# 1. Menu

## Options Used

1. **dataSource**: Configures the data source for the Menu.
2. **adaptivityEnabled**: Enables adaptivity for the Menu when the orientation is horizontal.
3. **orientation**: Specifies the orientation of the Menu.
   * Options: ‘horizontal’, ‘vertical’
4. **animation**: Configures the animation properties for the Menu.
   * **show**: Defines the show animation.
     + **type**: Specifies the type of animation.
     + **from**: Specifies the starting state of the animation.
     + **to**: Specifies the ending state of the animation.
     + **duration**: Specifies the duration of the animation.
   * **hide**: Defines the hide animation.
     + **type**: Specifies the type of animation.
     + **from**: Specifies the starting state of the animation.
     + **to**: Specifies the ending state of the animation.
     + **duration**: Specifies the duration of the animation.
5. **disabledExpr**: Specifies the field that provides the disabled state of menu items.
6. **cssClass**: Specifies a custom CSS class for the Menu.
7. **disabled**: Enables or disables the Menu.
8. **displayExpr**: Specifies the field that provides the display text for menu items.
9. **items**: Defines the menu items.
   * **beginGroup**: Specifies whether to begin a group for the item.
   * **items**: Specifies the submenu items.
   * **icon**: Specifies the icon for the item.
   * **closeMenuOnClick**: Specifies whether to close the menu when the item is clicked.
   * **selectable**: Specifies whether the item is selectable.
   * **selected**: Specifies whether the item is selected.
   * **template**: Specifies a custom template for the item.
10. **itemsExpr**: Specifies the field that provides the submenu items.
11. **hideSubmenuOnMouseLeave**: Specifies whether to hide the submenu when the mouse leaves.
12. **itemTemplate**: Custom template for the menu items.
13. **selectByClick**: Specifies whether to select items by click.
14. **selectedItem**: Specifies the selected item.
15. **selectedExpr**: Specifies the field that provides the selected state of menu items.
16. **selectionMode**: Specifies the selection mode for the Menu.
    * Options: ‘none’, ‘single’
17. **showFirstSubmenuMode**: Specifies the mode to show the first submenu.
    * Options: ‘onClick’, ‘onHover’
18. **showSubmenuMode**: Specifies the mode to show submenus.
    * Options: ‘onClick’, ‘onHover’
19. **submenuDirection**: Specifies the direction of submenus.
    * Options: ‘auto’, ‘leftOrTop’, ‘rightOrBottom’

## Events Used

1. **onItemClick**: Triggered when a menu item is clicked.
2. **onItemContextMenu**: Triggered when a context menu is opened for a menu item.
3. **onItemRendered**: Triggered when a menu item is rendered.
4. **onSubmenuHidden**: Triggered when a submenu is hidden.
5. **onSubmenuHiding**: Triggered when a submenu is hiding.
6. **onSubmenuShowing**: Triggered when a submenu is showing.
7. **onSubmenuShown**: Triggered when a submenu is shown.

## Methods Used

1. **dxMenu**: Initializes the Menu component.
2. **option**: Sets the specified option of a component instance.

# 2. TreeView

## Options Used

1. **dataSource**: Configures the data source for the TreeView.
2. **animationEnabled**: Enables or disables animation for the TreeView.
3. **displayExpr**: Specifies the field that provides the display text for tree items.
4. **createChildren**: Custom logic to load data on demand.
5. **dataStructure**: Specifies the structure of the data.
   * Options: ‘plain’, ‘tree’
6. **disabledExpr**: Specifies the field that provides the disabled state of tree items.
7. **expandAllEnabled**: Enables or disables the ability to expand all nodes.
8. **expandedExpr**: Specifies the field that provides the expanded state of tree items.
9. **expandEvent**: Specifies the event to expand nodes.
   * Options: ‘dblClick’, ‘click’
10. **expandNodesRecursive**: Specifies whether or not all parent nodes of an initially expanded node are displayed expanded.
11. **hasItemsExpr**: Specifies the field that indicates if a node has children.
12. **itemHoldTimeout**: Specifies the period in milliseconds before the onItemHold event is raised.
13. **keyExpr**: Specifies the field that provides the key for tree items.
14. **noDataText**: Specifies the text displayed when there are no items.
15. **selectByClick**: Specifies whether to select items by click.
16. **selectedItem**: Specifies the selected item.
17. **selectedExpr**: Specifies the field that provides the selected state of tree items.
18. **selectionMode**: Specifies the selection mode for the TreeView.
    * Options: ‘none’, ‘single’, ‘multiple’
19. **showCheckBoxesMode**: Specifies the mode to show checkboxes.
    * Options: ‘normal’, ‘selectAll’, ‘none’
20. **selectAllText**: Specifies the text for the “Select All” checkbox.
21. **selectNodesRecursive**: Specifies whether to select all child nodes when a parent node is selected.
22. **virtualModeEnabled**: Enables or disables virtual mode to load child nodes on demand.
23. **useNativeScrolling**: Specifies whether to use native scrolling.
24. **parentIdExpr**: Specifies the field that provides the parent ID for tree items in plain structure.
25. **rootValue**: Specifies the root value for tree items in plain structure.
26. **scrollDirection**: Specifies the scroll direction.
    * Options: ‘both’, ‘horizontal’, ‘vertical’
27. **searchEditorOptions**: Configures the search editor options.
    * **visible**: Specifies whether the search editor is visible.
    * **width**: Specifies the width of the search editor.
    * **placeholder**: Specifies the placeholder text for the search editor.
28. **searchExpr**: Specifies the fields to search.
29. **searchMode**: Specifies the search mode.
    * Options: ‘contains’, ‘startswith’, ‘equals’
30. **searchEnabled**: Enables or disables search.
31. **searchTimeout**: Specifies the timeout for search in milliseconds.

## Events Used

1. **onItemClick**: Triggered when a tree item is clicked.
2. **onItemContextMenu**: Triggered when a context menu is opened for a tree item.
3. **onItemRendered**: Triggered when a tree item is rendered.
4. **onSubmenuHidden**: Triggered when a submenu is hidden.
5. **onSubmenuHiding**: Triggered when a submenu is hiding.
6. **onSubmenuShowing**: Triggered when a submenu is showing.
7. **onSubmenuShown**: Triggered when a submenu is shown.
8. **onItemSelectionChanged**: Triggered when the selection of a tree item changes.
9. **onItemCollapsed**: Triggered when a tree item is collapsed.
10. **onItemExpanded**: Triggered when a tree item is expanded.
11. **onItemHold**: Triggered when a tree item is held.
12. **onSelectAllValueChanged**: Triggered when the “Select All” value changes.

## Methods Used

1. **dxTreeView**: Initializes the TreeView component.
2. **option**: Sets the specified option of a component instance.
3. **getSelectedNodes**: Retrieves the selected nodes from the TreeView.

# 3. LoadIndicator

## Options Used

{  
 dataField: "images",  
 caption: "Images",  
 alignment: "center",  
 cellTemplate: function (container, options) {  
 // Create a container for the load indicator  
 const $indicator =  
 $("<div id='light'>").appendTo(container);  
  
 // Initialize the dxLoadIndicator  
 $indicator.dxLoadIndicator({  
 height: 150,  
 width: 150,  
 indicatorSrc: "../Data/loader.gif",  
 });  
 // Customize the images column to display the first image  
 // Create an image element  
 const $img = $("<img>").attr("src", options.value).css({  
 maxWidth: "100%",  
 height: "250px",  
 display: "none", // Hidden initially  
 });  
 // Append the image to the container  
 $img.appendTo(container);  
 // When the image loads, hide the indicator and show the image  
 $img.on("load", () => {  
 setTimeout(() => {  
 $indicator  
 .dxLoadIndicator("instance")  
 .option("visible", false);  
 $img.css("display", "block");  
 }, 3000); // 1-second delay  
 });  
 },  
 },

## Events Used

1. **load**: Triggered when the image in the cellTemplate is loaded.

# 4. Load Panel

## Options Used

1. **closeOnOutsideClick**: Allows closing the LoadPanel when clicking outside of it.
   * Options: boolean, function
2. **shadingColor**: Specifies the shading color.
3. **position**: Specifies the position of the LoadPanel.
4. **visible**: Controls the visibility of the LoadPanel.
5. **delay**: Specifies the delay before showing the LoadPanel.
6. **message**: Specifies the message to be displayed in the LoadPanel (note: may not work with Material UI CSS).
7. **showIndicator**: Shows or hides the loading indicator.
8. **showPane**: Shows or hides the LoadPanel pane.
9. **shading**: Enables or disables shading.
10. **hideOnOutsideClick**: Specifies whether to hide the LoadPanel when clicking outside of it.

## Methods Used

1. **dxLoadPanel**: Initializes the LoadPanel component.
2. **instance**: Retrieves the instance of the LoadPanel component.
3. **hide**: Hides the LoadPanel.

## Events Used

1. **onShown**: Triggered when the LoadPanel is shown.
2. **onHidden**: Triggered when the LoadPanel is hidden.

# 5. Popup

## Options Used

1. **title**: Specifies the title displayed on top of the popup.
2. **width**: Specifies the width of the popup.
3. **height**: Specifies the height of the popup.
4. **visible**: Controls the visibility of the popup.
   * Options: true, false
5. **closeOnOutsideClick**: Allows closing the popup when clicking outside of it.
6. **container**: Specifies the container element for the popup.
7. **showCloseButton**: Shows or hides the close button on the popup.
8. **dragEnabled**: Enables or disables dragging of the popup.
9. **resizeEnabled**: Enables or disables resizing of the popup.
10. **shading**: Enables or disables shading.
11. **shadingColor**: Specifies the shading color.
12. **fullScreen**: Displays the popup in full screen.
13. **showTitle**: Shows or hides the title of the popup.
14. **animation**: Configures the animation properties for the popup.
    * **show**: Defines the show animation.
      + **type**: Specifies the type of animation.
      + **duration**: Specifies the duration of the animation.
    * **hide**: Defines the hide animation.
      + **type**: Specifies the type of animation.
      + **duration**: Specifies the duration of the animation.
15. **toolbarItems**: Defines the items displayed in the toolbar of the popup.
    * **toolbar**: Specifies the toolbar’s location.
    * **location**: Specifies the location of the item within the toolbar.
    * **widget**: Specifies the widget type.
    * **options**: Specifies the options for the widget.
16. **position**: Specifies the position of the popup.
    * **my**: Specifies the alignment of the popup.
    * **at**: Specifies the target alignment.
    * **of**: Specifies the target element.
17. **contentTemplate**: Custom template for the popup content.

## Methods Used

1. **dxPopup**: Initializes the Popup component.
2. **instance**: Retrieves the instance of the Popup component.
3. **hide**: Hides the Popup.
4. **option**: Sets the specified option of a component instance.

## Events Used

1. **onShowing**: Triggered when the popup is about to show.
2. **onShown**: Triggered when the popup is shown.
3. **onHiding**: Triggered when the popup is about to hide.
4. **onHidden**: Triggered when the popup is hidden.

# 6. Popover

## Options Used

1. **title**: Specifies the title displayed on top of the popover.
2. **width**: Specifies the width of the popover.
3. **height**: Specifies the height of the popover.
4. **visible**: Controls the visibility of the popover.
   * Options: true, false
5. **closeOnOutsideClick**: Allows closing the popover when clicking outside of it.
6. **showCloseButton**: Shows or hides the close button on the popover.
7. **showTitle**: Shows or hides the title of the popover.
8. **position**: Specifies the position of the popover.
   * **my**: Specifies the alignment of the popover.
   * **at**: Specifies the target alignment.
   * **of**: Specifies the target element.
9. **animation**: Configures the animation properties for the popover.
   * **show**: Defines the show animation.
     + **type**: Specifies the type of animation.
     + **duration**: Specifies the duration of the animation.
   * **hide**: Defines the hide animation.
     + **type**: Specifies the type of animation.
     + **duration**: Specifies the duration of the animation.
10. **toolbarItems**: Defines the items displayed in the toolbar of the popover.
    * **toolbar**: Specifies the toolbar’s location.
    * **location**: Specifies the location of the item within the toolbar.
    * **widget**: Specifies the widget type.
    * **options**: Specifies the options for the widget.
11. **hoverStateEnabled**: Enables or disables hover state for the popover.
12. **showEvent**: Specifies the event to show the popover.
13. **hideEvent**: Specifies the event to hide the popover.
14. **target**: Specifies the target element for the popover.
15. **contentTemplate**: Custom template for the popover content.

## Methods Used

1. **dxPopover**: Initializes the Popover component.
2. **instance**: Retrieves the instance of the Popover component.
3. **show**: Shows the Popover.
4. **hide**: Hides the Popover.
5. **option**: Sets the specified option of a component instance.

## Events Used

1. **onHidden**: Triggered when the popover is hidden.

# 7. Toast

## Options Used

1. **message**: Specifies the message to be displayed in the toast.
2. **type**: Specifies the type of the toast.
   * Options: ‘custom’, ‘error’, ‘info’, ‘success’, ‘warning’
3. **displayTime**: Specifies the display time of the toast in milliseconds.
4. **animation**: Configures the animation properties for the toast.
   * **show**: Defines the show animation.
     + **type**: Specifies the type of animation.
     + **duration**: Specifies the duration of the animation.
   * **hide**: Defines the hide animation.
     + **type**: Specifies the type of animation.
     + **duration**: Specifies the duration of the animation.
5. **closeOnClick**: Specifies whether the toast should close when clicked.
6. **width**: Specifies the width of the toast.
7. **height**: Specifies the height of the toast.
8. **maxWidth**: Specifies the maximum width of the toast.
9. **minWidth**: Specifies the minimum width of the toast.
10. **rtlEnabled**: Specifies whether to enable right-to-left text direction.
11. **closeOnOutsideClick**: Specifies whether the toast should close when clicking outside of it.
12. **closeOnSwipe**: Specifies whether the toast should close when swiped.
13. **shading**: Specifies whether to enable shading (commented out in the provided code).
14. **shadingColor**: Specifies the shading color (commented out in the provided code).

## Methods Used

1. **dxToast**: Initializes the Toast component.
2. **show**: Shows the Toast.