

How to Customize the Better Web Help DITA OT Plugin

Contents

- Custom Heading & Logo for BWH.....3**
- Set Up a Side Nav Drawer for BWH.....4**
- Adding a Manifest File to your BWH..... 6**
- BWH Styling.....7**
- Custom Fonts with Better Web Help..... 8**
- BWH user-styling.css Width Example.....9**

Custom Heading & Logo for BWH

How to add a custom heading and logo to the Better Web Help (bwh) DITA OT plugin output.

The default page for the bwh output is automatically built to insert any heading it finds in the output directory. So, all you have to do is design it.


1. First, design a heading. It might look something like this:

```
<div>
  
  <h1>Web Help</h1>
</div>
```

2. There is styling already associated with <h1> and tags, however you are welcome to override that styling with inline CSS, such as:

```
<h1 style="background-color: red; text-weight: bold; font-size: 28;">
Web Help</h1>
```

3. Once the header is created, save it as "heading.html" and put it in the root directory of the DITA OT output.

4.  **Note:** The heading.html file will only contain what's necessary for the heading (i.e. it can begin with the <div> as shown above). There is no need for <html>, <head>, or <body> tags.

5. If you have other content such as images or other files, it is recommended to have a "heading" folder into which you can put all of those. As in Step 1, in your source for your heading, make sure each reference looks into the heading folder. ()

6. The next time you load the page, your header should pop up on top of the web help. If it doesn't look the way you want, go back in and adjust the inline styling in heading.html.

So, to add a heading to your web help system, all you have to do is include a heading.html file in the root directory of your web app and you're all set.

Set Up a Side Nav Drawer for BWH

How to set up a sliding side nav panel on mobile for Better Web Help (bwh) plugin for the DITA OT.

This tutorial assumes you've already built the BWH site using the DITA Open Toolkit. I will use {root} as the root directory of the webpage (where you specified "outdir" in the DITA OT).

1. Open {root}/index.html in a text editing program (Notepad, TextEdit, etc).
2. You should see a document that begins with something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=0"
      name="viewport" />
    <meta name="viewport" content="width=device-width, minimal-ui" />
    <meta name="mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <link rel="icon" sizes="196x196" href="help-icon.png" />
    <link rel="apple-touch-icon" sizes="196x196" href="help-icon.png" />
    <link rel="stylesheet" type="text/css" href="scripts/frameless.css" />
    <link rel="stylesheet" type="text/css" href="scripts/user-fonts.css" />
    <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/1.11.1/jquery.min.js">///</script>
    <script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/
jqueryui/1.11.0/jquery-ui.min.js">///</script>
    <script type="text/javascript" src="scripts/typeahead.bundle.js">///</
script>
    <script type="text/javascript" src="scripts/queryString.js">///</script>
    <script type="text/javascript" src="scripts/dragdealer.js">///</script>
    <script type="text/javascript" src="scripts/lunr.js">///</script>
    <script type="text/javascript" src="scripts/lunrScripts.js">///</script>
    <script type="text/javascript" src="scripts/loadContent.js">///</script>
    ...
```

3. Right below the line that contains 'href="frameless.css"', add the following line:

```
<link rel="stylesheet" type="text/css" href="scripts/side-drawer.css" />
```

4. It should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=0"
      name="viewport" />
    <meta name="viewport" content="width=device-width, minimal-ui" />
    <meta name="mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <link rel="icon" sizes="196x196" href="help-icon.png" />
    <link rel="apple-touch-icon" sizes="196x196" href="help-icon.png" />
    <link rel="stylesheet" type="text/css" href="scripts/frameless.css" />
    <link rel="stylesheet" type="text/css" href="scripts/side-drawer.css" />
```


```
<link rel="stylesheet" type="text/css" href="scripts/user-fonts.css" />
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/
jquery/1.11.1/jquery.min.js">///</script>
<script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/
jqueryui/1.11.0/jquery-ui.min.js">///</script>
<script type="text/javascript" src="scripts/typeahead.bundle.js">///</
script>
<script type="text/javascript" src="scripts/queryString.js">///</script>
<script type="text/javascript" src="scripts/dragdealer.js">///</script>
<script type="text/javascript" src="scripts/lunr.js">///</script>
<script type="text/javascript" src="scripts/lunrScripts.js">///</script>
<script type="text/javascript" src="scripts/loadContent.js">///</script>
...
```

When you next open your webpage, you should see a side drawer when the window size is less than 600 pixels wide.

Adding a Manifest File to your BWH


How to add a webapp manifest file to your Better Web Help.

A webapp manifest tells your webpage how to cache pages. Based on how it is used, it can allow your website to be accessed offline. There is already a manifest file built for BWH. All you need to do is modify it to include any additional resources.

1. The manifest file is located in the root directory of your webpage (bwh/ on the github repository) and is called webapp.manifest.
2. Under the "CACHE:" heading, you will see a long list of files. Here, add any additional files your page requires (such as files in the heading/ folder or additional scripts or images)
3.  **Note:** You shouldn't include all the content files here since you have to write each file out individually. That will be covered in the next step.
4. Under "NETWORK:" you see an asterisk. This means that the manifest will cache all other pages as they are encountered. This will cover all the content pages.
5. Then "FALLBACK:" says that if there is any page that isn't cached yet, it should load fallback.html. This file simply states "Oops, this files isn't cached yet, sorry!"
6. Once your manifest is built. You need to add it to the webpage. Change the opening <html> tag to the following:

```
<html manifest='webapp.manifest'>
```

When you next load the webpage if you have the developer tools open, you will see the manifest checking for each of the files listed (written out in the console).

 **Important:** If you alter any of the files listed in the manifest, they will not update on the website until the manifest itself is updated. Simply update the version line to reload all cached items.

BWH Styling

How the styling has been built for the Better Web Help DITA OT plugin.

The Better Web Help styling was built to be adaptive for mobile, tablet, and desktop. This was accomplished by adapting the frameless framework. To make the development simpler, we used LESS for CSS and compiled it before running the DITA OT.

The base styling is provided by the frameless.css stylesheet, compiled by frameless.less (both located in com.jorsek.bwh/resource/scripts/). This provides the basic styling for the webpage and for the drag-up nav panel on mobile.

This can be overwritten by side-drawer.css (as explained in BWH_Customization.pdf) which will change the nav panel to drag out from the left side.

Additionally, the user can add their own overwrite files. We have provided two example template files to get started: user-fonts.css and user-styling.css. These are automatically included in the <head> on build from the DITA OT, so you can just go ahead and edit them to change the styling.



Note: If you do change the user-fonts.css or user-styling.css files, you can safely rerun the DITA OT build script to the same folder and it will not overwrite your customizations.

Follow the examples to get more comfortable with how these customization files can be put to use.

Custom Fonts with Better Web Help

How to customize the fonts with output from the Better Web Help (bwh) DITA OT plugin.

This tutorial assumes that you have already built the website using the DITA OT.

1. Open the output directory that you supplied to the DITA OT.
2. Open "scripts/user-fonts.css."
3. You will see a comment at the top describing the classes and ids to specify to change a specific region:

```
/*
 *
 * Enter custom fonts below. Use the following key to change the specific
 * regions you require:
 *
 * #web-help-c1 {this will affect the navigation panel / table of contents}
 * #web-help-c2 {this will affect the entire content pane, excluding the
 * heading}
 * #web-help-c2 h1 {this will affect the title of the content pane}
 * #web-help-c2 .body {this will affect the body text of the content pane}
 * #web-help-c2 .related-links {this will affect the related links footer in
 * (the content pane}
 *
 * not recommended --> #heading {this will affect the heading (content from
 * heading.html)}
 *   edit styles within heading.html instead
 *
 */
```

4. Using the syntax below, enter the desired fonts.

```
{region} {font-family: {font1}, {font2}, {font3}, ...;}
```

The fonts on your web-help should now be changed. Note that if you rebuild the project to the same output directory, your custom fonts will remain. The build will not overwrite your user-fonts.css file.

BWH user-styling.css Width Example

An example on how to change the width of the columns in the Better Web Help DITA OT plugin output.

In the default output of the DITA OT, the widths of the columns are percent based. They change size depending on the size of the window. In this example, we will fix the width of the Navigation panel and adjust the size of the content pane accordingly.

1. Say you want to change the widths of the Navigation column and the Content column for your version of BWH.
2. Open the user-styling.css file located in bwh/scripts/.
3. The first comment gives you the IDs and classes of important regions of the document. The only ones we're going to worry about in this example are

```
#web-help-c1 {this will affect the navigation panel / table of contents}
#web-help-c2 {this will affect the entire content pane, excluding the
  heading}
```

4. If you scroll down in user-styling.css, you will see the templates for the different media (mobile, wide-mobile, tablet, desktop, and desktop zoom).
5. We don't want to affect the widths on mobile because the Navigation panel becomes a drawer, so under "desktop layout" ("min-width: 57em") let's add a line to fix the width of the Nav panel:

```
/*
 * desktop layout
 * 912-infinity px
 */
@media screen and (min-width: 57em) {
  #web-help-c1 {width: 15.1875em;}
}
```

6. If you are still using the default heading (with the easyDITA logo) you will notice that the panel is now the same width as the logo. We computed this by opening the developer tools on the website to measure the width of the logo (248px) minus the padding and margins on #web-help-c1 (5px) which gives us 243px. Then, since the body font size is 16px, $243\text{px}/16\text{px}/1\text{em} = 15.1875\text{em}$.
7. Now, we also want fixed width for tablet, but we want it a little smaller, so we can add the following line of code:

```
/*
 * Tablet layout
 * 600-911 px
 * Zoomed in above 600 px
 */
@media screen and (min-width: 37.5em) {
  #web-help-c1 {width: 10.625em;}
}
```

8. However, now we need to adjust the width of the content pane (#web-help-c2) to match the navigation panel.

```
/*
 * Tablet layout
 * 600-911 px
 * Zoomed in above 600 px
 */
@media screen and (min-width: 37.5em) {
  #web-help-c1 {width: 10.625em;}
  #web-help-c2 {left: 13em; right: 0; width: auto;}
}

/*
 * desktop layout
 * 912-infinity px
 */
@media screen and (min-width: 57em) {
  #web-help-c1 {width: 15.1875em;}
```

```
#web-help-c2 {left: 18em; right: 0; width: auto;}  
}
```

9. We rounded up on the left values so that there is a little extra padding between the navigation panel and the content pane.

At this point, you should have a constant width navigation panel when resizing your window and the content pane will resize appropriately.