



ONC SITE DROPDOWN MENU COMPONENT

Design Document

Abstract

Design document for the page header component used in ONC SPO sites for all the modern style pages, to be consistent with the current site.

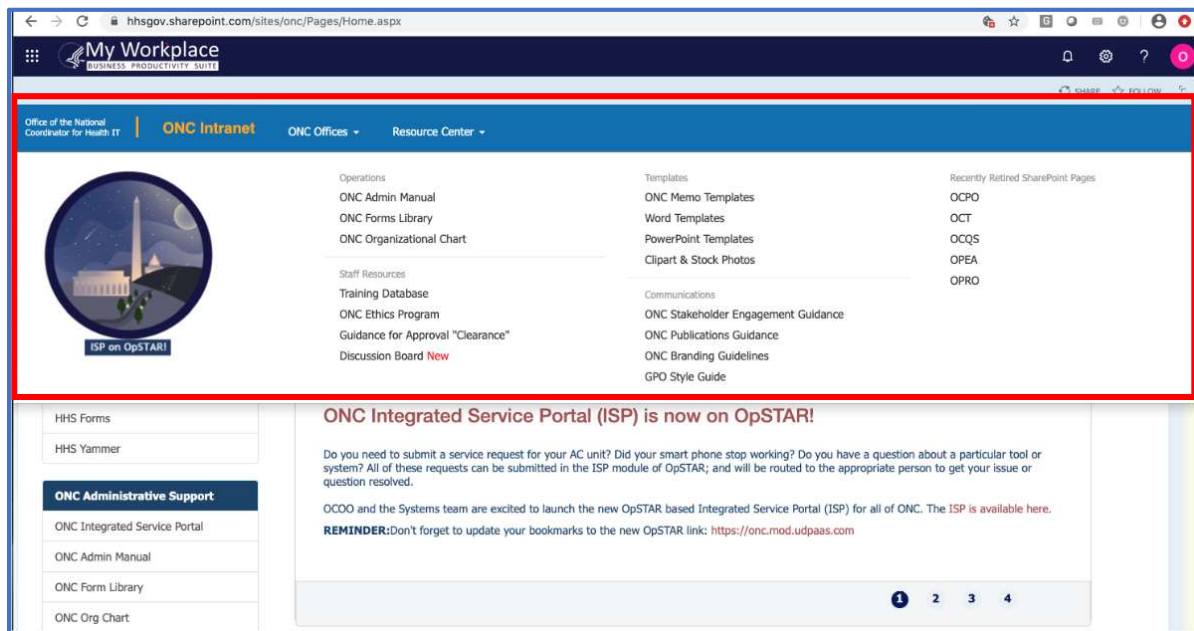
ONC Site Administrator
farrah.darbouze@hhs.gov

TABLE OF CONTENTS

Goal	2
Design.....	3
Implementation Plan	4
Tools and Software Packages.....	5
Build, Test and Deployment.....	6
Maintenance and Change Control	7

GOAL

In ONC SPO site, we have a dropdown menu on top of all the pages (as highlighted below):



This mega dropdown menu is implemented inside the site master page.

Recently when we are going through our site redesign, we want to use the more appealing modern style pages made available by Microsoft. However, master page or layout pages are currently no available in the modern style sites.

Our goal is to implement a new header dropdown menu component that utilizes the new SharePoint Framework to display the current ONC dropdown menu on all the modern style pages.

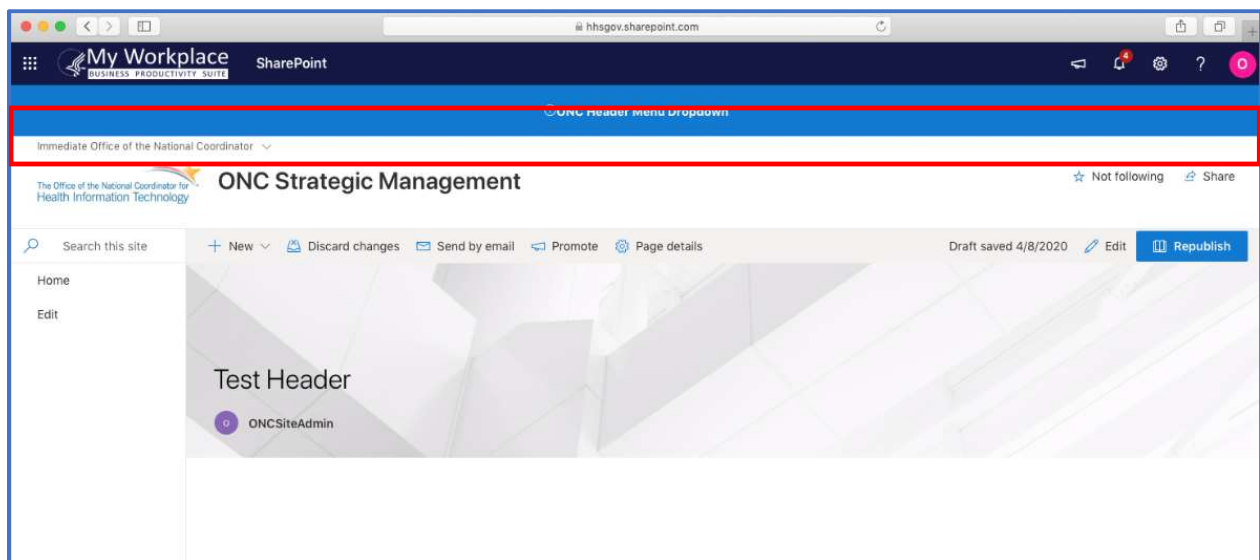
DESIGN

SharePoint Framework Extensions allows user to use the familiar SharePoint Framework tools and libraries for client-side development to extend the SharePoint the modern pages and document libraries. Specifically, SharePoint Framework Extensions type of Application Customizers, allows user to customize the header (PlaceholderName .Top) and footer (PlaceholderName .Bottom) of all modern pages in the site with the extensions application applied to.

In this design, we choose to use Application Customizers load:

1. custom scripts: handle dropdown menu behaviors;
2. third-party javascript libraries: include jquery and bootstrap;
3. styles: include bootstrap, fontawesome, and inline custom css;
4. and, custom HTML elements: display the dropdown menu contents.

And render the content in the placeholder “PlaceholderName .Top”:



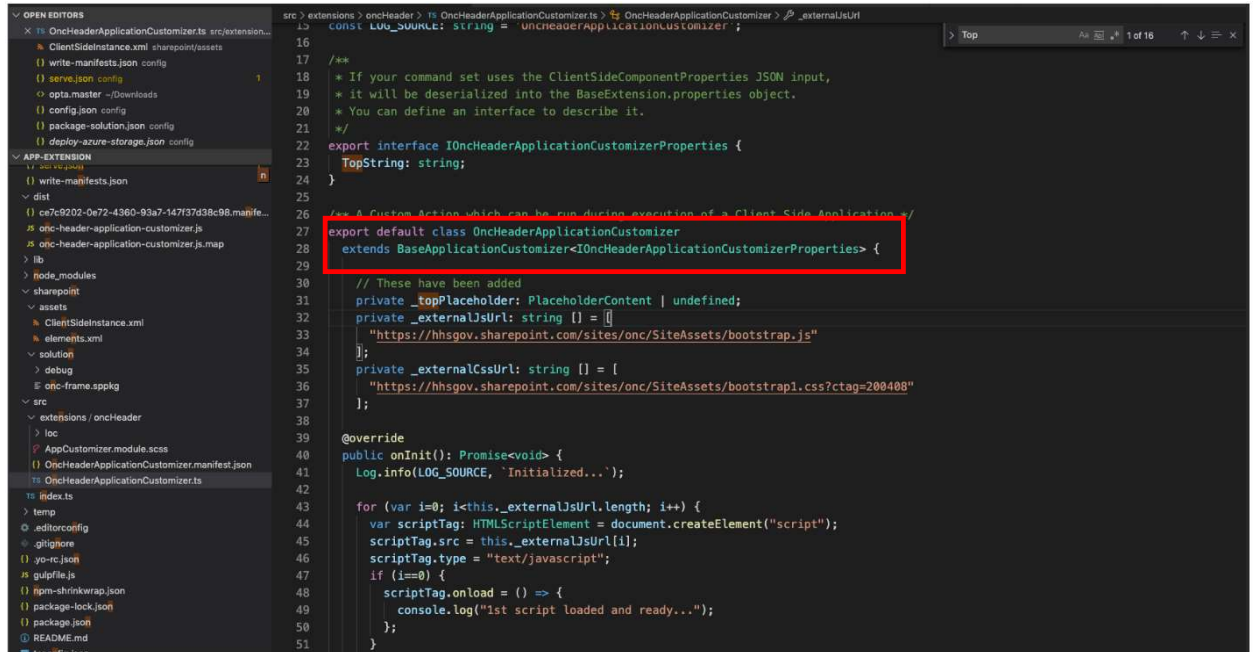
Lastly, Application Customizer extensions are supported with Site, Web, and List scopes. And we are planning to use it on the site level by associating the application customizer to the UserCustomAction collection in Site object level.

REFERENCE: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/using-page-placeholder-with-extensions>

IMPLEMENTATION PLAN

The implementation plan is:

1. Extract and assemble all necessary code, library, css and HTMLs from the current site for the dropdown menu so that we can have an independent component ready to be injected to the new site;
2. Create the SharePoint Framework Extension project for Application Customizer;
3. Extends the BaseApplicationCustomizer class to handle the Header placeholder rendering, and injects all the artifacts prepared in step #1;



```
16
17
18 /**
19  * If your command set uses the ClientSideComponentProperties JSON input,
20  * it will be deserialized into the BaseExtension.properties object.
21  * You can define an interface to describe it.
22  */
23 export interface IOncHeaderApplicationCustomizerProperties {
24   TopString: string;
25 }
26
27 /** A Custom Action which can be run during execution of a Client Side Application */
28 export default class OncHeaderApplicationCustomizer
29 extends BaseApplicationCustomizer<IOncHeaderApplicationCustomizerProperties> {
30
31   // These have been added
32   private _topPlaceholder: PlaceholderContent | undefined;
33   private _externalJsUrl: string[] = [
34     "https://hhsgov.sharepoint.com/sites/onc/SiteAssets/bootstrap.js"
35   ];
36   private _externalCssUrl: string[] = [
37     "https://hhsgov.sharepoint.com/sites/onc/SiteAssets/bootstrap1.css?ctag=200408"
38   ];
39
40   @override
41   public onInit(): Promise<void> {
42     Log.info(LOG_SOURCE, 'Initialized...');
43
44     for (var i=0; i<this._externalJsUrl.length; i++) {
45       var scriptTag: HTMLScriptElement = document.createElement("script");
46       scriptTag.src = this._externalJsUrl[i];
47       scriptTag.type = "text/javascript";
48       if (i==0) {
49         scriptTag.onload = () => {
50           console.log("1st script loaded and ready...");
51         };
52       }
53     }
54   }
55 }
```

4. Test the application;
5. Deploy the application.

NOTE: Per the Microsoft document online, to deploy the client-side application needs to be deployed in an application catalog site which we do not have privilege to operate on. That's where we need support from the tenant administrator.

TOOLS AND SOFTWARE PACKAGES

As SharePoint recommended, for our development tools, we are using:

1. NodeJS for application coding;
2. Yeoman for project/framework generating
3. Visual Code for IDE; and,
4. Gulp for build

To reduce development efforts to the minimum, we want to reuse as much code from the current dropdown menu as possible. Some of the third-party libraries/styles used there include:

1. jquery,
2. bootstrap, and
3. fontawesome.

BUILD, TEST AND DEPLOYMENT

Once the SharePoint project is setup with Yeoman generator, we should be able to do:

1. build and test with command:

gulp serve

2. to build a deployable package:

gulp bundle

gulp package-solution

3. to deploy the package (by our tenant administrator):
 - a. ONC developer will send the deployable client-side solution package (onc-solution.sppkg) to HHS site administrator.
 - b. Administrator Upload or drag-and-drop the client-side solution package to the app catalog in the tenant, and then select the Deploy button. Notice the assets associated with this project will be automatically hosted with Office 365 CDN, hence our Office 365 CDN should be turned on (see <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/hosting-extension-from-office365-cdn>).
 - c. Add the customer solution to required ONC office sites.

MAINTENANCE AND CHANGE CONTROL

The contents header dropdown menu component is rather static. We'll have some of the content pulling from our current site. This allows us to reduce the needs of updating the component while maintain control of the displayed information in the meantime. Use other words, once test and deployed, the component should not require much maintenance.

Nevertheless, there can be requirements for major update of the component. In that case, we'll follow the current build, test and deployment procedure to redeploy the component.