
        this.cityObj.enabled = true;
        // clear the existing selection
        this.cityObj.text = null;
        // bind the property change to city DropDownList
        this.cityObj.dataBind();
    }
    public render() {
        return (
            <div>
                /* specifies the tag for render the country DropDownList
component */
                <DropDownListComponent id="country-ddl" ref={(scope) => {
this.countryObj = scope; }} fields={this.countryField}
dataSource={this.countryData} placeholder='Select a country'
change={this.onCountryChange = this.onCountryChange.bind(this)} />
                <br />
                /* specifies the tag for render the state DropDownList
component */
                <DropDownListComponent id="state-ddl" ref={(scope) => {
this.stateObj = scope; }} enabled={false} fields={this.stateField}
dataSource={this.stateData} placeholder='Select a state'
change={this.onStateChange = this.onStateChange.bind(this)} />
                <br />
                /* specifies the tag for render the city DropDownList
component */

```

```

        <DropDownListComponent id="city-ddl" ref={(scope) => {
this.cityObj = scope; }} enabled={false} fields={this.cityField}
dataSource={this.cityData} placeholder='Select a city' />
        </div>
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

Clear item in React Drop down list component

You can clear the selected item in the below two different ways.

By clicking on the **clear icon** which is shown in DropDownList element, you can clear the selected item in DropDownList through **interaction**. By using [showClearButton](#) property, you can enable the clear icon in DropDownList element.

Through **programmatic** you can set **null** value to anyone of the index, text or value property to clear the selected item in DropDownList.

The following example demonstrate about how to clear the selected item in DropDownList.

INDEX.JSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    dropDownListObject;
    sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    onclick() {
        this.dropDownListObject.value = null;
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent id="ddlelement" ref={(scope) => {
this.dropDownListObject = scope; }} dataSource={this.sportsData}
placeholder="Select a game"/>
                <button id='btn' className="e-control e-btn" onClick={this.onclick =
this.onclick.bind(this)}> Set null to value property</button>
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {

```

```

private dropDownListObject: any;
private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
public onclick() {
    this.dropDownListObject.value = null;
}
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <div>
            <DropDownListComponent id="ddlelement" ref={(scope) => {
                this.dropDownListObject = scope; }} dataSource={this.sportsData}
                placeholder="Select a game" />
            <button id='btn' className="e-control e-btn" onClick={this.onclick =
                this.onclick.bind(this)}> Set null to value property</button>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

Close popup in React Drop down list component

By using the `hidePopup` method in `DropDownList`, you can close the popup on scroll when triggered the windows scroll event.

The following example demonstrate about how to close the popup on scroll.

INDEX.JSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    dropDownListObject;
    // define the data with category
    sportsData = [
        'Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'
    ];
    constructor(props) {
        super(props);
        window.addEventListener('scroll', this.handleScroll.bind(this));
    }
    // on scroll event
    handleScroll() {
        this.dropDownListObject.hidePopup();
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" ref={(scope) => {
                this.dropDownListObject = scope; }} dataSource={this.sportsData}
                placeholder="Select a game"/>);
    }
}
{% endraw %}

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

INDEX.TSX

```
{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
  private dropDownListObject: any;
  // define the data with category
  private sportsData: { [key: string]: Object }[] = [
    'Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
  constructor(props: any) {
    super(props);
    window.addEventListener('scroll', this.handleScroll.bind(this));
  }
  // on scroll event
  public handleScroll() {
    this.dropDownListObject.hidePopup();
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" ref={ (scope) => {
this.dropDownListObject = scope; }} dataSource={this.sportsData}
placeholder="Select a game" />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}
```

Group header in React Drop down list component

The following example demonstrate about how to disable the Fixed group header in DropDownList through CSS by using **visibility** attribute.

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the data with category
  vegetableData = [
    { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
    { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
    { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
  ],
  { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
  { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
  { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
  { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
  { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
}
```



```

        { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
        { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
        { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    fields = { groupBy: 'Category', text: 'Vegetable', value: 'Id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight='200px'
fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
vegetable"/>);
        }
    }
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the data with category
    private vegetableData: { [key: string]: Object }[] = [
        { Vegetable: 'Cabbage', Category: 'Leafy and Salad', Id: 'item1' },
        { Vegetable: 'Spinach', Category: 'Leafy and Salad', Id: 'item2' },
        { Vegetable: 'Wheat grass', Category: 'Leafy and Salad', Id: 'item3' },
        { Vegetable: 'Yarrow', Category: 'Leafy and Salad', Id: 'item4' },
        { Vegetable: 'Pumpkins', Category: 'Leafy and Salad', Id: 'item5' },
        { Vegetable: 'Chickpea', Category: 'Beans', Id: 'item6' },
        { Vegetable: 'Green bean', Category: 'Beans', Id: 'item7' },
        { Vegetable: 'Horse gram', Category: 'Beans', Id: 'item8' },
        { Vegetable: 'Garlic', Category: 'Bulb and Stem', Id: 'item9' },
        { Vegetable: 'Nopal', Category: 'Bulb and Stem', Id: 'item10' },
        { Vegetable: 'Onion', Category: 'Bulb and Stem', Id: 'item11' }
    ];
    // map the groupBy field with Category column
    private fields: object = { groupBy: 'Category', text: 'Vegetable', value:
    'Id' };
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" popupHeight='200px'
fields={this.fields} dataSource={this.vegetableData} placeholder="Select a
vegetable" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Highlight filtering in React Drop down list component

By using the `highlightSearch` method, you can highlight the matched character in DropDownList filtering.

The following example demonstrates about how to highlight the matched character in filtering.

INDEX.JSX

```
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent, highlightSearch } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url:
'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = {
        itemCreated: this.onItemCreated.bind(this),
        text: 'ContactName', value: 'CustomerID'
    };
    // bind the Query instance to query property
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    queryString;
    constructor(props) {
        super(props);
    }
    // highlight the item
    onItemCreated(e) {
        highlightSearch(e.item, this.queryString, true, 'StartsWith');
    }
    // filtering event handler to filter a customer
    onFiltering(e) {
        // take text for highlight the character in list items.
        this.queryString = e.text;
        let query = new Query().select(['ContactName', 'CustomerID']);
        // frame the query based on search string with filter type.
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        e.updateData(this.searchData, query);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={this.onFiltering} =
this.onFiltering.bind(this)} query={this.query} dataSource={this.searchData}
fields={this.fields} placeholder="Select a customer"/>);
        }
    }
    ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
```

```

import { DropDownListComponent, FilteringEventArgs, highlightSearch } from
'@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}>, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = {
        itemCreated: this.onItemCreated.bind(this),
        text: 'ContactName', value: 'CustomerID'
    };
    // bind the Query instance to query property
    private query: Query = new Query().select(['ContactName',
'CustomerID']).take(7);
    private queryString: string;
    constructor(props: any) {
        super(props);
    }
    // highlight the item
    public onItemCreated(e: any) {
        highlightSearch(e.item, this.queryString, true, 'StartsWith');
    }
    // filtering event handler to filter a customer
    public onFiltering(e: FilteringEventArgs) {
        // take text for highlight the character in list items.
        this.queryString = e.text;
        let query: Query = new Query().select(['ContactName', 'CustomerID']);
        // frame the query based on search string with filter type.
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
e.text, true) : query;
        // pass the filter data source, filter query to updateData method.
        e.updateData(this.searchData, query);
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
popupHeight="250px" filtering={this.onFiltering} =
this.onFiltering.bind(this)} query={this.query} dataSource={this.searchData}
fields={this.fields} placeholder="Select a customer" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Icons support in React Drop down list component

You can render **icons** to the list items by mapping the [iconCss](#) field. This [iconCss](#) field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with [iconCss](#) field.

INDEX.JSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sortFormatData = [
        { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
        { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
        { Class: 'filter', Type: 'Filter', Id: '3' },
        { Class: 'clear', Type: 'Clear', Id: '4' }
    ];
    // map the icon column to iconCSS field.
    fields = { text: 'Type', iconCss: 'Class', value: 'Id' };
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
            dataSource={this.sortFormatData} fields={this.fields} placeholder="Select a
            format"/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sortFormatData: { [key: string]: Object }[] = [
        { Class: 'asc-sort', Type: 'Sort A to Z', Id: '1' },
        { Class: 'dsc-sort', Type: 'Sort Z to A ', Id: '2' },
        { Class: 'filter', Type: 'Filter', Id: '3' },
        { Class: 'clear', Type: 'Clear', Id: '4' }
    ];
    // map the icon column to iconCSS field.
    private fields: object = { text: 'Type', iconCss: 'Class', value: 'Id' };
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement"
            dataSource={this.sortFormatData} fields={this.fields} placeholder="Select a
            format" />
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

Incremental search in React Drop down list component

DropDownList supports incremental search, by default. You can search the list item by focusing the DropDownList and typing the characters in it. The closely matched items are selected sequentially.

If the same key is searched once again, the next matched item is selected.

Modify data in React Drop down list component

When binding the remote data source, by using the [actionComplete](#) event, you can modify the result data before passing it to DropDownList.

The following sample demonstrate how to modify the result data.

INDEX.JSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    customerData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    // bind the Query instance to query property
    query = new Query().from('Customers').select(['ContactName',
    'CustomerID']).take(6);
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // sort the resulted items
    sortOrder = 'Ascending';
    onComplete(e) {
        // initially result contains 6 items
        console.log("Before modified the result: " + e.result.length);
        // remove first 2 items from result.
        e.result.splice(0, 2);
        // now displays the result count is 4.
        console.log("After modified the result: " + e.result.length);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" query={this.query}
            dataSource={this.customerData} fields={this.fields}
            sortOrder={this.sortOrder} placeholder="Select a customer"
            actionComplete={this.onComplete}/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
```

```

// bind the DataManager instance to dataSource property
private customerData: DataManager = new DataManager({
  adaptor: new ODataV4Adaptor,
  crossDomain: true,
  url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
});
// bind the Query instance to query property
private query: Query = new
Query().from('Customers').select(['ContactName', 'CustomerID']).take(6);
// maps the appropriate column to fields property
private fields: object = { text: 'ContactName', value: 'CustomerID' };
// sort the resulted items
private sortOrder: SortOrder = 'Ascending';
public onComplete(e: any){
  // initially result contains 6 items
  console.log("Before modified the result: " + e.result.length);
  // remove first 2 items from result.
  e.result.splice(0, 2);
  // now displays the result count is 4.
  console.log("After modified the result: " + e.result.length);
}
public render() {
  return (
    // specifies the tag for render the DropDownList component
    <DropDownListComponent id="ddlelement" query={this.query}
dataSource={this.customerData}
fields={this.fields} sortOrder={this.sortOrder}
placeholder="Select a customer" actionComplete={this.onComplete} />
  );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Multiple cascading in React Drop down list component

The following example demonstrate about how to preselect the list items in multiple cascading DropDownList.

[Class-component]

INDEX.JSX

```

{% raw %}
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // country DropDownList instance
  countryObj;
  // state DropDownList instance
  stateObj;
  // city DropDownList instance
  cityObj;
  // define the country DropDownList data
  countryData = [
    { CountryName: 'Australia', CountryId: '2' },

```

```

        { CountryName: 'United States', CountryId: '1' }
    ];
    // define the state DropDownList data
    stateData = [
        { StateName: 'New York', CountryId: '1', StateId: '101' },
        { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
        { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
    ];
    // define the city DropDownList data
    cityData = [
        { CityName: 'Albany', StateId: '101', CityId: 201 },
        { CityName: 'Beacon ', StateId: '101', CityId: 202 },
        { CityName: 'Emporia', StateId: '102', CityId: 206 },
        { CityName: 'Hampton ', StateId: '102', CityId: 205 },
        { CityName: 'Hobart', StateId: '105', CityId: 213 },
        { CityName: 'Launceston ', StateId: '105', CityId: 214 }
    ];
    // maps the country column to fields property
    countryField = { value: 'CountryId', text: 'CountryName' };
    // maps the state column to fields property
    stateField = { value: 'StateId', text: 'StateName' };
    // maps the city column to fields property
    cityField = { text: 'CityName', value: 'CityId' };
    // Index no of the country dropdownlist
    index = 1;
    constructor(props) {
        super(props);
        this.onCountryChange = this.onCountryChange.bind(this);
        this.onStateChange = this.onStateChange.bind(this);
    }
    onCountryChange() {
        // query the data source based on country DropDownList selected value
        this.stateObj.query = new Query().where('CountryId', 'equal',
this.countryObj.value);
        // enable the state DropDownList
        this.stateObj.enabled = true;
        // clear the existing selection.
        this.stateObj.text = null;
        // bind the property changes to state DropDownList
        this.stateObj.dataBind();
        // clear the existing selection in city DropDownList
        this.cityObj.text = null;
        // disable the city DropDownList
        this.cityObj.enabled = false;
        // bind the property change to City DropDownList
        this.cityObj.dataBind();
    }
    onStateChange() {
        // query the data source based on state DropDownList selected value
        this.cityObj.query = new Query().where('StateId', 'equal',
this.stateObj.value);
        // enable the city DropDownList
        this.cityObj.enabled = true;
        // clear the existing selection
        this.cityObj.text = null;
        // bind the property change to city DropDownList
        this.cityObj.dataBind();
    }

```

```

    }
    componentDidMount() {
        this.onCountryChange();
        // preselect an item from filtered state DropDownList
        this.stateObj.index = 0;
        this.stateObj.dataBind();
        // initially change event not triggered, so manually call the
        corresponding function
        this.onStateChange();
        // preselect an item from filtered city DropDownList
        this.cityObj.index = 0;
        this.cityObj.dataBind();
    }
    render() {
        return (<div>
            /* specifies the tag for render the country DropDownList component
            */
            <DropDownListComponent id="country-ddl" ref={(scope) => {
                this.countryObj = scope; }} index={this.index} fields={this.countryField}
                dataSource={this.countryData} placeholder='Select a country'
                change={this.onCountryChange}/>
            <br />
            /* specifies the tag for render the state DropDownList component */
            <DropDownListComponent id="state-ddl" ref={(scope) => { this.stateObj
                = scope; }} enabled={false} fields={this.stateField}
                dataSource={this.stateData} placeholder='Select a state'
                change={this.onStateChange}/>
            <br />
            /* specifies the tag for render the city DropDownList component */
            <DropDownListComponent id="city-ddl" ref={(scope) => { this.cityObj =
                scope; }} enabled={false} fields={this.cityField} dataSource={this.cityData}
                placeholder='Select a city' />
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // country DropDownList instance
    private countryObj: any;
    // state DropDownList instance
    private stateObj: any;
    // city DropDownList instance
    private cityObj: any;
    // define the country DropDownList data
    private countryData: { [key: string]: Object }[] = [
        { CountryName: 'Australia', CountryId: '2' },
        { CountryName: 'United States', CountryId: '1' }
    ]
}

```



```

];
// define the state DropDownList data
private stateData: { [key: string]: Object }[] = [
  { StateName: 'New York', CountryId: '1', StateId: '101' },
  { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
  { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
];
// define the city DropDownList data
private cityData: { [key: string]: Object }[] = [
  { CityName: 'Albany', StateId: '101', CityId: 201 },
  { CityName: 'Beacon ', StateId: '101', CityId: 202 },
  { CityName: 'Emporia', StateId: '102', CityId: 206 },
  { CityName: 'Hampton ', StateId: '102', CityId: 205 },
  { CityName: 'Hobart', StateId: '105', CityId: 213 },
  { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
// maps the country column to fields property
private countryField: object = { value: 'CountryId', text: 'CountryName' };
// maps the state column to fields property
private stateField: object = { value: 'StateId', text: 'StateName' };
// maps the city column to fields property
private cityField: object = { text: 'CityName', value: 'CityId' };
// Index no of the country dropdownlist
private index: number = 1;
constructor(props: any) {
  super(props);
  this.onCountryChange = this.onCountryChange.bind(this);
  this.onStateChange = this.onStateChange.bind(this);
}
public onCountryChange() {
  // query the data source based on country DropDownList selected value
  this.stateObj.query = new Query().where('CountryId', 'equal',
this.countryObj.value);
  // enable the state DropDownList
  this.stateObj.enabled = true;
  // clear the existing selection.
  this.stateObj.text = null;
  // bind the property changes to state DropDownList
  this.stateObj.dataBind();
  // clear the existing selection in city DropDownList
  this.cityObj.text = null;
  // disable the city DropDownList
  this.cityObj.enabled = false;
  // bind the property change to City DropDownList
  this.cityObj.dataBind();
}
public onStateChange() {
  // query the data source based on state DropDownList selected value
  this.cityObj.query = new Query().where('StateId', 'equal',
this.stateObj.value);
  // enable the city DropDownList
  this.cityObj.enabled = true;
  // clear the existing selection
  this.cityObj.text = null;
  // bind the property change to city DropDownList
  this.cityObj.dataBind();
}
}

```

```

public componentDidMount() {
    this.onCountryChange();
    // preselect an item from filtered state DropDownList
    this.stateObj.index = 0;
    this.stateObj.dataBind();
    // initially change event not triggered, so manually call the
    corresponding function
    this.onStateChange();
    // preselect an item from filtered city DropDownList
    this.cityObj.index = 0;
    this.cityObj.dataBind();
}
public render() {
    return (
        <div>
            /* specifies the tag for render the country DropDownList component
            */
            <DropDownListComponent id="country-ddl" ref={(scope) => {
                this.countryObj = scope; }} index={this.index} fields={this.countryField}
                dataSource={this.countryData} placeholder='Select a country'
                change={this.onCountryChange} />
            <br />
            /* specifies the tag for render the state DropDownList component */
            <DropDownListComponent id="state-ddl" ref={(scope) => { this.stateObj
                = scope; }} enabled={false} fields={this.stateField}
                dataSource={this.stateData} placeholder='Select a state'
                change={this.onStateChange} />
            <br />
            /* specifies the tag for render the city DropDownList component */
            <DropDownListComponent id="city-ddl" ref={(scope) => { this.cityObj =
                scope; }} enabled={false} fields={this.cityField} dataSource={this.cityData}
                placeholder='Select a city' />
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

[Functional-component]

INDEX.JSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { useState, useRef } from 'react';
import { Query } from '@syncfusion/ej2-data';
function App() {
    // state DropDownList instance
    const stateObj = useRef(null);
    // city DropDownList instance
    const cityObj = useRef(null);
    //define the country DropDownList data
    const countryData = [
        { CountryName: 'Australia', CountryId: '2' },
    ]
}

```

```

    { CountryName: 'United States', CountryId: '1' }
  ];
  //define the state DropDownList data
  const stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
  ];
  //define the city DropDownList data
  const cityData = [
    { CityName: 'Albany', StateId: '101', CityId: 201 },
    { CityName: 'Beacon ', StateId: '101', CityId: 202 },
    { CityName: 'Emporia', StateId: '102', CityId: 206 },
    { CityName: 'Hampton ', StateId: '102', CityId: 205 },
    { CityName: 'Hobart', StateId: '105', CityId: 213 },
    { CityName: 'Launceston ', StateId: '105', CityId: 214 }
  ];
  // maps the country column to fields property
  const countryFields = { value: 'CountryId', text: 'CountryName' };
  // maps the state column to fields property
  const stateFields = { value: 'StateId', text: 'StateName' };
  // maps the city column to fields property
  const cityFields = { text: 'CityName', value: 'CityId' };
  const [stateQuery, setStateQuery] = useState(null);
  const [stateText, setStateText] = useState(null);
  const [cityText, setCityText] = useState(null);
  const [cityQuery, setCityQuery] = useState(null);
  const countryChange = (args) => {
    // query the data source based on country DropDownList selected value
    let tempQuery = new Query().where('CountryId', 'equal', args.value);
    setStateQuery(tempQuery);
    // clear the existing selection.
    setStateText(null);
    // bind the property changes to state DropDownList
    stateObj.current.dataBind();
    // clear the existing selection.
    setCityText(null);
    // bind the property changes to city DropDownList
    cityObj.current.dataBind();
  };
  const stateChange = (args) => {
    // query the data source based on state DropDownList selected value
    let tempQuery1 = new Query().where('StateId', 'equal', args.value);
    setCityQuery(tempQuery1);
    // clear the existing selection.
    setCityText(null);
  };
  return (
    <div id="cascade">
      <div style={{ paddingTop: '35px' }}>
        <DropDownListComponent
          id="country"
          dataSource={countryData}
          fields={countryFields}
          popupHeight="auto"
          change={countryChange.bind(this)}
          placeholder="Select a country"

```

```

    />
  </div>
  <div style={{ paddingTop: '35px' }}>
    <DropDownListComponent
      id="state"
      dataSource={stateData}
      ref={stateObj}
      fields={stateFields}
      popupHeight="auto"
      change={stateChange.bind(this)}
      placeholder="Select a state"
      query={stateQuery}
      text={stateText}
    />
  </div>
  <div style={{ paddingTop: '35px' }}>
    <DropDownListComponent
      id="city"
      dataSource={cityData}
      ref={cityObj}
      fields={cityFields}
      popupHeight="auto"
      placeholder="Select a city"
      text={cityText}
      query={cityQuery}
    />
  </div>
</div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { useState, useRef } from 'react';
import { Query } from '@syncfusion/ej2-data';
function App() {
  // state DropDownList instance
  const stateObj = useRef(null);
  // city DropDownList instance
  const cityObj = useRef(null);
  const tempCountry = 'country';
  //define the country DropDownList data
  const countryData = [
    { CountryName: 'Australia', CountryId: '2' },
    { CountryName: 'United States', CountryId: '1' }
  ];
  const tempState = 'state';
  //define the state DropDownList data
  const stateData = [
    { StateName: 'New York', CountryId: '1', StateId: '101' },
    { StateName: 'Virginia ', CountryId: '1', StateId: '102' },
    { StateName: 'Tasmania ', CountryId: '2', StateId: '105' }
  ];

```

```

];
const tempCity = 'cities';
//define the city DropDownList data
const cityData = [
  { CityName: 'Albany', StateId: '101', CityId: 201 },
  { CityName: 'Beacon ', StateId: '101', CityId: 202 },
  { CityName: 'Emporia', StateId: '102', CityId: 206 },
  { CityName: 'Hampton ', StateId: '102', CityId: 205 },
  { CityName: 'Hobart', StateId: '105', CityId: 213 },
  { CityName: 'Launceston ', StateId: '105', CityId: 214 }
];
// maps the country column to fields property
const countryFields = { value: 'CountryId', text: 'CountryName' };
// maps the state column to fields property
const stateFields = { value: 'StateId', text: 'StateName' };
// maps the city column to fields property
const cityFields = { text: 'CityName', value: 'CityId' };
const [stateQuery, setStateQuery] = useState(null);
const [stateText, setStateText] = useState(null);
const [cityText, setCityText] = useState(null);
const [cityQuery, setCityQuery] = useState(null);
const countryChange = (args) => {
  // query the data source based on country DropDownList selected value
  let tempQuery = new Query().where('CountryId', 'equal', args.value);
  setStateQuery(tempQuery);
  // clear the existing selection.
  setStateText(null);
  // bind the property changes to state DropDownList
  stateObj.current.dataBind();
  // clear the existing selection.
  setCityText(null);
  // bind the property changes to city DropDownList
  cityObj.current.dataBind();
};
const stateChange = (args) => {
  // query the data source based on state DropDownList selected value
  let tempQuery1 = new Query().where('StateId', 'equal', args.value);
  setCityQuery(tempQuery1);
  // clear the existing selection.
  setCityText(null);
};
return (
  <div id="cascade">
    <div style={{ paddingTop: '35px' }}>
      <DropDownListComponent
        id="country"
        dataSource={countryData}
        fields={countryFields}
        popupHeight="auto"
        change={countryChange.bind(this)}
        placeholder="Select a country"
      />
    </div>
    <div style={{ paddingTop: '35px' }}>
      <DropDownListComponent
        id="state"
        dataSource={stateData}

```

```

        ref={stateObj}
        fields={stateFields}
        popupHeight="auto"
        change={stateChange.bind(this)}
        placeholder="Select a state"
        query={stateQuery}
        text={stateText}
      />
    </div>
    <div style={{ paddingTop: '35px' }}>
      <DropDownListComponent
        id="city"
        dataSource={cityData}
        ref={cityObj}
        fields={cityFields}
        popupHeight="auto"
        placeholder="Select a city"
        text={cityText}
        query={cityQuery}
      />
    </div>
  </div>
);
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Remote data bind in React Drop down list component

Before component rendering, you can get the total items count by using [actionComplete](#) event with its result arguments.

After rendering this component, you can get the total items count by using [getItem](#) method.

The following example demonstrate how to get the total items count.

INDEX.JSX

```

{% raw %}
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // bind the DataManager instance to dataSource property
  customerData = new DataManager({
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
  });
  dropDownListObject;
  // bind the Query instance to query property
  query = new Query().from('Customers').select(['ContactName',
    'CustomerID']).take(6);
  // maps the appropriate column to fields property
  fields = { text: 'ContactName', value: 'CustomerID' };
  // sort the resulted items

```

```

    sortOrder = 'Ascending';
    constructor(props) {
        super(props);
        this.onclick = this.onclick.bind(this);
    }
    onActionComplete(e) {
        // get total items count
        console.log("Total items count: " + e.result.length);
        const element = document.createElement('p');
        element.innerText = "Total items count: " + e.result.length;
        document.getElementById('event').append(element);
    }
    onclick() {
        // get items count using getItems method
        console.log("Total items count: " +
this.dropDownListObject.listData.length);
        const element = document.createElement('p');
        element.innerText = "Total items count: " +
this.dropDownListObject.listData.length;
        document.getElementById('event').append(element);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent ref={ (scope) => { this.dropDownListObject =
scope; }} id="ddlelement" query={this.query} dataSource={this.customerData}
fields={this.fields} sortOrder={this.sortOrder} placeholder="Select a
customer" actionComplete={this.onActionComplete}/>
                <div id='event' />
                <button id='btn' className="e-control e-btn" onClick={this.onclick}>
Get items </button>
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
// import DataManager related classes
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private customerData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/'
    });
    private dropDownListObject: any;
    // bind the Query instance to query property

```

```

    private query: Query = new Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // sort the resulted items
    private sortOrder: string = 'Ascending';
    constructor(props: any) {
        super(props);
        this.onclick = this.onclick.bind(this);
    }
    public onActionComplete(e: any) {
        // get total items count
        console.log("Total items count: " + e.result.length);
        const element: HTMLElement = document.createElement('p');
        element.innerText = "Total items count: " + e.result.length;
        (document.getElementById('event') as any).append(element);
    }
    public onclick() {
        // get items count using getItems method
        console.log("Total items count: " +
this.dropDownListObject.listData.length);
        const element: HTMLElement = document.createElement('p');
        element.innerText = "Total items count: " +
this.dropDownListObject.listData.length;
        (document.getElementById('event') as any).append(element);
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent ref={ (scope) => { this.dropDownListObject =
scope; }} id="ddlelement"
                    query={this.query} dataSource={this.customerData}
                    fields={this.fields} sortOrder={this.sortOrder} placeholder="Select
a customer" actionComplete={this.onActionComplete} />
                <div id='event' />
                <button id='btn' className="e-control e-btn" onClick={this.onclick}>
Get items </button>
                </div>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

Remove item in React Drop down list component

The following example demonstrate about how to remove an item from DropDownList.

INDEX.JSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data

```



```

sportsData = [
    { Id: 'game1', Game: 'Badminton' },
    { Id: 'game2', Game: 'Football' },
    { Id: 'game3', Game: 'Tennis' }
];
// maps the appropriate column to fields property
fields = { text: 'Game', value: 'Id' };
dropDownListObject;
onclick() {
    if (this.dropDownListObject.list) {
        // Remove the selected value if 0th index selected
        if (this.dropDownListObject.index === 0) {
            this.dropDownListObject.value = null;
            this.dropDownListObject.dataBind();
        }
        // remove first item in list
        (this.dropDownListObject.list.querySelectorAll('li')[0]).remove();
        if (!this.dropDownListObject.list.querySelectorAll('li')[0]) {
            this.dropDownListObject.dataSource = [];
            // enable the nodata template when no data source is empty.
            this.dropDownListObject.list.classList.add('e-nodata');
        }
    }
    else {
        // remove first item in list
        this.dropDownListObject.dataSource.splice(0, 1);
    }
}
render() {
    return (
        // specifies the tag for render the DropDownList component
        <div>
            <DropDownListComponent id="ddlelement" ref={(scope) => {
                this.dropDownListObject = scope; }} dataSource={this.sportsData}
                fields={this.fields} placeholder="Select a game"/>
            <button id='btn' className="e-control e-btn"
                onClick={this.onclick = this.onclick.bind(this)}>
                Remove 0th item </button>
            </div>);
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: { [key: string]: Object }[] = [
        { Id: 'game1', Game: 'Badminton' },
        { Id: 'game2', Game: 'Football' },

```

```

        { Id: 'game3', Game: 'Tennis' }
    ];
    // maps the appropriate column to fields property
    private fields: object = { text: 'Game', value: 'Id' };
    private dropDownListObject:any;
    public onclick(){
        if (this.dropDownListObject.list) {
            // Remove the selected value if 0th index selected
            if (this.dropDownListObject.index === 0) {
                this.dropDownListObject.value = null;
                this.dropDownListObject.dataBind();
            }
            // remove first item in list
            (this.dropDownListObject.list.querySelectorAll('li')[0]).remove();
            if (!this.dropDownListObject.list.querySelectorAll('li')[0]) {
                this.dropDownListObject.dataSource = [];
                // enable the nodata template when no data source is empty.
                this.dropDownListObject.list.classList.add('e-nodata');
            }
        } else {
            // remove first item in list
            this.dropDownListObject.dataSource.splice(0, 1);
        }
    }
    public render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent id="ddlelement" ref={ (scope) => {
                    this.dropDownListObject = scope; }} dataSource={this.sportsData}
                    fields={this.fields} placeholder="Select a game" />
                <button id='btn' className="e-control e-btn"
                    onClick={this.onclick = this.onclick.bind(this)}>
                    Remove 0th item </button>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

Search on filtering in React Drop down list component

The following example demonstrates about how to set limit the search result on filtering.

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // bind the DataManager instance to dataSource property
    searchData = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
    });
}

```

```

        url:
        'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    fields = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    query = new Query().select(['ContactName', 'CustomerID']).take(7);
    // sort the resulted items
    sortOrder = 'Ascending';
    constructor(props) {
        super(props);
        this.onFiltering = this.onFiltering.bind(this);
    }
    // filtering event handler to filter a customer
    onFiltering(e) {
        // set limit as 4 to search result
        let query = new Query().select(['ContactName',
        'CustomerID']).take(4);
        query = (e.text !== '') ? query.where('ContactName', 'startswith',
        e.text, true) : query;
        e.updateData(this.searchData, query);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" allowFiltering={true}
            popupHeight="250px" filtering={this.onFiltering} sortOrder={this.sortOrder}
            query={this.query} dataSource={this.searchData} fields={this.fields}
            placeholder="Select a customer"/>);
        )
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { SortOrder } from '@syncfusion/ej2-lists';
import { DropDownListComponent, FilteringEventArgs } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // bind the DataManager instance to dataSource property
    private searchData: DataManager = new DataManager({
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
        url: 'https://services.odata.org/V4/Northwind/Northwind.svc/Customers'
    });
    // maps the appropriate column to fields property
    private fields: object = { text: 'ContactName', value: 'CustomerID' };
    // bind the Query instance to query property
    private query: Query = new Query().select(['ContactName',
    'CustomerID']).take(7);
    // sort the resulted items
    private sortOrder: SortOrder = 'Ascending';
    constructor(props: any) {

```

```

    super(props);
    this.onFiltering = this.onFiltering.bind(this);
  }
  // filtering event handler to filter a customer
  public onFiltering(e: FilteringEventArgs) {
    // set limit as 4 to search result
    let query: Query = new Query().select(['ContactName',
    'CustomerID']).take(4);
    query = (e.text !== '') ? query.where('ContactName', 'startswith',
    e.text, true) : query;
    e.updateData(this.searchData, query);
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <DropDownListComponent id="ddlelement" allowFiltering={true}
      popupHeight="250px" filtering={this.onFiltering} sortOrder={this.sortOrder}
      query={this.query} dataSource={this.searchData} fields={this.fields}
      placeholder="Select a customer" />
    );
  }
}
ReactDOM.render(<App />, document.getElementById('sample'));

```

Tooltip in React Drop down list component

You can achieve this behavior by using `ej2-tooltip` component. When the mouse hover on the DropDownList option that tooltip display some details related to hovered list item.

INDEX.JSX

```

{% raw %}
import { Tooltip } from '@syncfusion/ej2-popups';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
  // define the JSON of data
  countryData = [
    { id: '1', text: 'Australia', content: 'National sports is Cricket' },
    { id: '2', text: 'Bhutan', content: 'National sports is Archery' },
    { id: '3', text: 'China', content: 'National sports is Table Tennis' },
    { id: '4', text: 'Cuba', content: 'National sports is Baseball' },
    { id: '5', text: 'India', content: 'National sports is Hockey' },
    { id: '6', text: 'Spain', content: 'National sports is Football' },
    { id: '7', text: 'United States', content: 'National sports is Baseball' }
  ];
  // maps the appropriate column to fields property
  fields = { text: 'text', value: 'id' };
  tooltip;
  dropDownListObject;
  constructor(props) {
    super(props);
    this.onBeforeRender = this.onBeforeRender.bind(this);
  }

```

```

        this.onClose = this.onClose.bind(this);
    }
    onBeforeRender(args) {
        // get the target element
        const result = this.dropDownListObject.dataSource;
        let i;
        for (i = 0; i < result.length; i++) {
            if (result[i].text === args.target.textContent) {
                this.tooltip.content = result[i].content;
                this.tooltip.dataBind();
                break;
            }
        }
    }
    onClose() {
        this.tooltip.close();
    }
    componentDidMount() {
        this.tooltip = new Tooltip({
            beforeRender: this.onBeforeRender,
            content: 'Loading...',
            position: 'TopCenter',
            target: '.e-list-item'
        });
        this.tooltip.appendTo('body');
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <DropDownListComponent id="ddlelement" ref={(scope) => {
                this.dropDownListObject = scope; }} dataSource={this.countryData}
                fields={this.fields} placeholder="Select a country" close={this.onClose}/>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { Tooltip } from '@syncfusion/ej2-popups';
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the JSON of data
    private countryData: { [key: string]: Object }[] = [
        { id: '1', text: 'Australia', content: 'National sports is Cricket' },
        { id: '2', text: 'Bhutan', content: 'National sports is Archery' },
        { id: '3', text: 'China', content: 'National sports is Table Tennis' },
        { id: '4', text: 'Cuba', content: 'National sports is Baseball' },
        { id: '5', text: 'India', content: 'National sports is Hockey' },
        { id: '6', text: 'Spain', content: 'National sports is Football' },
        { id: '7', text: 'United States', content: 'National sports is Baseball' }
    ]
};

```

```

// maps the appropriate column to fields property
private fields: object = { text: 'text', value: 'id' };
private tooltip: any;
private dropDownListObject: any;
constructor(props: any) {
    super(props);
    this.onBeforeRender = this.onBeforeRender.bind(this);
    this.onClose = this.onClose.bind(this);
}
public onBeforeRender(args: any) {
    // get the target element
    const result = this.dropDownListObject.dataSource;
    let i;
    for (i = 0; i < result.length; i++) {
        if (result[i].text === args.target.textContent) {
            this.tooltip.content = result[i].content;
            this.tooltip.dataBind();
            break;
        }
    }
}
public onClose() {
    this.tooltip.close();
}
public componentDidMount() {
    this.tooltip = new Tooltip({
        beforeRender: this.onBeforeRender,
        content: 'Loading...',
        position: 'TopCenter',
        target: '.e-list-item'
    });
    this.tooltip.appendTo('body');
}
public render() {
    return (
        // specifies the tag for render the DropDownList component
        <DropDownListComponent id="ddlelement" ref={(scope) => {
            this.dropDownListObject = scope; }} dataSource={this.countryData}
            fields={this.fields} placeholder="Select a country" close={this.onClose} />
    );
}
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

Value change in React Drop down list component

You can check about whether value change happened by manual or programmatic by using [change](#) event argument that argument name is `isInteracted`.

The following example demonstrate, how to check whether value change happened by manual or programmatic.

INDEX.JSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';

```

```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component {
    // define the array of data
    sportsData = ['Badminton', 'Cricket', 'Football', 'Golf', 'Tennis'];
    dropDownListObject;
    onclick() {
        this.dropDownListObject.value = 'Football';
    }
    onChange(args) {
        const element = document.createElement('p');
        if (args.isInteracted) {
            element.innerText = 'Changes happened by Interaction';
        }
        else {
            element.innerText = 'Changes happened by programmatic';
        }
        document.getElementById('event').append(element);
    }
    render() {
        return (
            // specifies the tag for render the DropDownList component
            <div>
                <DropDownListComponent id="ddlelement" ref={(scope) => {
                    this.dropDownListObject = scope; }} dataSource={this.sportsData}
                    placeholder="Select a game" change={this.onChange}/>
                <button id='btn' className="e-control e-btn" onClick={this.onclick =
                    this.onclick.bind(this)}>
                    Set value dynamically</button>
                <div id='event' />
            </div>);
    }
}
ReactDOM.render(<App />, document.getElementById('sample'));
{% endraw %}

```

INDEX.TSX

```

{% raw %}
import { DropDownListComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
export default class App extends React.Component<{}, {}> {
    // define the array of data
    private sportsData: string[] = ['Badminton', 'Cricket', 'Football', 'Golf',
    'Tennis'];
    private dropDownListObject:any;
    public onclick(){
        this.dropDownListObject.value = 'Football';
    }
    public onChange(args: any){
        const element: HTMLElement = document.createElement('p');
        if (args.isInteracted) {
            element.innerText = 'Changes happened by Interaction';
        }
        else {
            element.innerText = 'Changes happened by programmatic';
        }
    }
}

```

```

    }
    (document.getElementById('event') as any).append(element);
  }
  public render() {
    return (
      // specifies the tag for render the DropDownList component
      <div>
        <DropDownListComponent id="ddlelement" ref={(scope) => {
          this.dropDownListObject = scope; }} dataSource={this.sportsData}
          placeholder="Select a game" change={this.onChange} />
        <button id='btn' className="e-control e-btn" onClick={this.onClick =
          this.onClick.bind(this)}>
          Set value dynamically</button>
        <div id='event' />
        </div>
      );
    }
  }
  ReactDOM.render(<App />, document.getElementById('sample'));
  {% endraw %}

```

Value support in React Drop down list component

yes, value for each list items should be unique.

Ej1 api migration in React Drop down list component

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

DataBinding

{% raw %}

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *dataSource*
<EJ.DropDownList dataSource={groups}></EJ.DropDownList> | **Property:** *dataSource*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }} dataSource={this.cityData} />|

| **Fields for mapping** | **Property:** *fields*
<EJ.DropDownList fields-value="parentId" fields-text="text" ></EJ.DropDownList>| **Property:** *fields*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }} fields={this.cityField} />|

| **Query** | **Property:** *query*
<EJ.DropDownList query={query}></EJ.DropDownList>| **Property:** *query*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }} query={this.query} />|

| **Begin event** | **Event:** *actionBegin*
<EJ.DropDownList actionBegin="actionBegin"></EJ.DropDownList> | **Event:** *actionBegin*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }} actionBegin={this.actionBegin.bind(this)} />|

| **Complete event** | **Event:** *actionComplete*
<EJ.DropDownList
 actionComplete="actionComplete"></EJ.DropDownList> | **Event:** *actionComplete*

<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 actionComplete={this.actionComplete.bind(this)} /> |

| **Failure event** | **Event:** *actionFailure*
<EJ.DropDownList
 actionFailure="actionFailure"></EJ.DropDownList> | **Event:** *actionFailure*

<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 actionFailure={this.actionFailure.bind(this)} /> |

| **Success event** | **Event:** *actionSuccess*
<EJ.DropDownList
 actionSuccess="actionSuccess"></EJ.DropDownList> | **Not Applicable** |

| **Data binding event** | **Event:** *dataBound*
<EJ.DropDownList
 dataBound="dataBound"></EJ.DropDownList> | **Event:** *dataBind*
<DropDownListComponent
 id="ddl" ref={{scope}} => { this.ddlObj = scope; }} dataBind={this.dataBind.bind(this)} /> |

Filtering

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *enableFilterSearch*
<EJ.DropDownList
 enableFilterSearch={true}></EJ.DropDownList> | **Property:** *allowFiltering*

<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 allowFiltering={true} /> |

| **Server filtering** | **Property:** *enableServerFiltering*
<EJ.DropDownList
 enableServerFiltering={true}></EJ.DropDownList> | **Property:** *allowFiltering*

<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 allowFiltering={true} /> |

| **Filter type** | **Property:** *filterType*
<EJ.DropDownList
 filterType="contains"></EJ.DropDownList>
[| https://ej2.syncfusion.com/react/demos/#/material/drop-down-list/filtering](https://ej2.syncfusion.com/react/demos/#/material/drop-down-list/filtering) |

| **No Records Template** | **Not Applicable** | **Property:** *noRecordsTemplate*

 <DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 noRecordsTemplate={this.noRecordsTemplate} /> |

| **Filter Bar watermark text** | **Not Applicable** | **Property:** *filterBarPlaceholder*

<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 filterBarPlaceholder={this.filterBarPlaceholder} /> |

| **Ignore casing and diacritics** | **Not Applicable** | **Property:**
ignoreAccent
<DropDownListComponent id="ddl" ref={{scope}} => { this.ddlObj = scope; }}
 ignoreAccent={true} /> |

| **Incremental search** | **Property:** *enableIncrementalSearch*
<EJ.DropDownList
 enableIncrementalSearch={true}></EJ.DropDownList> | **By default it is true** |

| **Case sensitivity** | **Property:** *caseSensitiveSearch*
<EJ.DropDownList

caseSensitiveSearch={true}></EJ.DropDownList> |

<https://ej2.syncfusion.com/react/demos/#/material/drop-down-list/filtering> |

| **Search event** | **Event:** *search*
<EJ.DropDownList search="search"></EJ.DropDownList> |

Event: *filtering*
<DropDownListComponent id="ddl" ref={({scope}) => { this.ddlObj = scope; }}
filtering={this.filtering.bind(this)} /> |

Template

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *template*
<EJ.DropDownList template={template}></EJ.DropDownList>

| **Property:** *itemTemplate*
<DropDownListComponent id="ddl" ref={({scope}) => { this.ddlObj =
scope; }} itemTemplate={this.itemTemplate} /> |

| **Group Template** | **Not Applicable** | **Property:** *groupTemplate*
<DropDownListComponent
id="ddl" ref={({scope}) => { this.ddlObj = scope; }} groupTemplate={this.groupTemplate} /> |

| **ValueTemplate** | **Not Applicable** | **Property:** *valueTemplate*
<DropDownListComponent
id="ddl" ref={({scope}) => { this.ddlObj = scope; }} valueTemplate={this.valueTemplate} /> |

| **Header Template** | **Property:** *headerTemplate*
<EJ.DropDownList
headerTemplate={headerTemplate}></EJ.DropDownList> | **Property:** *headerTemplate*

<DropDownListComponent id="ddl" ref={({scope}) => { this.ddlObj = scope; }}
headerTemplate={this.headerTemplate} /> |

| **FooterTemplate** | **Not applicable** | **Property:** *footerTemplate*
<DropDownListComponent
id="ddl" ref={({scope}) => { this.ddlObj = scope; }} footerTemplate={this.footerTemplate} /> |

| **No records Template** | **Not applicable** | **Property:** *noRecordsTemplate*

<DropDownListComponent id="ddl" ref={({scope}) => { this.ddlObj = scope; }}
noRecordsTemplate={this.noRecordsTemplate} /> |

| **Action failure Template** | **Not applicable** | **Property:** *actionFailureTemplate*

<DropDownListComponent id="ddl" ref={({scope}) => { this.ddlObj = scope; }}
actionFailureTemplate={this.actionFailureTemplate} /> |

Virtual Scrolling

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *allowVirtualScrolling*
<EJ.DropDownList

allowVirtualScrolling={true}></EJ.DropDownList> | **Not applicable** |

Applying CSS

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *cssClass* `
<EJ.DropDownList
cssClass="customClass"></EJ.DropDownList>` | **Property:** *cssClass*
`
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
cssClass={this.cssclass} />|`

| **showRoundedCorner** | **Property:** *showRoundedCorner* `
<EJ.DropDownList
showRoundedCorner={true}></EJ.DropDownList>` | Achievable through the [cssClass](#) property. |

Sorting

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *enableSorting* `
<EJ.DropDownList
enableSorting={true}></EJ.DropDownList>` | Achievable through the [sortOrder](#) property. |

| **Order of sorting** | **Property:** *sortOrder* `
<EJ.DropDownList
sortOrder="ascending"></EJ.DropDownList>` | **Property:** *sortOrder*
`
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
sortOrder={this.sortOrder} />|`

Popup

<!-- markdownlint-disable MD010 -->

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

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

| **Popup height** | **Property:** *popupHeight* `
<EJ.DropDownList
popupHeight="500px"></EJ.DropDownList>` | **Property:** *popupHeight*
`
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
popupHeight={this.popupHeight} />|`

| **Popup width** | **Property:** *popupWidth* `
<EJ.DropDownList
popupWidth="500px"></EJ.DropDownList>` | **Property:** *popupWidth*
`
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
popupWidth={this.popupWidth} />|`

| **Popup show on load** | **Property:** *showPopupOnLoad* `
<EJ.DropDownList
showPopupOnLoad={true}></EJ.DropDownList>` | **By default, the data load on demand.** |

| **enableAnimation** | **Property:** *enableAnimation* `
<EJ.DropDownList
enableAnimation={true}></EJ.DropDownList>` | **Not applicable** |

| **Popup resizing** | **Property:** *enablePopupResize* `
<EJ.DropDownList
enablePopupResize={true}></EJ.DropDownList>` | **Not applicable** |

| **Maximum Popup height** | **Property:** *maxPopupHeight* `
<EJ.DropDownList
maxPopupHeight="500px"></EJ.DropDownList>` | **Not applicable** |

| **Minimum Popup height** | **Property:** *minPopupHeight*
<EJ.DropDownList
minPopupHeight="500px"></EJ.DropDownList>
}); | **Not applicable** |

| **Maximum Popup width** | **Property:** *maxPopupWidth*
<EJ.DropDownList
maxPopupWidth="500px"></EJ.DropDownList> | **Not applicable** |

| **Minimum Popup width** | **Property:** *minPopupWidth*
<EJ.DropDownList
minPopupWidth="500px"></EJ.DropDownList> | **Not applicable** |

| **Loading data** | **Property:** *loadOnDemand*
<EJ.DropDownList
loadOnDemand={true}></EJ.DropDownList> | **By default, it is true** |

| **Popup showing manually** | **Method:** *showPopup*

<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('showPopup'
) | **Method:** *showPopup*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj =
scope; }} />
this.ddlObj.showPopup();|

| **Popup hiding manually** | **Method:** *hidePopup*

<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('hidePopup') |
Method: *hidePopup*
<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj =
scope; }} />
this.ddlObj.hidePopup();|

| **Before Popup hide event** | **Event:** *beforePopupHide*
<EJ.DropDownList
beforePopupHide="beforePopupHide"></EJ.DropDownList> | **Not applicable** |

| **Before Popup shown event** | **Event:** *beforePopupShown*
<EJ.DropDownList
beforePopupShown="beforePopupShown"></EJ.DropDownList> | **Event:** *beforeOpen*

<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }}
(beforeOpen)={this.beforeOpen.bind(this)} />|

| **Popup hide event** | **Event:** *popupHide*
<EJ.DropDownList
popupHide="popupHide"></EJ.DropDownList> | **Event:** *close*
<DropDownListComponent
id="ddl" ref={(scope) => { this.ddlObj = scope; }} (close)={this.close.bind(this)} /> |

| **Popup resize event** | **Event:** *popupResize*
<EJ.DropDownList
popupResize="popupResize"></EJ.DropDownList> | **Not applicable** |

| **Popup resize start event** | **Event:** *popupResizeStart*
<EJ.DropDownList
popupResizeStart="popupResizeStart"></EJ.DropDownList> | **Not applicable** |

| **Popup resize stop event** | **Event:** *popupResizeStop*
<EJ.DropDownList
popupResizeStop="popupResizeStop"></EJ.DropDownList> | **Not applicable** |

| **Popup shown event** | **Event:** *popupShown*
<EJ.DropDownList
popupShown="popupShown"></EJ.DropDownList> | **Event:** *open*

<DropDownListComponent id="ddl" ref={(scope) => { this.ddlObj = scope; }}
(open)={this.open.bind(this)} />|

Placeholder

<!-- markdownlint-disable MD010 -->

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

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

| **Watermark text** | **Property:** *watermarkText*
<EJ.DropDownList
watermarkText="select"></EJ.DropDownList> | **Property:** *placeholder*
var
dropDownListObject = new ej.dropdowns.DropDownList({placeholder:
"select"});dropDownListObject.appendTo('#ddlelement');|

| **Floating of watermark text** | **Not applicable** | **Property:** *floatLabelType*
var
dropDownListObject = new ej.dropdowns.DropDownList({floatLabelType:
"Auto"});dropDownListObject.appendTo('#ddlelement');|

Grouping

<!-- markdownlint-disable MD010 -->

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

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

| **Default** | **Property:** *fields.groupBy*
<EJ.DropDownList fields-value="parentId" fields-
groupBy="text" ></EJ.DropDownList> | **Property:** *fields.groupBy*
<DropDownListComponent
id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} fields={this.field} /> |

| **Group Template** | **Not applicable** | **Property:** *groupTemplate*
<DropDownListComponent
id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} groupTemplate={this.groupTemplate} /> |

Accessibility

<!-- markdownlint-disable MD010 -->

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

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

| **Globalization** | **Property:** *locale*
<EJ.DropDownList fields-value="parentId" locale="fr-FE"
></EJ.DropDownList> | **Property:** *locale*
<DropDownListComponent id="ddl" ref={{(scope) => {
this.ddlObj = scope; }}} locale={this.locale} /> |

| **Rtl support** | **Property:** *enableRTL*
<EJ.DropDownList fields-value="parentId"
enableRTL={true} ></EJ.DropDownList> | **Property:** *enableRtl*
<DropDownListComponent
id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} enableRtl={true} /> |

Miscellaneous

<!-- markdownlint-disable MD010 -->

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

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

| **Enable/disable** | **Property:** *enabled*
<EJ.DropDownList enabled={true} ></EJ.DropDownList>
| **Property:** *enabled*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj =
scope; }}} enabled={true} /> |

| **Read only** | **Property:** *readOnly*
<EJ.DropDownList readOnly={true} ></EJ.DropDownList> |

Property: *readOnly*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj =
scope; }}} readOnly={true} /> |

| Persistence of data | **Property:** `enablePersistence`
 <EJ.DropDownList enablePersistence={true} ></EJ.DropDownList> | **Property:** `enablePersistence`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} enablePersistence={true} /> |

| **Disable** | **Method:** `disable`
 <EJ.DropDownList></EJ.DropDownList>
 \$('#dropdown').ejDropDownList('disable') | **Property:** `enabled`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} enabled={true} /> |

| **Enable** | **Method:** `enable`
 <EJ.DropDownList></EJ.DropDownList>
 \$('#dropdown').ejDropDownList('enable') | **Property:** `enabled`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} enabled={true} /> |

| **Height** | **Property:** `height`
 <EJ.DropDownList height="500px" ></EJ.DropDownList> | **Property:** `height`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} height={this.height} /> |

| **Width** | **Property:** `width`
 <EJ.DropDownList width="100px" ></EJ.DropDownList> | **Property:** `width`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} width={this.width} /> |

Selection

<!-- markdownlint-disable MD010 -->

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

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

| Selecting particular index | **Property:** `selectedIndex`
 <EJ.DropDownList selectedIndex={selectedIndex} ></EJ.DropDownList> | **Property:** `index`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} index={this.index} /> |

| Selecting particular value | **Property:** `value`
 <EJ.DropDownList value="data" ></EJ.DropDownList> | **Property:** `value`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} value={this.value} /> |

| Selecting particular text | **Property:** `text`
 <EJ.DropDownList text="data" ></EJ.DropDownList> | **Property:** `text`
 <DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} text={this.text} /> |

| Target id | **Property:** `targetId`
 <EJ.DropDownList targetId="id" ></EJ.DropDownList> | **Not applicable** |

| Selecting item using text | **Method:** `selectItemByText`
 <EJ.DropDownList ></EJ.DropDownList>
 \$('#dropdown').ejDropDownList('selectItemByText', 'car') | **Not applicable** |

| Unselect item using text | **Method:** `unselectItemByText`
 <EJ.DropDownList></EJ.DropDownList>
 \$('#dropdown').ejDropDownList('unselectItemByText', 'car') | **Not applicable** |

| **Selecting item using value** | **Method:** *selectItemByValue*
<EJ.DropDownList>
></EJ.DropDownList>
\$('#dropdown').ejDropDownList('selectItemByValue', 'car') | **Not applicable** |

| **Getting data by using value** | **Method:** *getItemDataByValue*
<EJ.DropDownList>
></EJ.DropDownList>
\$('#dropdown').ejDropDownList('unselectItemByValue', 'car') |
Method: *getDataByValue*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}}>
this.ddlObj.getDataByValue(); |

| **Get selected value** | **Method:** *getSelectedItem*
<EJ.DropDownList>
></EJ.DropDownList>
\$('#dropdown').ejDropDownList('getSelectedItem') | **Not applicable** |

| **Get selected text** | **Method:** *getSelectedText*

<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('getSelectedText') | **Property:** *text*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} text={this.text} /> |

| **Select event** | **Event:** *select*
<EJ.DropDownList select="onSelect" ></EJ.DropDownList> |
Event: *select*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}} (select)={this.select.bind(this)} /> |

| **Addition of Html attributes** | **Property:** *htmlAttributes*
<EJ.DropDownList
htmlAttributes={attributes} ></EJ.DropDownList> | **Property:**
htmlAttributes
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}}
htmlAttributes={this.htmlAttributes} /> |

Common

<!-- markdownlint-disable MD010 -->

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

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

| **Adding new item** | **Method :**

addItem
<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList("add
tem", {text:"India"}); | **Method:** *addItem*
<DropDownListComponent id="ddl" ref={{(scope)
=> { this.ddlObj = scope; }}}>
this.ddlObj.addItem("data"); |

| **Clearing the text** | **Method :**

clearText
<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('clea
rText') | **Property:** *value*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj =
scope; }}} value={this.value} /> |

| **Destroy the component** | **Method :**

destroy
<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('destr
oy') | **Method:** *destroy*
<DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj =
scope; }}}>
this.ddlObj.destroy(); |

| **Getting the data** | **Method :**

getListData
<EJ.DropDownList></EJ.DropDownList>
\$('#dropdown').ejDropDownList('g
etListData') | **Method :** *getItems*
<DropDownListComponent id="ddl" ref={{(scope) => {
this.ddlObj = scope; }}}>
this.ddlObj.getItems(); |

```
| Create event | Event: create<br/><EJ.DropDownList create="create" ></EJ.DropDownList> |
Event: created<br/><DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
(created)={this.created.bind(this)}} /> |

| Destroy event | Event: destroy<br/><EJ.DropDownList destroy="destroy"
></EJ.DropDownList> | Event: destroyed<br/><DropDownListComponent id="ddl" ref={{(scope) =>
{ this.ddlObj = scope; }} (destroyed)={this.destroyed.bind(this)}} /> |

| Cascade event | Event: cascade<br/><EJ.DropDownList cascade="cascade"
></EJ.DropDownList> | https://ej2.syncfusion.com/react/demos/#/material/drop-down-list/cascading
|

| Change event | Event: change<br/><EJ.DropDownList change="change" ></EJ.DropDownList> |
Event: change<br/><DropDownListComponent id="ddl" ref={{(scope) => { this.ddlObj = scope; }}
(change)={this.change.bind(this)}} /> |

| Focus out event | Event: focusOut<br/><EJ.DropDownList focusOut="focusOut"
></EJ.DropDownList> | Event: blur<br/><DropDownListComponent id="ddl" ref={{(scope) => {
this.ddlObj = scope; }} (blur)={this.blur.bind(this)}} /> |

| Focus in event | Event: focusIn<br/><br/><EJ.DropDownList focusIn="focusIn"
></EJ.DropDownList> | Event: focus<br/><DropDownListComponent id="ddl" ref={{(scope) => {
this.ddlObj = scope; }} (focus)={this.focus.bind(this)}} /> |

{% endraw %}
```

Dropdown Tree

Getting Started

This section explains you about how to create a simple **Dropdown Tree** component and configure its available functionalities in React.

Dependencies

The following list of dependencies are required to use the **Dropdown Tree** component in your application.

```
`javascript
|-- @syncfusion/ej2-react-dropdowns
|-- @syncfusion/ej2-base
|-- @syncfusion/ej2-data
|-- @syncfusion/ej2-react-base
|-- @syncfusion/ej2-dropdowns
|-- @syncfusion/ej2-lists
|-- @syncfusion/ej2-inputs
|-- @syncfusion/ej2-navigations
|-- @syncfusion/ej2-popups
```



```
|-- @syncfusion/ej2-buttons  
,
```

Installation and configuration

You can use [Create-react-app](#) to setup the applications. To install `create-react-app` run the following command.

```
`bash  
npm install -g create-react-app  
,
```

To set-up a React application in TypeScript environment, run the following command.

```
`bash  
npx create-react-app my-app --template typescript  
cd my-app  
npm start  
,
```

To set-up a React application in JavaScript environment, run the following command.

```
`bash  
npx create-react-app my-app  
cd my-app  
npm start  
,
```

Adding syncfusion packages

All the available Essential JS 2 packages are published in [npmjs.com](https://www.npmjs.com) public registry. You can choose the component that you want to install.

To install Dropdown Tree component, use the following command

```
`bash  
npm install @syncfusion/ej2-react-dropdowns --save  
,
```

Adding Dropdown Tree component

Now, you can start adding Dropdown Tree component in the application. For getting started, add the Dropdown Tree component in `src/App.tsx` file using following code. Add the below code in the `src/App.tsx` to initialize the Dropdown Tree.

```
`ts  
  
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';  
import * as React from 'react';  
import * as ReactDOM from "react-dom";
```

```
function App() {
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id='dropdowntree' />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
`ts
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from "react-dom";
function App() {
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id='dropdowntree' />);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`
```

Adding CSS reference

Import the Dropdown Tree component required CSS references as follows in `src/App.css`.

```
`css
/ import the Dropdown Tree dependency styles /
@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-navigations/styles/material.css";
@import "../node_modules/@syncfusion/ej2-react-dropdowns/styles/material.css";
`
```

Binding data source

The Dropdown Tree component can load the data either from local data sources or remote data services. This can be done using the `dataSource` property that is a member of the `fields` property. The

dataSource property supports array of JavaScript objects and DataManager. Here, an array of JSON values is passed to the Dropdown Tree component.

```
`ts
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data: { [key: string]: Object }[] = [
    {
      nodeId: '01', nodeText: 'Music',
      nodeChild: [
        { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
      ]
    },
    {
      nodeId: '02', nodeText: 'Videos', expanded: true,
      nodeChild: [
        { nodeId: '02-01', nodeText: 'Naturals.mp4' },
        { nodeId: '02-02', nodeText: 'Wild.mpeg' },
      ]
    },
    {
      nodeId: '03', nodeText: 'Documents',
      nodeChild: [
        { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
        { nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
        { nodeId: '03-03', nodeText: 'Global Warming.ppt' },
        { nodeId: '03-04', nodeText: 'Social Network.pdf' },
        { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
      ]
    },
  ];
};
```

```
let fields: Object = { dataSource: data, value: 'nodeId', text: 'nodeText', child: 'nodeChild' };
return (
  // specifies the tag for render the DropDownTree component
  <DropDownTreeComponent id="dropdownntree" fields={fields} />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
`ts
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // definining the dataSource
  let data = [
    {
      nodeId: '01', nodeText: 'Music',
      nodeChild: [
        { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
      ]
    },
    {
      nodeId: '02', nodeText: 'Videos', expanded: true,
      nodeChild: [
        { nodeId: '02-01', nodeText: 'Naturals.mp4' },
        { nodeId: '02-02', nodeText: 'Wild.mpeg' },
      ]
    },
    {
      nodeId: '03', nodeText: 'Documents',
      nodeChild: [
        { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
```

```

{ nodeId: '03-02', nodeText: 'Global Water, Sanitation, & Hygiene.docx' },
{ nodeId: '03-03', nodeText: 'Global Warming.ppt' },
{ nodeId: '03-04', nodeText: 'Social Network.pdf' },
{ nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
]
},
];

let fields = { dataSource: data, value: 'nodeId', text: 'nodeText', child: 'nodeChild' };
return (
  // specifies the tag for render the DropDownTree component
  <DropDownTreeComponent id="dropdownntree" fields={fields}/>;
)
}

export default App;

ReactDOM.render(<App />, document.getElementById('sample'));
`

```

Run the application

After completing the configuration required to render a basic Dropdown Tree, run the following command to display the output in your default browser.

```
npm start
```

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // definining the dataSource
  let data = [
    {
      nodeId: '01', nodeText: 'Music',
      nodeChild: [
        { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
      ]
    },
    {
      nodeId: '02', nodeText: 'Videos', expanded: true,
      nodeChild: [
        { nodeId: '02-01', nodeText: 'Naturals.mp4' },
        { nodeId: '02-02', nodeText: 'Wild.mpeg' },
      ]
    },
  ],
  {

```

```

        nodeId: '03', nodeText: 'Documents',
        nodeChild: [
            { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
            { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
Hygiene.docx' },
            { nodeId: '03-03', nodeText: 'Global Warming.ppt' },
            { nodeId: '03-04', nodeText: 'Social Network.pdf' },
            { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
        ]
    },
];
let fields = { dataSource: data, value: 'nodeId', text: 'nodeText',
child: 'nodeChild' };
return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // definining the dataSource
    let data: { [key: string]: Object }[] = [
        {
            nodeId: '01', nodeText: 'Music',
            nodeChild: [
                { nodeId: '01-01', nodeText: 'Gouttes.mp3' }
            ]
        },
        {
            nodeId: '02', nodeText: 'Videos', expanded: true,
            nodeChild: [
                { nodeId: '02-01', nodeText: 'Naturals.mp4' },
                { nodeId: '02-02', nodeText: 'Wild.mpeg' },
            ]
        },
        {
            nodeId: '03', nodeText: 'Documents',
            nodeChild: [
                { nodeId: '03-01', nodeText: 'Environment Pollution.docx' },
                { nodeId: '03-02', nodeText: 'Global Water, Sanitation, &
Hygiene.docx' },
                { nodeId: '03-03', nodeText: 'Global Warming.ppt' },
                { nodeId: '03-04', nodeText: 'Social Network.pdf' },
                { nodeId: '03-05', nodeText: 'Youth Empowerment.pdf' },
            ]
        },
    ];
    let fields: Object = { dataSource: data, value: 'nodeId', text: 'nodeText',
child: 'nodeChild' };
    return (

```

```
// specifies the tag for render the DropDownTree component
<DropDownTreeComponent id="dropdownntree" fields={fields} />
);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Data binding in React Drop down tree component

The Dropdown Tree component provides an option to load the data either from local data sources or from remote data services. This can be done through **dataSource** property that is a member of the **fields** property. The **dataSource** property supports array of JavaScript objects and **DataManager**. It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

Dropdown Tree has **load on demand** (Lazy load) option. It reduces the bandwidth size when consuming the huge data. By default, the **loadOnDemand** is set to false. By enabling this property, it loads first level items initially, and when parent item is expanded, loads the child items based on the **parentValue/child** member.

Local data

To bind local data to the Dropdown Tree, you can assign a JavaScript object array to the **dataSource** property.

The Dropdown Tree component requires three fields (Value, text, and parentValue) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the **DataManager**. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

Hierarchical data

Dropdown Tree can be populated with the hierarchical data source that contains nested array of JSON objects. You can directly map the hierarchical data and the field members with corresponding key values from the hierarchical data to the **fields** property.

In the following example, **code**, **name**, and **countries** columns from the hierarchical data have been mapped to **value**, **text**, and **child** fields, respectively.

INDEX.JSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // definining the dataSource
  let data = [
    {
      code: 'AF', name: 'Africa', countries: [
        { code: 'NGA', name: 'Nigeria' },
        { code: 'EGY', name: 'Egypt' },
        { code: 'ZAF', name: 'South Africa' }
      ]
    }
  ]
}
```

```

    ],
    {
      code: 'AS', name: 'Asia', expanded: true, countries: [
        { code: 'CHN', name: 'China' },
        { code: 'IND', name: 'India', selected: true },
        { code: 'JPN', name: 'Japan' }
      ]
    },
    {
      code: 'EU', name: 'Europe', countries: [
        { code: 'DNK', name: 'Denmark' },
        { code: 'FIN', name: 'Finland' },
        { code: 'AUT', name: 'Austria' }
      ]
    },
    {
      code: 'NA', name: 'North America', countries: [
        { code: 'USA', name: 'United States of America' },
        { code: 'CUB', name: 'Cuba' },
        { code: 'MEX', name: 'Mexico' }
      ]
    },
    {
      code: 'SA', name: 'South America', countries: [
        { code: 'BRA', name: 'Brazil' },
        { code: 'COL', name: 'Colombia' },
        { code: 'ARG', name: 'Argentina' }
      ]
    },
    {
      code: 'OC', name: 'Oceania', countries: [
        { code: 'AUS', name: 'Australia' },
        { code: 'NZL', name: 'New Zealand' },
        { code: 'WSM', name: 'Samoa' }
      ]
    },
    {
      code: 'AN', name: 'Antarctica', countries: [
        { code: 'BVT', name: 'Bouvet Island' },
        { code: 'ATF', name: 'French Southern Lands' }
      ]
    },
  ],
];
let fields = { dataSource: data, value: 'code', text: 'name', child:
'countries' };
return (
  // specifies the tag for render the DropDownTree component
  <DropDownTreeComponent id="dropdownntree" fields={fields}/>
)
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
```



```

import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data: { [key: string]: Object; }[] = [
    {
      code: 'AF', name: 'Africa', countries: [
        { code: 'NGA', name: 'Nigeria' },
        { code: 'EGY', name: 'Egypt' },
        { code: 'ZAF', name: 'South Africa' }
      ]
    },
    {
      code: 'AS', name: 'Asia', expanded: true, countries: [
        { code: 'CHN', name: 'China' },
        { code: 'IND', name: 'India', selected: true },
        { code: 'JPN', name: 'Japan' }
      ]
    },
    {
      code: 'EU', name: 'Europe', countries: [
        { code: 'DNK', name: 'Denmark' },
        { code: 'FIN', name: 'Finland' },
        { code: 'AUT', name: 'Austria' }
      ]
    },
    {
      code: 'NA', name: 'North America', countries: [
        { code: 'USA', name: 'United States of America' },
        { code: 'CUB', name: 'Cuba' },
        { code: 'MEX', name: 'Mexico' }
      ]
    },
    {
      code: 'SA', name: 'South America', countries: [
        { code: 'BRA', name: 'Brazil' },
        { code: 'COL', name: 'Colombia' },
        { code: 'ARG', name: 'Argentina' }
      ]
    },
    {
      code: 'OC', name: 'Oceania', countries: [
        { code: 'AUS', name: 'Australia' },
        { code: 'NZL', name: 'New Zealand' },
        { code: 'WSM', name: 'Samoa' }
      ]
    },
    {
      code: 'AN', name: 'Antarctica', countries: [
        { code: 'BVT', name: 'Bouvet Island' },
        { code: 'ATF', name: 'French Southern Lands' }
      ]
    }
  ];
  let fields: Object = { dataSource: data, value: 'code', text: 'name',
    child: 'countries' };
  return (

```

```

        // specifies the tag for render the DropDownTree component
        <DropDownTreeComponent id="dropdownntree" fields={fields} />
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Self-referential data

Dropdown Tree can be populated from the self-referential data structure that contains array of JSON objects with **parentValue** mapping.

You can directly assign the self-referential data and map all the field members with corresponding key values from self-referential data to the **fields** property.

To render the root level items, specify the **parentValue** as null or no need to specify the **parentValue** in the **dataSource**.

In the following example, **id**, **pid**, **hasChild**, and **name** columns from self-referential data have been mapped to **value**, **parentValue**, **hasChildren**, and **text** fields, respectively.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // definining the dataSource
    let data = [
        { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Hot Singles' },
        { id: 3, pid: 1, name: 'Rising Artists' },
        { id: 4, pid: 1, name: 'Live Music' },
        { id: 6, pid: 1, name: 'Best of 2017 So Far' },
        { id: 7, name: 'Sales and Events', hasChild: true },
        { id: 8, pid: 7, name: '100 Albums' },
        { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
        { id: 10, pid: 7, name: 'CD Deals' },
        { id: 11, name: 'Categories', hasChild: true },
        { id: 12, pid: 11, name: 'Songs' },
        { id: 13, pid: 11, name: 'Bestselling Albums' },
        { id: 14, pid: 11, name: 'New Releases' },
        { id: 15, pid: 11, name: 'Bestselling Songs' },
        { id: 16, name: 'MP3 Albums', hasChild: true },
        { id: 17, pid: 16, name: 'Rock' },
        { id: 18, pid: 16, name: 'Gospel' },
        { id: 19, pid: 16, name: 'Latin Music' },
        { id: 20, pid: 16, name: 'Jazz' },
        { id: 21, name: 'More in Music', hasChild: true },
        { id: 22, pid: 21, name: 'Music Trade-In' },
        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
    "pid", hasChildren: 'hasChild' };
    return (
        // specifies the tag for render the DropDownTree component

```

```

    <DropDownTreeComponent id="dropdownntree" fields={fields}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data: { [key: string]: Object }[] = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },
    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },
    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  let fields: Object = { dataSource: data, value: 'id', text: 'name',
    parentValue: "pid", hasChildren: 'hasChild' };
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Remote data

Dropdown Tree can also be populated from a remote data service with the help of the **DataManager** component and **Query** property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

You can assign service data as an instance of **DataManager** to the **dataSource**. To interact with remote data source, you must provide the endpoint **url**.

The **DataManager** that acts as an interface between the service endpoint and the Dropdown Tree requires the following information to interact with service endpoint properly.

- **DataManager->url**: Defines the service endpoint to fetch data.
- **DataManager->adaptor**: Defines the adaptor option. By default, **ODataAdaptor** is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The **@syncfusion/ej2-data** package provides some pre-defined adaptors designed to interact with service endpoints. They are,

- **UrlAdaptor**: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from the remote services. The **EmployeeID**, **FirstName**, and **EmployeeID**

columns from the Employees table have been mapped to **value**, **text**, and **hasChildren** fields respectively for first level items.

The **OrderID**, **EmployeeID**, and **ShipName** columns from the orders table have been mapped to **value**, **parentValue**, and **text** fields respectively for second level items.

INDEX.JSX

```
import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
  });
  let query = new
  Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
  let query1 = new
  Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
  let fields = {
    dataSource: data, query: query, value: 'EmployeeID', text:
    'FirstName', hasChildren: 'EmployeeID',
    child: { dataSource: data, query: query1, value: 'OrderID',
    parentValue: 'EmployeeID', text: 'ShipName' }
  };
};
```

```

    return (
      // specifies the tag for render the DropDownTree component
      <DropDownTreeComponent id="dropdownntree" fields={fields}/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svc',
    adaptor: new ODataV4Adaptor,
    crossDomain: true,
  });
  let query: Query = new
  Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
  let query1: Query = new
  Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
  let fields: object = {
    dataSource: data, query: query, value: 'EmployeeID', text: 'FirstName',
    hasChildren: 'EmployeeID',
    child: { dataSource: data, query: query1, value: 'OrderID', parentValue:
    'EmployeeID', text: 'ShipName' }
  }
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Prevent Node selection

You can prevent the selection of individual tree node by using the [selectable](#) property. The tree node selection is not allowed while disable this property.

The `selectable` property is disabled and the selection is prevented for parent nodes in below sample.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // definining the dataSource
  let data = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true,
    selectable: false },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
  ]

```

```

    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true, selectable: false
  },

  { id: 8, pid: 7, name: '100 Albums' },
  { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  { id: 10, pid: 7, name: 'CD Deals' },
  { id: 11, name: 'Categories', hasChild: true, selectable: false },
  { id: 12, pid: 11, name: 'Songs' },
  { id: 13, pid: 11, name: 'Bestselling Albums' },
  { id: 14, pid: 11, name: 'New Releases' },
  { id: 15, pid: 11, name: 'Bestselling Songs' },
  { id: 16, name: 'MP3 Albums', hasChild: true, selectable: false },
  { id: 17, pid: 16, name: 'Rock' },
  { id: 18, pid: 16, name: 'Gospel' },
  { id: 19, pid: 16, name: 'Latin Music' },
  { id: 20, pid: 16, name: 'Jazz' },
  { id: 21, name: 'More in Music', hasChild: true, selectable: false },
  { id: 22, pid: 21, name: 'Music Trade-In' },
  { id: 23, pid: 21, name: 'Redeem a Gift Card' },
  { id: 24, pid: 21, name: 'Band T-Shirts' },
];
let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
"pid", hasChildren: 'hasChild', selectable: 'selectable' };
return (
  // specifies the tag for render the DropDownTree component
  <DropDownTreeComponent id="dropdownntree" fields={fields}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // definining the dataSource
  let data: { [key: string]: Object }[] = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true,
selectable: false },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true, selectable: false
  },

  { id: 8, pid: 7, name: '100 Albums' },
  { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  { id: 10, pid: 7, name: 'CD Deals' },
  { id: 11, name: 'Categories', hasChild: true, selectable: false },
  { id: 12, pid: 11, name: 'Songs' },
  { id: 13, pid: 11, name: 'Bestselling Albums' },
  { id: 14, pid: 11, name: 'New Releases' },
  { id: 15, pid: 11, name: 'Bestselling Songs' },
  { id: 16, name: 'MP3 Albums', hasChild: true, selectable: false },
  { id: 17, pid: 16, name: 'Rock' },

```

```

    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true, selectable: false },
    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  let fields: Object = { dataSource: data, value: 'id', text: 'name',
    parentValue: "pid", hasChildren: 'hasChild', selectable: 'selectable' };
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields} />
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Templates in React Drop down tree component

The Dropdown Tree provides support to customize each list item, header, and footer elements.

Item template

The content of each list item within the Dropdown Tree can be customized with the help of [itemTemplate](#) property.

In the following sample, the Dropdown Tree list items are customized with employee information such as **name** and **job** using the **itemTemplate** property.

The template expression should be provided inside the {...} interpolation syntax.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
      "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product
Manager", "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
      "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product
Manager", "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer" },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer" },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
    "pid", hasChildren: 'hasChild' };

```

```
function itemTemplate(data) {
    return (<div><span className="ename"> {data.name} - </span><span
className="ejjob"> {data.job} </span></div>);
}
return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" itemTemplate={itemTemplate} popupHeight="270px" cssClass="custom"
width="100%"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    let data: { [key: string]: Object }[] = [
        { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
"hasChild": true, "expanded": true },
        { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
"hasChild": true },
        { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
"hasChild": true },
        { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
        { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer",
"status": "online" },
        { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
"hasChild": true },
        { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
"hasChild": true },
        { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer" },
        { "id": 11, "pid": 6, "name": "Mary", "job": "Developer" },
        { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
    ];
    let fields: Object = { dataSource: data, value: 'id', text: 'name',
parentValue: "pid", hasChildren: 'hasChild' };
    function itemTemplate(data: any): JSX.Element {
        return (<div><span className="ename"> {data.name} - </span><span
className="ejjob"> {data.job} </span></div>);
    }
    return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" itemTemplate={itemTemplate} popupHeight="270px" cssClass="custom"
width="100%"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Header template

The header element is shown statically at the top of the popup list items within the Dropdown Tree. A custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the header is customized with the custom element.

INDEX.JSX


```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    let data = [
        { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
        "hasChild": true, "expanded": true },
        { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product
Manager", "hasChild": true },
        { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
        "hasChild": true },
        { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
        { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
        "online" },
        { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product
Manager", "hasChild": true },
        { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
        "hasChild": true },
        { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
        { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
        { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
    ];
    let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
    "pid", hasChildren: 'hasChild' };
    function headerTemplate(data) {
        return (<div className="head"> Employee List </div>);
    }
    return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" headerTemplate={headerTemplate} popupHeight="270px"
cssClass="custom"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    let data: { [key: string]: Object }[] = [
        { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
        "hasChild": true, "expanded": true },
        { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
        "hasChild": true },
        { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
        "hasChild": true },
        { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
        { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer",
        "status": "online" },
        { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
        "hasChild": true },
        { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
        "hasChild": true },
        { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
        { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    ];

```

```

    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields: Object = { dataSource: data, value: 'id', text: 'name',
  parentValue: "pid", hasChildren: 'hasChild' };
  function headerTemplate(data: any): JSX.Element {
    return (<div className="head"> Employee List </div>);
  }
  return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" headerTemplate={headerTemplate} popupHeight="270px"
cssClass="custom"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Footer template

The Dropdown Tree has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, the footer element displays the total number of employees present in the Dropdown Tree.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product
Manager", "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
    "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product
Manager", "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
  "pid", hasChildren: 'hasChild' };
  function footerTemplate(data) {
    return (<div class='foot'> Total number of employees: 10 </div>);
  }
  return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" footerTemplate={footerTemplate} popupHeight="270px"
cssClass="custom"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data: { [key: string]: Object }[] = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
    "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer",
    "status": "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
    "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer" },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer" },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields: object = { dataSource: data, value: 'id', text: 'name',
  parentValue: 'pid', hasChildren: 'hasChild' };
  function footerTemplate(data: any): JSX.Element {
    return (<div class='foot'> Total number of employees: 10 </div>);
  }
  return (<DropDownTreeComponent fields={fields} placeholder="Select an
  employee" footerTemplate={footerTemplate} popupHeight="270px"
  cssClass="custom"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

No records template

The Dropdown Tree is supports to display custom design in the popup list content using the [noRecordsTemplate](#) property when no matches found on search.

In the following sample, popup list content displays the notification of no data available.

INDEX.JSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data = [];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
  "pid", hasChildren: 'hasChild' };
  function noRecordsTemplate(data) {
    return (<div class='norecord'> NO DATA AVAILABLE </div>);
  }
}
```

```

    return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" noRecordsTemplate={noRecordsTemplate} popupHeight="270px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    let data: { [key: string]: Object }[] = [ ];
    let fields: object = { dataSource: data, value: 'id', text: 'name',
parentValue: "pid", hasChildren: 'hasChild' };
    function noRecordsTemplate(data: any): JSX.Element {
        return (<div class='norecord'> NO DATA AVAILABLE </div>);
    }
    return (<DropDownTreeComponent fields={fields} placeholder="Select an
employee" noRecordsTemplate={noRecordsTemplate} popupHeight="270px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Action failure template

The Dropdown Tree provides an option to custom design the popup list content using [actionFailureTemplate](#) property, when the data fetch request fails at the remote server.

In the following sample, when the data fetch request fails, the Dropdown Tree displays the notification.

INDEX.JSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    let data = new DataManager({
        url: 'https://services.odata.org/V4/Northwind/Northwind.svcs',
        adaptor: new ODataV4Adaptor,
        crossDomain: true,
    });
    let query = new
Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
    let query1 = new
Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
    let fields = {
        dataSource: data, query: query, value: 'EmployeeID', text:
'FirstName', hasChildren: 'EmployeeID',
        child: { dataSource: data, query: query1, value: 'OrderID',
parentValue: 'EmployeeID', text: 'ShipName' }
    };
    function actionFailureTemplate(data) {
        return (<div class='action-failure'> Data fetch request fails
</div>);
    }
}

```

```

    return (<DropDownTreeComponent fields={fields}
    actionFailureTemplate={actionFailureTemplate} placeholder="Select an
    employee" popupHeight="200px"/>);
  }
  export default App;
  ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DataManager, ODataV4Adaptor, Query } from '@syncfusion/ej2-data';
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data: DataManager = new DataManager({
    url: 'https://services.odata.org/V4/Northwind/Northwind.svcs',
    adaptor: new ODataV4Adaptor,
    crossDomain: true, });
  let query: Query = new
  Query().from('Employees').select('EmployeeID,FirstName,Title').take(5);
  let query1: Query = new
  Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5);
  let fields: object = {
    dataSource: data, query: query, value: 'EmployeeID', text: 'FirstName',
    hasChildren: 'EmployeeID',
    child: { dataSource: data, query: query1, value: 'OrderID', parentValue:
    'EmployeeID', text: 'ShipName' }
  };
  function actionFailureTemplate (data: any): JSX.Element {
    return (<div class='action-failure'> Data fetch request fails </div>);
  }
  return (<DropDownTreeComponent fields={fields}
  actionFailureTemplate={actionFailureTemplate} placeholder="Select an
  employee" popupHeight="200px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Custom template to show selected items in input

In Dropdown Tree, while selecting more than one items via checkbox or multi selection support, all the selected items will be displayed in the input. Instead of displaying all the selected item text, the custom template can be displayed by setting the the [mode](#) property as **Custom** and [customTemplate](#) property.

When the **mode** property is set as **Custom**, the Dropdown Tree displays the default template value (**\${value.length} item(s) selected**) like **1 item(s) selected** or **2 item(s) selected**. The default template can be customized by setting **customTemplate** property.

In the following sample, the Dropdown Tree is rendered with default value of the **customTemplate** property like **"1 item(s) selected or 2 item(s) selected"**.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';

```

```
function App() {
  let data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
      "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product
Manager", "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
      "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product
Manager", "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
    "pid", hasChildren: 'hasChild' };
  let showCheckBox = true;
  let treeSettings = { autoCheck: true };
  let customTemplate = `${value.length} item(s) selected`;
  return (<DropDownTreeComponent fields={fields}
treeSettings={treeSettings} customTemplate={customTemplate}
showCheckBox={showCheckBox} mode="Custom" placeholder="Select an employee"
popupHeight="200px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data: { [key: string]: Object }[] = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
      "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
      "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer",
      "status": "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
      "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
```

```

    let fields: object = { dataSource: data, value: 'id', text: 'name',
    parentValue: "pid", hasChildren: 'hasChild' };
    let showCheckBox: boolean = true;
    let treeSettings: Object = { autoCheck: true };
    let customTemplate: string = "${value.length} item(s) selected";
    return (<DropDownTreeComponent fields={fields} treeSettings={treeSettings}
    customTemplate={customTemplate} showCheckBox={showCheckBox} mode="Custom"
    placeholder="Select an employee" popupHeight="200px"/>);
  }
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

In the following sample, the Dropdown Tree is rendered with custom value of the **customTemplate** property like **Selected items count: 2**.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  let data = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
    "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product
Manager", "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
    "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer", "status":
    "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product
Manager", "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
    "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
  "pid", hasChildren: 'hasChild' };
  let showCheckBox = true;
  let treeSettings = { autoCheck: true };
  let customTemplate = "Selected items count: ${value.length} item(s) ";
  return (<DropDownTreeComponent fields={fields}
  treeSettings={treeSettings} customTemplate={customTemplate}
  showCheckBox={showCheckBox} mode="Custom" placeholder="Select an employee"
  popupHeight="200px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';

```

```
import * as ReactDOM from 'react-dom';
function App() {
  let data: { [key: string]: Object }[] = [
    { "id": 1, "name": "Steven Buchanan", "job": "General Manager",
      "hasChild": true, "expanded": true },
    { "id": 2, "pid": 1, "name": "Laura Callahan", "job": "Product Manager",
      "hasChild": true },
    { "id": 3, "pid": 2, "name": "Andrew Fuller", "job": "Team Lead",
      "hasChild": true },
    { "id": 4, "pid": 3, "name": "Anne Dodsworth", "job": "Developer" },
    { "id": 10, "pid": 3, "name": "Lilly", "job": "Developer",
      "status": "online" },
    { "id": 5, "pid": 1, "name": "Nancy Davolio", "job": "Product Manager",
      "hasChild": true },
    { "id": 6, "pid": 5, "name": "Michael Suyama", "job": "Team Lead",
      "hasChild": true },
    { "id": 7, "pid": 6, "name": "Robert King", "job": "Developer " },
    { "id": 11, "pid": 6, "name": "Mary", "job": "Developer " },
    { "id": 9, "pid": 1, "name": "Janet Leverling", "job": "HR" }
  ];
  let fields: object = { dataSource: data, value: 'id', text: 'name',
    parentValue: "pid", hasChildren: 'hasChild' };
  let showCheckBox: boolean = true;
  let treeSettings: Object = { autoCheck: true };
  let customTemplate: string = "Selected items count: ${value.length} item(s)
";
  return (<DropDownTreeComponent fields={fields} treeSettings={treeSettings}
    customTemplate={customTemplate} showCheckBox={showCheckBox} mode="Custom"
    placeholder="Select an employee" popupHeight="200px"/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

Checkbox in React Drop down tree component

The Dropdown Tree component allows you to check more than one item from the tree without affecting the UI's appearance by enabling the `showCheckBox` property. When this property is enabled, checkbox appears before each item text in the popup.

In the following example, the `showCheckBox` property is enabled.

INDEX.JSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums - $5 Each' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
  ];
```



```

    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },
    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },
    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
  "pid", hasChildren: 'hasChild' };
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields}
    showCheckBox={true}/>);
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data: { [key: string]: Object }[] = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 5, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums - $5 Each' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },
    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },
    { id: 22, pid: 21, name: 'Music Trade-In' },
    { id: 23, pid: 21, name: 'Redeem a Gift Card' },
  ];
}

```

```

    { id: 24, pid: 21, name: 'Band T-Shirts' },
  ];
  let fields: Object = { dataSource: data, value: 'id', text: 'name',
  parentValue: "pid", hasChildren: 'hasChild' };
  return (
    // specifies the tag for render the DropDownTree component
    <DropDownTreeComponent id="dropdownntree" fields={fields}
    showCheckBox={true}/>
  );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Auto Check

By default, the checkbox state of the parent and child items in the Dropdown Tree will not be dependent over each other. If you need dependent checked state, then enable the `autoCheck` property which is a member of `treeSettings` property.

- If one or more child items are not in the checked state, then the parent item will be in the intermediate state.
- If all the child items are checked, then the parent item will also be in the checked state.
- If a parent item is checked, then all the child items will also be changed to the checked state.

In the following example, the `autoCheck` property is enabled.

INDEX.JSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
  // defining the dataSource
  let data = [
    { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
    { id: 2, pid: 1, name: 'Hot Singles' },
    { id: 3, pid: 1, name: 'Rising Artists' },
    { id: 4, pid: 1, name: 'Live Music' },
    { id: 6, pid: 1, name: 'Best of 2017 So Far' },
    { id: 7, name: 'Sales and Events', hasChild: true },
    { id: 8, pid: 7, name: '100 Albums - $5 Each' },
    { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
    { id: 10, pid: 7, name: 'CD Deals' },
    { id: 11, name: 'Categories', hasChild: true },
    { id: 12, pid: 11, name: 'Songs' },
    { id: 13, pid: 11, name: 'Bestselling Albums' },
    { id: 14, pid: 11, name: 'New Releases' },
    { id: 15, pid: 11, name: 'Bestselling Songs' },
    { id: 16, name: 'MP3 Albums', hasChild: true },
    { id: 17, pid: 16, name: 'Rock' },
    { id: 18, pid: 16, name: 'Gospel' },
    { id: 19, pid: 16, name: 'Latin Music' },
    { id: 20, pid: 16, name: 'Jazz' },
    { id: 21, name: 'More in Music', hasChild: true },
    { id: 22, pid: 21, name: 'Music Trade-In' },
  ];
}

```

```

        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
"pid", hasChildren: 'hasChild' };
    let treeSettings = { autoCheck: true };
    return (
        // specifies the tag for render the DropDownTree component
        <DropDownTreeComponent id="dropdownntree" fields={fields}
showCheckBox={true} treeSettings={treeSettings}/>);
    }
    export default App;
    ReactDOM.render(<App />, document.getElementById('sample'));

```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // definining the dataSource
    let data: { [key: string]: Object }[] = [
        { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Hot Singles' },
        { id: 3, pid: 1, name: 'Rising Artists' },
        { id: 4, pid: 1, name: 'Live Music' },
        { id: 6, pid: 1, name: 'Best of 2017 So Far' },
        { id: 7, name: 'Sales and Events', hasChild: true },
        { id: 8, pid: 7, name: '100 Albums - $5 Each' },
        { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
        { id: 10, pid: 7, name: 'CD Deals' },
        { id: 11, name: 'Categories', hasChild: true },
        { id: 12, pid: 11, name: 'Songs' },
        { id: 13, pid: 11, name: 'Bestselling Albums' },
        { id: 14, pid: 11, name: 'New Releases' },
        { id: 15, pid: 11, name: 'Bestselling Songs' },
        { id: 16, name: 'MP3 Albums', hasChild: true },
        { id: 17, pid: 16, name: 'Rock' },
        { id: 18, pid: 16, name: 'Gospel' },
        { id: 19, pid: 16, name: 'Latin Music' },
        { id: 20, pid: 16, name: 'Jazz' },
        { id: 21, name: 'More in Music', hasChild: true },
        { id: 22, pid: 21, name: 'Music Trade-In' },
        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    let fields: Object = { dataSource: data, value: 'id', text: 'name',
parentValue:"pid", hasChildren: 'hasChild' };
    let treeSettings: Object = { autoCheck: true }
    return (
        // specifies the tag for render the DropDownTree component
        <DropDownTreeComponent id="dropdownntree" fields={fields}
showCheckBox={true} treeSettings={treeSettings}/>
    );
}
export default App;

```

```
ReactDOM.render(<App />, document.getElementById('sample'));
```

Select All

The Dropdown Tree component has in-built support to select all the tree items using Select All options in the header.

When the `showSelectAll` property is set to true, a checkbox will be displayed in the popup header that allows you to select or deselect all the tree items in the popup.

By default, `Select All` and `unSelect All` text values will be showcased along with the checkbox in the popup header to indicate the action to be performed on checking or unchecking the checkbox. You can customize these name attributes by using `selectAllText` and `unSelectAllText` properties respectively.

INDEX.JSX

```
import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // defining the dataSource
    let data = [
        { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Hot Singles' },
        { id: 3, pid: 1, name: 'Rising Artists' },
        { id: 4, pid: 1, name: 'Live Music' },
        { id: 6, pid: 1, name: 'Best of 2017 So Far' },
        { id: 7, name: 'Sales and Events', hasChild: true },
        { id: 8, pid: 7, name: '100 Albums - $5 Each' },
        { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
        { id: 10, pid: 7, name: 'CD Deals' },
        { id: 11, name: 'Categories', hasChild: true },
        { id: 12, pid: 11, name: 'Songs' },
        { id: 13, pid: 11, name: 'Bestselling Albums' },
        { id: 14, pid: 11, name: 'New Releases' },
        { id: 15, pid: 11, name: 'Bestselling Songs' },
        { id: 16, name: 'MP3 Albums', hasChild: true },
        { id: 17, pid: 16, name: 'Rock' },
        { id: 18, pid: 16, name: 'Gospel' },
        { id: 19, pid: 16, name: 'Latin Music' },
        { id: 20, pid: 16, name: 'Jazz' },
        { id: 21, name: 'More in Music', hasChild: true },
        { id: 22, pid: 21, name: 'Music Trade-In' },
        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    let fields = { dataSource: data, value: 'id', text: 'name', parentValue:
    "pid", hasChildren: 'hasChild' };
    return (
        // specifies the tag for render the DropDownTree component
        <DropDownTreeComponent id="dropdownntree" fields={fields}
        showCheckBox={true} showSelectAll={true} selectAllText={"Check All"}
        unSelectAllText={"UnCheck All"} />;
    )
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));
```

INDEX.TSX

```

import { DropDownTreeComponent } from '@syncfusion/ej2-react-dropdowns';
import * as React from 'react';
import * as ReactDOM from 'react-dom';
function App() {
    // defining the dataSource
    let data: { [key: string]: Object }[] = [
        { id: 1, name: 'Discover Music', hasChild: true, expanded: true },
        { id: 2, pid: 1, name: 'Hot Singles' },
        { id: 3, pid: 1, name: 'Rising Artists' },
        { id: 4, pid: 1, name: 'Live Music' },
        { id: 6, pid: 1, name: 'Best of 2017 So Far' },
        { id: 7, name: 'Sales and Events', hasChild: true },
        { id: 8, pid: 7, name: '100 Albums - $5 Each' },
        { id: 9, pid: 7, name: 'Hip-Hop and R&B Sale' },
        { id: 10, pid: 7, name: 'CD Deals' },
        { id: 11, name: 'Categories', hasChild: true },
        { id: 12, pid: 11, name: 'Songs' },
        { id: 13, pid: 11, name: 'Bestselling Albums' },
        { id: 14, pid: 11, name: 'New Releases' },
        { id: 15, pid: 11, name: 'Bestselling Songs' },
        { id: 16, name: 'MP3 Albums', hasChild: true },
        { id: 17, pid: 16, name: 'Rock' },
        { id: 18, pid: 16, name: 'Gospel' },
        { id: 19, pid: 16, name: 'Latin Music' },
        { id: 20, pid: 16, name: 'Jazz' },
        { id: 21, name: 'More in Music', hasChild: true },
        { id: 22, pid: 21, name: 'Music Trade-In' },
        { id: 23, pid: 21, name: 'Redeem a Gift Card' },
        { id: 24, pid: 21, name: 'Band T-Shirts' },
    ];
    let fields: Object = { dataSource: data, value: 'id', text: 'name',
    parentValue: "pid", hasChildren: 'hasChild' };
    return (
        // specifies the tag for render the DropDownTree component
        <DropDownTreeComponent id="dropdownntree" fields={fields}
        showCheckBox={true} showSelectAll={true} selectAllText={"Check All"}
        unSelectAllText={"UnCheck All"}/>
    );
}
export default App;
ReactDOM.render(<App />, document.getElementById('sample'));

```

Localization in React Drop down tree component

The [Localization](#) library allows you to localize default text content of the Dropdown Tree and it can be localized to any culture by defining the texts and messages of the Dropdown Tree in the corresponding culture. The default locale of the Dropdown Tree is **en** (English). The following table represents the default texts and messages of the Dropdown Tree in **en** culture.

KEY	Text/Message
-----	--------------

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

noRecordsTemplate	No records found
actionFailureTemplate	Request failed
overflowCountTemplate	+\${count} more..
totalCountTemplate	\${count} selected

Accessibility in React Dropdown Tree component

The Dropdown Tree component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Dropdown Tree component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

`<div> - Some features of the component do not meet the requirement.</div>`

`<div> - The component does not meet the requirement.</div>`

WAI-ARIA attributes

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

| Attributes | Purpose |

| --- | --- |

| `role=checkbox` | All list items are contained within the element. |

| `aria-disabled` | Indicates element is perceivable but disabled. |

| `aria-owns` | This attribute contains the ID of the popup list to indicate popup as a child element. |

| `aria-haspopup` | Indicates whether the Dropdown Tree input element has a popup list or not. |

| `aria-expanded` | Indicates the state of the popup list for Dropdown Tree and the parent node's expansion status for TreeView. |

| `aria-activedescendent` | This attribute holds the ID of the active list item to focus its descendant child element. |

| `aria-labelledby` | This attribute points to the element(s) labeling the element it's applied to. |

| `aria-describedby` | This attribute points to the element(s) describing the one it's set on. |

| `role=tree` | All tree nodes are contained within the element. |

| `role=treeitem` | Specifies the role of each tree node in a selectable TreeView and its containment within the tree. |

| `role=group` | Specifies the role of each parent node container in the TreeView. |

| `role=checkbox` | Indicates checkbox control along with treeitem element. |

| `aria-multiselectable` | Indicates whether the TreeView enables multiple selection or not. |

| `aria-selected` | Indicates the selected node. |

| `aria-level` | Indicates the level of node in TreeView. |

| `aria-checked` | Indicates the current checked state of TreeView checkbox. |

| `aria-label` | Indicates the contextual message for the TreeView checkbox and Dropdown Tree. |

| `aria-activedescendant` | Identifies the currently active element when focusing on the TreeView. |

Keyboard interaction

The Dropdown Tree component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Dropdown Tree component.

| Interaction Keys | Description |

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

| Alt + Down | Opens the popup. |

| Alt + Up | Closes the popup. |

| Esc(Escape) | Closes the popup when it is in an open state. |

| Arrow Up | Goes to the previous item in the popup. |

| Arrow Down | Goes to the next item in the popup. |

| Arrow Right | Expands the current item in the popup. |

| Arrow Left | Collapses the current item in the popup. |

| Home | Goes to the first item in the popup. |

| End | Goes to the last item in the popup. |

| Enter | Selects the focused item in the popup. |

| Space | Checks the current item in the popup. |

Ensuring accessibility

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

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

See also

- [Accessibility in Syncfusion React components](#)