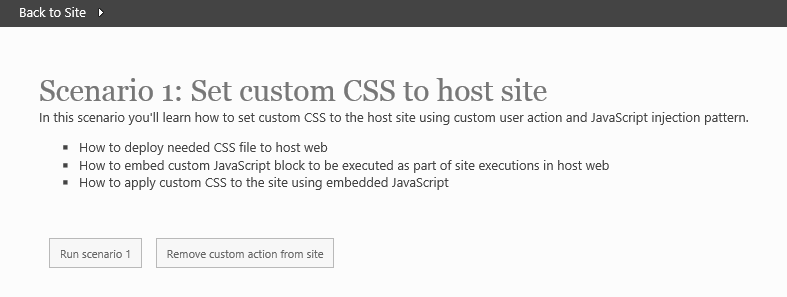
# Office AMS: CSS injection pattern

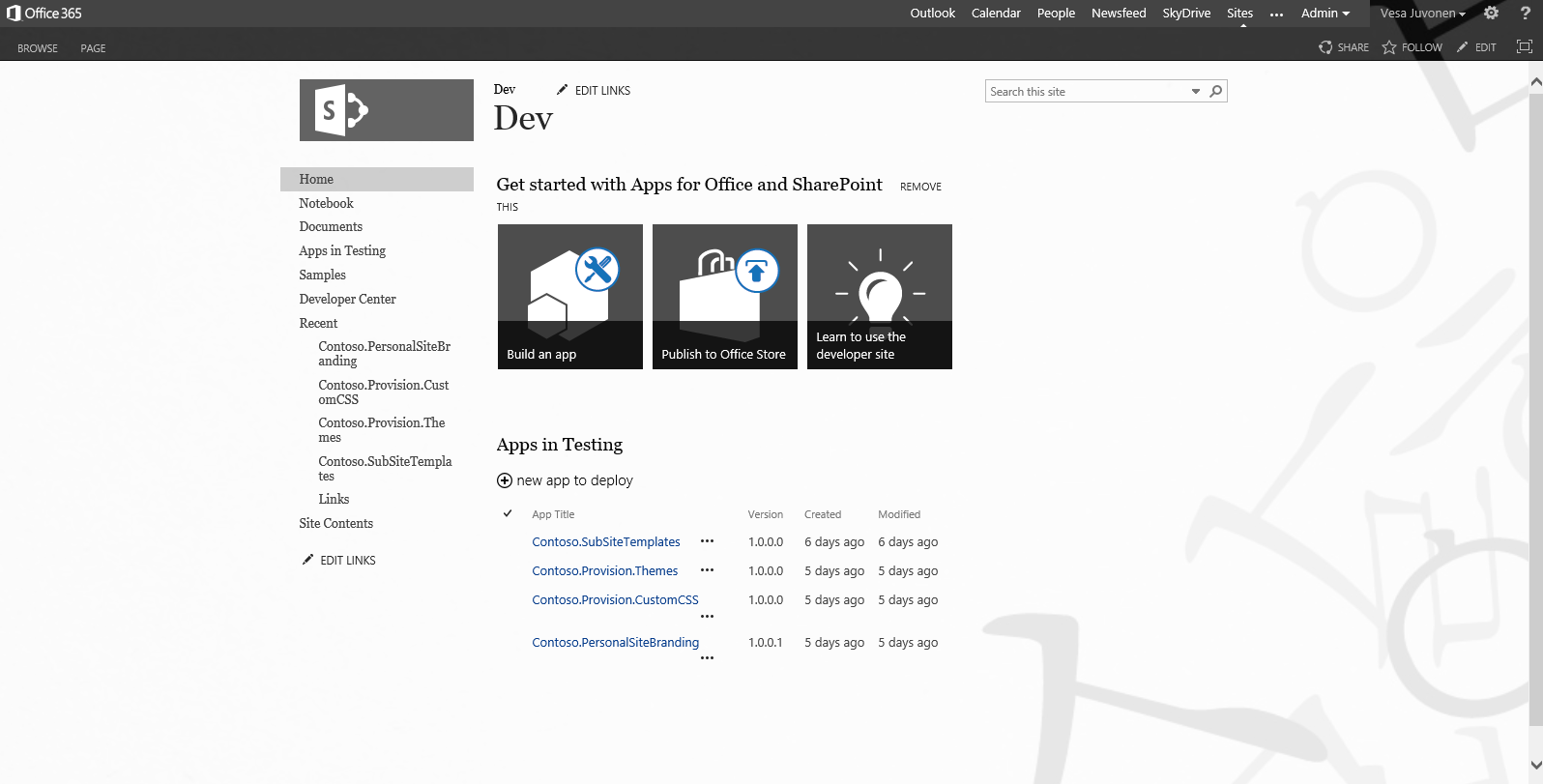
|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample shows how to inject custom CSS to the host web using SP App | * Office 365 Multi Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Branding.CustomCSS, version 1.0 |
| Author: | Vesa Juvonen (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // 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.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

# Scenario 1: Inject custom CSS from App to host web

This scenario shows how to modify host web branding or rending by injecting custom CSS on the site using JavaScript injection pattern. This is one way to modify the host web branding without providing a custom master page on the site.



Once the modification has been injected, we can see the host web to have centralized alignment.



## Injecting CSS to the host web

Injection of the CSS is done by using so called JavaScript injection pattern, which gives us opportunity to add custom JavaScript to be executed when pages in the sites are viewed. This pattern is based on usage of user custom actions, which we can add to the host web by using CSOM as follows.

// Build a custom action to write a link to our new CSS file

UserCustomAction cssAction = web.UserCustomActions.Add();

cssAction.Location = "ScriptLink";

cssAction.Sequence = 100;

cssAction.ScriptBlock = @"document.write('<link rel=""stylesheet"" href=""" + assetLibrary.RootFolder.ServerRelativeUrl + @"/contoso.css"" />');";

cssAction.Name = actionName;

// Apply

cssAction.Update();

clientContext.ExecuteQuery();

This means that on each page load in the site, we will also execute the assigned JavaScript, which will take care of the CSS injection to the page.

We’ll obviously need to also upload the CSS file to the host web as well, unless we want to reference that from somewhere else. CSS file can be uploaded to any location in the host and even though in this example it’s uploaded to the *Site Assets* library, which might not be the best option, since it could be accidently deleted. Better option would be to use simply for example custom folders, which are not visible by using browser.

List assetLibrary = web.Lists.GetByTitle("Site Assets");

clientContext.Load(assetLibrary, l => l.RootFolder);

// Get the path to the file which we are about to deploy

string file = System.Web.Hosting.HostingEnvironment.MapPath(string.Format("~/{0}", "CSS/contoso.css"));

// Use CSOM to uplaod the file in

FileCreationInformation newFile = new FileCreationInformation();

newFile.Content = System.IO.File.ReadAllBytes(file);

newFile.Url = "contoso.css";

newFile.Overwrite = true;

Microsoft.SharePoint.Client.File uploadFile = assetLibrary.RootFolder.Files.Add(newFile);

clientContext.Load(uploadFile);

clientContext.ExecuteQuery();

## Removing the custom action from host web

Removal of the custom action can be done by accessing the *UserCustomActions* collection of the host web and removing the reference by using action name as follows:

Web web = clientContext.Web;

string actionName = "ContosoCSSLink";

// Clean up existing actions that we may have deployed

var existingActions = web.UserCustomActions;

clientContext.Load(existingActions);

clientContext.ExecuteQuery();

// Clean up

foreach (var existingAction in existingActions)

{

if (existingAction.Name.Equals(actionName, StringComparison.InvariantCultureIgnoreCase))

existingAction.DeleteObject();

}

clientContext.ExecuteQuery();