8 Dialogues And Notification

8.1 LoadIndicator

The LoadIndicator is a UI element notifying the viewer that a process is in progress.

Specify Dimensions

LoadIndicator allows us to specify its dimensions. We can set the *height* and *width* properties to the size we need.

Apply a Custom Indicator

If we want to apply a custom indicator, we can assign the path to a new image or GIF to the *indicatorSrc* property.

We can use *LoadIndicator* within other UI components also.

8.2 LoadPanel

The LoadPanel is an overlay component used to notify users that a process is in progress.

Show and Hide the Indicator and Pane

LoadPanel elements *(a message and an animated load indicator)* are displayed on a *pane*. If we want to hide it, we can set the *showPane* property to *false*. We can also disable the *showIndicator* property to *hide* the animated *load indicator*. In this case, the LoadPanel displays only the message.

Configure the Background Shade

When LoadPanel is displayed, it shades the background. We can use the *shadingColor* property to specify the color of the shade. We can also specify the element that should be shaded. For this, assign the element's CSS selector to the container property. If we don’t want to shade the background, *disable* the *shading* property.

Show and Hide LoadPanel

We can change the LoadPanel visibility by setting the *visible* property. Alternatively, we can call the *show()* and *hide()* or *toggle(showing)* methods.

Users can hide LoadPanel when they click outside it if you enable the *hideOnOutsideClick* property.

LoadPanel also allows you to handle the show and hide events. We can use the *onShowing* and *onHiding* functions to handle the events before they occur and possibily cancel them. We can use the *onShown* and *onHidden* functions to perform required actions after the events are raised.

8.3 Popup

The JavaScript Popup is a pop-up window overlaying the current view.

Show and Hide the Popup

We can call the *show()* method to display the popup. We can close popup, by choosing one of the following options :

Built-in close button (Enable the *showCloseButton* property to display the Close button in a popup's top toolbar)

Custom close button

On outside click (Enable the *hideOnOutsideClick* property to allow users to hide the popup by clicking outside the component)

Configure the popup

The popup inner area is divided into three parts :

Top toolbar

We can set *showTitle* to *true* and use the *title* property to specify the *caption*. The *Close button* will appear if you do *not disable* the *showCloseButton* property.

Also we can add *toolbarItems* markup and set each item's toolbar property to top.

Content area

To populate the popup with content, we can use the *contentTemplate* property.

Bottom toolbar

To enable the bottom toolbar, we can declare the *toolbarItems* array. We can set each item's toolbar property to bottom.

Resize and Position

To specify popup size, we can use the *height* and *width* properties.

We can set the *my, at,* and *of* properties of the *position* object.

We uses the *container* property to select the container in which you want to render the popup.

If we set the *container* property to an element on the page, the *shading* applies to this element.

We can turn on the *dragEnabled* option to allow users to *move* the popup around the page.

8.4 Popover

The Popover component shows pop-up notifications within a box with an arrow that points to a specified UI element.

Attach Popover to a Page Element

Popover displays an arrow that points to a page element. To specify the element, set the *target* property to a CSS selector.

We can use the *position* property to position Popover relative to the target element. If we don't specify this property, Popover is *displayed under the element*.

Show and Hide Popover

The *showEvent* and *hideEvent* properties allow us to *show* and *hide* Popover in response to certain events. These properties can accept *one or multiple names of DOM events* or *DevExtreme UI events separated by a space character*.

We can also specify a *delay* before the events occur. We can Set the *showEvent* and *hideEvent* properties to an object with the name (one or multiple event names) and *delay* properties.

Specify Content

Popover consists of *content area* and a *title*. We can specify static content in the HTML markup. If Popover should display dynamic content, we can use the *contentTemplate* property to specify a template. To display the title, we can enable the *showTitle* property and set the *title* property to the title text.

Animate Popover

We can show and hide Popover with animation effects, by assigning an object with the *show* and *hide* fields to the *animation* property. Each of these fields accepts an object that configures the animation *type* and other properties.

Shade the Background

We can show Popover with a shaded background. To do this, we can enable the *shading* property and specify a *shadingColor*.

8.5 Popup vs Popover

Usage : Popups are used for more significant interactions requiring user focus, whereas Popovers are used for supplementary information related to a specific element.

Modal vs Non-Modal : Popups can be modal, blocking other interactions; Popovers are non-modal.

Positioning : Popups can be positioned anywhere on the screen, while Popovers are positioned relative to a specific element.

Content Complexity : Popups can handle complex content; Popovers are typically for simpler, more concise content.

Overlay : Popups usually include an overlay; Popovers do not.

8.6 Toast

The Toast is a UI component that displays pop-up notifications.

We can specify one of the four predefined types of notifications, depending on the *mode* of the message :
'info'(A toast with a message), *'warning'*(yellow toast), *'error'*(red toast), *'success'*(green toast).
We can also customize the Toast appearance by setting the type property to *'custom'* and use a *contentTemplate*.
If we need to specify other Toast properties in addition to *type* and *displayTime*, we can set other Toast properties, such as *shading, position, width, height,* and others.