# Office AMS: UI Element Personalization

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This application sample demonstrates how to personalize UI elements using JavaScript injection, user profile and SharePoint lists. It also uses HTML5 local storage to keep the number of calls to target SharePoint services to a minimum. | * Office 365 Multi-Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Branding.UIElementPersonalization, version 1.0 |
| Author: | Brian Michely, Vesa Juvonen, Bert Jansen, Frank Marasco (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

## Version log

Change log between different releases.

|  |  |
| --- | --- |
| Version | Notes |
| 1.0 | Initial |

# General comments

This sample shows how one can personalize elements within the UI by the use of some javascript injection (derives from the javascript injection sample), as well as using values from the user profile and SharePoint lists. This example also shows the use of some HTML5 local storage to reduce round trips to target services.



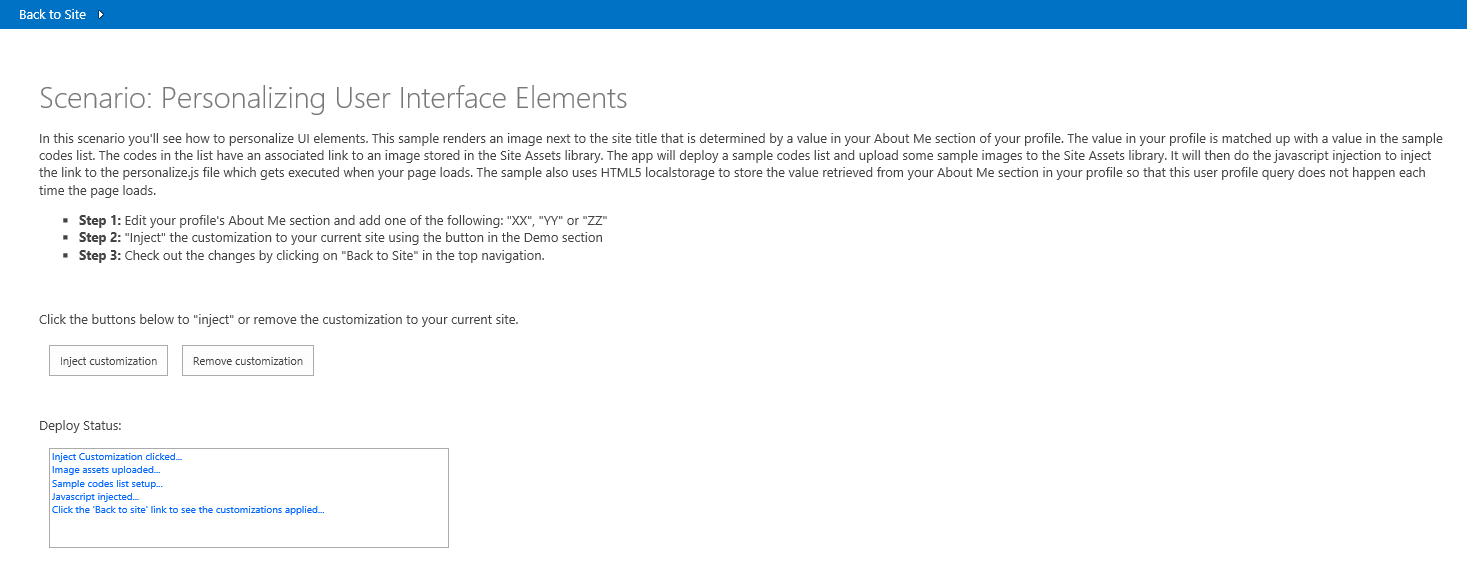
# SCENARIo: UI ELEMENT PERSONALIZATION

The app page does a few things up front to support the sample. It uploads three images into the Site Assets library as well as creates a list named “CodesList”, and then creates three list items that contains a title and a URL to one of the three images that were uploaded to the Site Assets library. The list items are titled as “XX”, “YY” and “ZZ” as fictitious business unit codes.

It is assumed the user has one of these three codes listed in their ‘About Me’ section of their profile.

The last task the app page performs, is the javascript injection.

Once these tasks are done, the user clicks the ‘Back to Site’ link, and their site will load, and that when the injected javascript kicks in and check local storage for a saved profile value. If it does not exist, or is expired, the javascript will query the user profile and look for one of the codes above. It then stores it into local storage and queries the CodesList to find the matching code’s image URL to the Site Assets library. If it is found, the image will be rendered in the Head section within the Page Title area



Code for the Inject Customization button:

/// <summary>

protected void btnSubmit\_Click(object sender, EventArgs e)

{

status.Items.Clear();

status.Items.Add("Inject Customization clicked...");

var spContext = SharePointContextProvider.Current.GetSharePointContext(Context);

using (ClientContext clientContext = spContext.CreateUserClientContextForSPHost())

{

// Upload the assets to host web

UploadAssetsToHostWeb(clientContext.Web);

status.Items.Add("Image assets uploaded...");

// Setup sample codes list for demo use only

SetupCodesList(clientContext, clientContext.Web);

status.Items.Add("Sample codes list setup...");

// Inject the JsLink

AddPersonalizeJsLink(clientContext, clientContext.Web);

status.Items.Add("Javascript injected...");

status.Items.Add("Click the 'Back to site' link to see the customizations applied...");

}

}

**Javascript Injection:**

SharePoint team sites by default make use of the Minimal Download Strategy (MDS) technique to improve performance. If we want to load custom JavaScript files we have to take this in account by loading the scripts via the below pattern:

// Register script for MDS if possible

RegisterModuleInit("personalize.js", RemoteManager\_Inject); //MDS registration

RemoteManager\_Inject(); //non MDS run

if (typeof (Sys) != "undefined" && Boolean(Sys) && Boolean(Sys.Application)) {

Sys.Application.notifyScriptLoaded();h

}

if (typeof (NotifyScriptLoadedAndExecuteWaitingJobs) == "function") {

NotifyScriptLoadedAndExecuteWaitingJobs("scenario1.js");

}

When the page that contains your script is loaded either the MDS engine (when MDS is enabled) launches your main function (RemoteManager\_Inject) or your function is launched directly for non MDS invocations. The function that’s called is your entry point to load other scripts and to perform the required customizations. Loading other scripts often is needed: the sample shows how you can load the popular jQuery library. When loading other scripts it’s important that the script parts that depend on the loaded script are only executed after the other script was loaded and this is guaranteed via the below construct:

function RemoteManager\_Inject() {

var jQuery = "https://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.0.2.min.js";

// load jQuery

loadScript(jQuery, function () {

personalizeIt();

});

}

The personalizeIt() function:

function personalizeIt() {

clientContext = SP.ClientContext.get\_current();

var fileref = document.createElement('script');

fileref.setAttribute("type", "text/javascript");

fileref.setAttribute("src", "/\_layouts/15/SP.UserProfiles.js");

document.getElementsByTagName("head")[0].appendChild(fileref);

SP.SOD.executeOrDelayUntilScriptLoaded(function () {

// Get localstorage values if they exist

buCode = localStorage.getItem("bucode");

buCodeTimeStamp = localStorage.getItem("buCodeTimeStamp");

// Check to see if the page already has injected personalized image

var pageTitle = $('#pageTitle')[0].innerHTML;

if (pageTitle.indexOf("img") > -1) {

personalized = true;

}

else {

personalized = false;

}

// If nothing in localstorage, get profile data, which will also populate localstorage

if (buCode == "" || buCode == null) {

getProfileData(clientContext);

personalized = false;

}

else {

// Check for expiration

if (isKeyExpired("buCodeTimeStamp")) {

getProfileData(clientContext);

if (buCode != "" || buCode != null) {

// Set timestamp for expiration

currentTime = Math.floor((new Date().getTime()) / 1000);

localStorage.setItem("buCodeTimeStamp", currentTime);

// Set personalized to false so that the code can check for a new image in case buCode was updated

personalized = false;

}

}

}

// Load image or make sure it is current based on value in AboutMe

if (!personalized) {

loadPersonalizedImage(buCode);

}

}, 'SP.UserProfiles.js');

}

There is some functionality to manage local storage key expiration as this is not built into HTML local storage. This function will check to see if the key is expired for example.

// Check to see if the key has expired

function isKeyExpired(TimeStampKey) {

// Retrieve the example setting for expiration in seconds

var expiryStamp = localStorage.getItem(TimeStampKey);

if (expiryStamp != null && cacheTimeout != null) {

// Retrieve the timestamp and compare against specified cache timeout settings to see if it is expired

var currentTime = Math.floor((new Date().getTime()) / 1000);

if (currentTime - parseInt(expiryStamp) > parseInt(cacheTimeout)) {

return true; //Expired

}

else {

return false;

}

}

else {

//default

return true;

}

}

# Dependencies

* Microsoft.SharePoint.Client.dll
* Microsoft.SharePoint.Client.Runtime.dll