Customizing the Light Navigation Control

The *Light Navigation Control* is design to be highly customizable so you can fully customize and style its layout by editing the default templates that the control provides.

Customizing the Templates of Navigation Control

In order to show the edit template window follow these steps:

1) Drag the *Light Navigation Control* on the page
2) Click the *Edit* button in the upper-right corner
3) Select the template that you want to modify from the drop down.
4) Click *Edit Selected Template*. The *Edit Template* dialog will open.

How to customize the Horizontal With DropDown Menus template

Some of the templates by default use Telerik Kendo controls, so you can customize them using the options, methods and events from the Kendo API:

http://docs.kendoui.com/api/web/menu

http://docs.kendoui.com/api/web/treeview

Let's assume that we want to modify the default behavior of the menu template in order to open the child menus on click instead on hover. The Kendo Menu control exposes property *openOnClick* which could be used for this functionality, still this property can be used only on the initial loading of the control. Based on this we could remove the default initialization of the Kendo menu that the template provide and add external JavaScript file where we initialize the Kendo menu with *openOnClick* set to true. Here is a small walkthrough how it could be customized to open on click instead of hover.

1) Open the template for edit
2) Create new JavaScript file inside the SitefinityWebApp project with the following content:

```
$(document).ready(function () {
    var kendoMenu = $('.sfNavHorizontalDropDown').kendoMenu({
        animation: false,
        openOnClick:true
    }).data('kendoMenu');

});
```

3) Add it to the template just below the <ResourceLinks> section:

```
<script src="/MyScript.js" > </script>
```

4) Click *Save Changes* button. You will return to the *Edit* dialog

Similarly to the example with TreeView the same modification in the default behavior could be achieved by modifying the script tag directly in the *Horizontal With DropDown Menus* template instead of creating external js file:

```
var kendoMenu =  $($get('<%= navigationUl.ClientID %>')).kendoMenu({
     animation: false,
     openOnClick:true
 }).data('kendoMenu');
```

How to customize the Tree(Vertical With Sub-Levels) template

Here is a small walkthrough how it could be customized in order to expand its nodes on the initial loading through <script> tag directly in the template.

1) Open *Edit Template* dialog for the *Tree(Vertical With Sub-Levels)* .
2) Add the following snippet at the bottom of the <script> tag

```
// expands all nodes
    kendoTreeView.expand(".k-item");
```

3) Click *Save Changes* button. You will return to the *Edit* dialog
4) Click *Save* button.
5) Publish the page and notice that the tree will be rendered in the expanded state.

You can achieve the same functionality by moving the mentioned customization in external JavaScript file. We recommend using this approach since this way the browser caches the external scripts:

1) Create new JavaScript file inside the SitefinityWebApp project with the following content:

```
$(document).ready(function () {

    var kendoTreeView = $('.sfNavTreeview').kendoTreeView({
        animation: false
    }).data('kendoTreeView');

    kendoTreeView.expand(".k-item");

});
```

2) Add it to the template just below the <ResourceLinks> section:

```
<script src="/MyScript.js" > </script>
```

3) Click *Save Changes* button. You will return to the *Edit* dialog

How to limit navigation to a certain level of depth

The example is for the Tree template but it is applicable for all other templates as well. The goal is to limit the tree to render page nodes up to two levels in depth.

1) Open Edit Template dialog for the Tree(Vertical With Sub-Lebels).

You will see this snippet at the bottom:

```
    <ul class="sfNavTreeview sfNavList" runat="server" id="navigationUl">
        <navigation:NavigationContainer runat="server" DataSourceID="dataSource">
            <Templates>
                 <navigation:NavigationTemplate>
                      <Template>
                          <li>
                              <a runat="server" href='<%# Eval("Url") %>'><%# Eval(
"Title") %></a>
                              <ul runat="server" id="childNodesContainer"></ul>
                          </li>
                      </Template>
                     <SelectedTemplate>
                          <li>
                              <a runat="server" href='<%# Eval("Url") %>' class="sf
Sel"><%# Eval("Title") %></a>
                              <ul runat="server" id="childNodesContainer"></ul>
                          </li>
                      </SelectedTemplate>
                  </navigation:NavigationTemplate>
              </Templates>
        </navigation:NavigationContainer>
    </ul>
```

The definition of NavigationContainer states that every node will be rendered with the specified NavigationTemplate and their respective child nodes will be rendered inside the UL with id **childNodesContainer**.

2) Add another NavigationTemplate element and specify Level attribute for both of them as shown in this snippet:

```
    <ul class="sfNavTreeview sfNavList" runat="server" id="navigationUl">
        <navigation:NavigationContainer runat="server" DataSourceID="dataSource">
            <Templates>
                <navigation:NavigationTemplate Level="0">
                    <Template>
                        <li>
                            <a runat="server" href='<%# Eval("Url") %>'><%# Eval(
"Title") %></a>
                            <ul runat="server" id="childNodesContainer"></ul>
                        </li>
                    </Template>
                </navigation:NavigationTemplate>
```

```
                <navigation:NavigationTemplate Level="1">
                    <Template>
                        <li>
                            <a runat="server" href='<%# Eval("Url") %>'><%# Eval(
"Title") %></a>
                        </li>
                    </Template>
                </navigation:NavigationTemplate>
            </Templates>
        </navigation:NavigationContainer>
    </ul>
```

This template specifies a template for nodes depending on their depth in the sitemap. You can see that the second template does not specify a childNodesContainer element thus nodes at this level won't have their children rendered.

Levels of the navigation can be limited also by using the control designer of the Light Navigation control using "Levels to include" dropdown.

How to allow an extra level in the "Sitemap in rows" default template

1) Open Edit Template dialog for Sitemap in rows (up to 2 levels).

You will see this snippet at the bottom:

```
    <ul class="sfNavList sfNavVerticalSiteMap">
        <navigation:NavigationContainer runat="server" DataSourceID="dataSource" >
            <Templates>
                    <navigation:NavigationTemplate>
                        <Template>
                            <li>
                                <a runat="server" href='<%# Eval("Url") %>'><%# Eval(
"Title") %></a>
                                <ul runat="server" id="childNodesContainer"></ul>
                            </li>
                        </Template>
                    </navigation:NavigationTemplate>
                    <navigation:NavigationTemplate Level="1">
                        <Template>
                            <li>
                                <a runat="server" href='<%# Eval("Url") %>'><%# Eval(
"Title") %></a>
                            </li>
                        </Template>
                    </navigation:NavigationTemplate>
                </Templates>
        </navigation:NavigationContainer>
    </ul>
```

This definition of NavigationContainer states that nodes that are one level in depth won't have any children rendered because their template does not contain a childNodesContainer element thus leaving the navigation with a maximum of two levels of depth.

2) To allow a third level all you need to do is set the second NavigationTemplates' Level property to 2. This will cause the nodes of the second level to use the NavigationTemplate that does not have a level specified. Their children nodes will find the NavigationTemplate with Level="2" and use it thus limiting the rendering to 3 levels.

Applying Responsive Transformations to Navigation Control

1. Go to section Design> Responsive and Mobile Design
2. Create new rule or open existing for which you want to enable Navigation transformation
3. In the Rule screen expand section "Navigation transformations"
4. Select to which navigation widgets to apply transformation (could be to all or ones marked with certain class)
5. Choose the transformation. The ones supported out of the box are: To Dropdown list and To Toggle Vertical Menu
6. Click on Done, the selected transformation will start applying to the navigation

How to customize the Toggle Menu HTML

1. Go to section Design> Widget Templates
2. Find template "Toggle menu", open it and apply the needed modifications
   The toggle menu will be updated in all navigation transformations

How to customize the Toggle Menu transformation CSS

1. Go to section Settings> Advanced Settings
2. Browse to ResponsiveDesign>NavigationTransformations>ToToggleMenu
3. Edit the CSS and Save Changes
   The transformation CSS will be updated.