



UWP



Toolbars



WINDOWS > GUIDES > USER INTERFACE > TOOLBARS

Customizing the toolbar in our UWP viewer

Nutrient comes with a very useful toolbar by default, but it's also important for individuals to modify toolbar items to suit their needs. As such, it's possible to add, remove, and reorder all the buttons on the toolbar, and there are also more advanced options, like the responsive grouping of buttons and custom icons.

Adding a toolbar item

The next section covers how to add a toolbar item.

Built-in toolbar items

There are many built-in toolbar items available. A list of these can be found in the [API reference](#). The following code shows how to add a single built-in item:

```
1 var toolbarItems = PdfView.GetToolbarItems();  
2 toolbarItems.Add(new PagerToolbarItem());  
3 await PdfView.SetToolbarItemsAsync(toolbarItems);
```



In some instances, you might want to configure how the toolbar looks or set it up in a particular order. To do this, you can create an `IList` of `IToolbarItem`s and pass this to `PdfView.SetToolbarItemsAsync`. The ordering of the buttons is determined by the order of the container:



ASK AI



```

1 var defaultToolBarItems = new List<IToolbarItem>
2 {
3     new SidebarThumbnailsToolBarItem(),
4     new SidebarDocumentOutlineToolBarItem(),
5     new SidebarAnnotationToolBarItem(),
6     new PagerToolBarItem(),
7     new LayoutConfigToolBarItem(),
8     new PanToolBarItem(),
9     new ZoomOutToolBarItem()
10 };

```



Some built-in items are automatically grouped together with a dropdown arrow. For example, all the sidebar options are located under the same button, all line drawing annotations will be grouped together, and the highlighter and ink tool can be found under the same button.

Also note that excluding annotation items will not affect the user's editing capabilities for existing annotations of the specified type. [This guide](#) offers more information on controlling annotation editing capabilities.

Custom toolbar items

Along with the useful built-in toolbar items, we have also made it possible to provide your own custom toolbar items. Currently, you can define a `ButtonToolBarItem` and a `ToggleButtonToolBarItem`. The `ButtonToolBarItem` is a one-shot button that has no state, whereas the `ToggleButtonToolBarItem` has a state in which it can either be selected or not. This will be shown in the UI via a change of color. These buttons are added in the same way, so we'll only show the `ButtonToolBarItem` example here:



```

1 var toolbarItem = new ButtonToolBarItem()
2 {
3     Attributes =
4     {
5         Id = "id",
6         Title = "title",
7         ClassName = "class-name",
8         ResponsiveGroup = "responsive-group",
9         MediaQueries = new List<string> { "max-width" },
10    },
11    Disabled = false,
12    Icon = new Uri("ms-appx-web:///Assets/ToolBarIcons/status_completed.svg")
13 };
14 toolbarItem.OnItemPressEvent += ToolbarItem_OnPressAsync;
15
16 var toolbarItems = PdfView.GetToolBarItems();

```



```
17 toolbarItems.Add(toolbarItem);
18 await PdfView.SetToolbarItemsAsync(toolbarItems);
```

Set the `Id` to ensure the correct `onPress` listener is called, and set a `Title` to be shown. Note that if an `Icon` is defined, the title will be used to show a hint if the mouse hovers over the icon button.

`OnItemPressEvent` has a function callback assigned to receive notifications when the user clicks the custom button. The toolbar item can then be inserted into the list of toolbar items. The position of the item in the list determines the position where it appears on the toolbar. The `Selected` property of `ToggleButtonToolbarItem` denotes whether or not the item is selected. It can also be queried at runtime.

Removing a toolbar item

Removing an item from the toolbar is even simpler than adding an item is:

```
1 var toolbarItems = PdfView.GetToolbarItems();
2 toolbarItems.RemoveAt(0);
3 await PdfView.SetToolbarItemsAsync(toolbarItems);
```



The code above will remove the item at position zero, i.e. it will remove the item at the start of the toolbar. Extra logic can be applied to find the required item and remove it from the list before

`SetToolbarItemsAsync`.

Reordering the toolbar items

As seen when adding and removing items, the order is determined by the order an item is inserted into the list. As such, you can simply reorder the items on the toolbar by reordering the items in the list. The following example shows how to randomly shuffle the items on the toolbar:

```
1 var toolbarItems = PdfView.GetToolbarItems();
2 var shuffledList = toolbarItems.OrderBy(a => Guid.NewGuid()).ToList();
3 await PdfView.SetToolbarItemsAsync(shuffledList);
```



Responsive group

As the size of screen is not often known, it is great to have the ability to dynamically hide toolbar items dependent upon a given metric. With Nutrient, you can define a `ResponsiveGroupToolbarItem` item that other `IToolbarItem`s can reference in order to create a collapsed grouping:

```
1 var responsiveGroup = new ResponsiveGroupToolbarItem
2 {
3     Attributes =
4     {
5         Id = "my-group",
6         MediaQueries = new List<string> { "(max-width:4000px)" },
7         Title = "Responsive"
8     }
9 };
10
11 var toolbarItem = new PagerToolbarItem
12 {
13     Attributes =
14     {
15         ResponsiveGroup = "my-group"
16     }
17 };
18 toolbarItem.OnItemPressEvent += ToolbarItem_OnPressAsync;
19
20 var toolbarItems = PdfView.GetToolbarItems();
21 toolbarItems.Insert(0, responsiveGroup);
22 toolbarItems.Add(toolbarItem);
23 await PdfView.SetToolbarItemsAsync(toolbarItems);
```

If the view is less than 600px wide, then the `ResponsiveGroupToolbarItem` will be shown at index 0. When the responsive group button is pressed, the custom button is revealed. But when the view is larger than 600px wide, the button will be shown as normal.

Dropdown group

The grouping of toolbar items can save space in the toolbar and make for logical grouping specific to your custom UI. A good example of this is how, by default, `Ink`, `Highlighter`, and `TextHighlighter` appear as one item with an arrow by the side.

It's possible to override this behavior and/or create your own groupings.

To add your own group, simply set the `DropdownGroup` string. Each subsequent item that also has a `DropdownGroup` of the same string will be grouped together:

```
1 // Create an arrow toolbar item in the dropdown group of `my-group`.
2 var arrowToolbarItem = new ArrowToolbarItem()
3 {
4     Attributes =
5     {
6         DropdownGroup = "my-group",
7     }
8 };
```



Some default items are already part of a default dropdown group. To override this behavior, simply set `DropdownGroup` with an empty string:

```
1 // Create an ellipse toolbar item without a dropdown group.
2 var ellipseToolbarItem = new EllipseToolbarItem()
3 {
4     Attributes =
5     {
6         DropdownGroup = "",
7     }
8 };
```



Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

