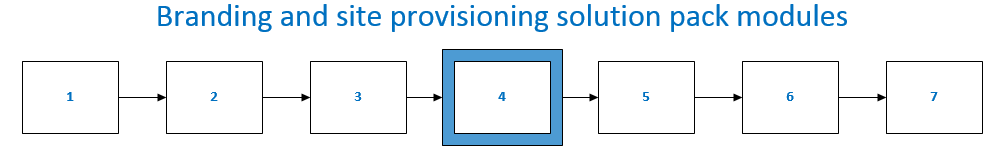
Module 4: Composed looks, themes, and SharePoint CSS

# About composed looks, themes, and SharePoint CSS

Module 4 builds on your knowledge of the SharePoint page model, design tools, and design packaging options presented in Modules 2 and 3. Here, you’ll learn about options for customizing the look and feel of a SharePoint site using composed looks, the new SharePoint 2013 theming engine, and CSS using the cloud app model (CAM) and CSOM APIs.



This module:

* Defines key terms and concepts related to composed looks, themes, and SharePoint CSS.
* Provides an **overview** of SharePoint branding options, from composed looks and themes to custom CSS and custom master pages and content pages.
* Describes the **theming engine** and its moving parts, including an example custom theme.
* Introduces **CSS** and specific points about CSS to keep in mind when working with SharePoint.
* Includes code samples that demonstrate how to **apply themes** to SharePoint sites using the remote provisioning pattern.

**Important** Download the [Branding and site provisioning solution pack’s sample pack](http://code.msdn.microsoft.com/SharePoint-2013-Sample-81b03d1e) to get samples that demonstrate how to use the remote provisioning pattern to apply themes, work with custom CSS, and more.

# Key terms and concepts

**Table 1. Key terms and concepts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Term** | **Definition** | **Default file location or how to access through the UI** | **Guidance** |
| Alternate CSS | A CSS file other than the default that you can apply to the look and feel of your site. | File is not stored locally. | Use alternate CSS to apply custom CSS to a site and all of its subsites. |
| Cascading Style Sheets | A language that tells a browser how to render an HTML or XML document’s styles. CSS separates document content (HTML or XML) from how the content is presented. |  |  |
| Composed Look | A combination of fonts, a color palette, a background image, and an associated master page that are applied to the site. Font scheme and color images are optional. | Theme Gallery\15 folder | Composed looks are a convenient way to change the look and feel of sites without having to make any changes to the structure of a site.  SharePoint 2013 ships several composed looks by default. When a user applies a composed look, SharePoint applies all of the associated design elements of the composed look to a site. |
| Content search web part (CSWP) | Renders content from search results based on a specified query. |  | [Content Search Web Part](http://social.technet.microsoft.com/wiki/contents/articles/15843.content-search-web-part-in-sharepoint-2013.aspx) (TechNet)  [Content Search Web Part](http://msdn.microsoft.com/en-us/library/jj163789.aspx) (MSDN) |
| corev15.css | CSS file that contains most of the main functionality for SharePoint. | \_layouts\15 folder | The main CSS file in SharePoint 2013. |
| CSS | See Cascading Style Sheets. |  |  |
| CSSRegistration | A reference in a master page, such as seattle.master, that loads most CSS that is applied to most of the default UI. | .master page | Use the **CSSRegistration** control in a master page override default CSS. |
| [Custom action](http://msdn.microsoft.com/library/office/jj163954.aspx) | Actions you can use to customize and interact with lists and the ribbon on the host web. |  | [How to: Create custom actions to deploy with apps for SharePoint](http://msdn.microsoft.com/en-us/library/office/apps/jj163954.aspx) |
| [Device channels](http://msdn.microsoft.com/en-us/library/jj862343.aspx) | Render a single SharePoint Publishing site in more than one way by using unique channels to target content rendering on specific devices. |  | See also “user agent string.”  For use with SharePoint 2013 Publishing sites. |
| [Display templates](http://msdn.microsoft.com/en-us/library/jj945138.aspx) | Templates used by Search Web Parts to show the results of a query made to the search index. |  | For use with SharePoint 2013 Publishing sites. |
| [Image rendition](http://msdn.microsoft.com/en-us/library/jj720398.aspx) | Display differently sized versions of an image on a Publishing site based on the same source image. |  | For use with SharePoint 2013 Publishing sites. |
| [Managed metadata](http://msdn.microsoft.com/en-us/library/jj163949.aspx) | Sometimes referred to as taxonomy, the managed metadata features in SharePoint 2010 and 2013 enable you to define terms, term sets, groups, and labels for terms. In SharePoint 2013, the managed metadata system is the foundation for managed navigation. |  |  |
| [Managed navigation](http://msdn.microsoft.com/en-us/library/jj163978.aspx) | Navigation for publishing sites that is built based on managed metadata. Navigation is built from a specified term set in the term store. |  |  |
| [Master page](http://msdn.microsoft.com/en-us/library/office/dn205271.aspx) | A page that standardizes the behavior and presentation the left navigation and top navigation areas of a SharePoint page. |  |  |
| [Master Page Gallery](http://msdn.microsoft.com/en-us/library/dn205271.aspx) | The Master Page Gallery is a special document library in SharePoint 2013 where all branding elements—master pages, page layouts, JavaScript files, CSS, and images—are stored by default. Every site has its own Master Page Gallery. |  | The Master Page Gallery contains catalogs that store branding assets such as master pages and CSS files.  When creating custom branding elements, it’s a best practice to store custom assets in the default Master Page Gallery file structure. |
| Oslo.master | A master page in SharePoint 2013. |  | Moves the current navigation into the same position as the top navigation region. |
| Page content control | A control on a publishing site where a Web Part can be added. |  |  |
| [Page Layout](http://msdn.microsoft.com/en-us/library/office/dn205271.aspx) | A template for a SharePoint Publishing site page that enables users to lay out information on the page in a consistent way. |  | [How to: Create a page layout in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj822368.aspx) |
| Quick Launch | Manages the navigation elements on the left side of the page of a collaboration site. |  | You can add heading links to group navigation items. |
| [REST](http://msdn.microsoft.com/library/office/fp142380.aspx) | REpresentional State Transfer. A stateless architectural style that abstracts architectural elements and uses HTTP verbs read and write data from Web pages that contain XML files. |  |  |
| Root web | The first web inside of a site collection. |  | The root web is also sometimes referred to as the “Web Application Root”. |
| Seattle.master | The default .master page for SharePoint 2013 Team sites and Publishing sites. |  |  |
| Site layout | See master page. |  | The site layout combined the .master page of a theme with its corresponding .preview file. |
| Structured navigation | A navigation structure for publishing sites that is based on the site hierarchy of the publishing site. This approach was originally introduced in SharePoint Server 2010 and is unchanged in SharePoint 2013.  You can add headers and links to manually replace or customize the structured navigation that SharePoint automatically generates. |  | [How to: Customize navigation in SharePoint Server 2010 (ECM)](http://msdn.microsoft.com/en-us/library/office/ms558975(v=office.14).aspx) |
| [Theme](http://msdn.microsoft.com/en-us/library/office/jj927174.aspx) | A simple way to apply light branding to a SharePoint site. | *\_themes* folder of the site. | Themes are the first recommended course of action for applying custom branding to SharePoint sites in SharePoint 2013 and in SharePoint Online.  [How to: Deploy a custom theme in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj927175.aspx) |
| Theming engine | Set of files and functionality that define the look, feel, behavior, and file associations of composed looks. |  | “Composed Look” is the new name for “theme” in SharePoint 2013 and SharePoint Online. |
| User Agent String | Information that a browser passes to a website that identifies the software that makes the request from the server. |  | [Device channels](http://msdn.microsoft.com/en-us/library/jj862343.aspx) |
| User Custom Action | A CSOM property that returns the collection of custom actions for a website, list, or site collection. | 15\TEMPLATE\FEATURES  Example: Hiding the Themes link under Galleries  <HideCustomAction  GroupId="Galleries"  HideActionId="Themes"  Location="Microsoft.SharePoint.SiteSettings"> </HideCustomAction> | [UserCustomAction](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.usercustomaction.aspx)  [How to: Create custom actions to deploy with apps for SharePoint](http://msdn.microsoft.com/en-us/library/jj163954.aspx)  [How to: Work with user custom actions](http://msdn.microsoft.com/en-us/library/office/ee538686(v=office.14).aspx) (SharePoint 2010)  [Default custom action locations and IDs](http://msdn.microsoft.com/en-us/library/office/bb802730(v=office.14).aspx) |

# Look and feel in SharePoint

SharePoint 2013 includes many out-of-the-box features that you can use to customize the look and feel of a SharePoint site. This section covers two basic approaches:

* Designers can use the **theming engine** to create custom themes, which are called composed looks in SharePoint 2013 and SharePoint Online. At a minimum, themes define colors. A complete theme defines colors, fonts, a background image, and the associated master page, and a .preview file that defines how the .master page is previewed. Developers can write remote provisioning code to apply themes using the remote provisioning pattern.
* Designers can create **custom CSS** to apply to SharePoint sites. Using the remote provisioning pattern, a developer can use an app for SharePoint to provision SharePoint sites to use custom CSS.

## Branding cost and complexity

Branding changes range from low-cost and simple to high-cost and complex. Through the UI, users can apply *composed looks*, which include a background image, color palette, fonts, a master page, and an associated .preview file for the master page. You can use the SharePoint 2013 theming engine to design composed looks and provision sites, and you can create custom CSS to modify the look and feel of your site and its elements.

**Important** While it’s possible to create custom master pages and other structural elements as part of a custom branding project, the long-term cost of supporting structural customizations is high, and may make it more costly for your organization to apply upgrades and provide ongoing support for customizations.

# Composed looks

The SharePoint 2013 theming engine renders composed looks, which are the out-of-the-box themes included in SharePoint 2013 and SharePoint Online. As a designer or developer, you can use the theming engine to apply custom composed looks.

Apply a composed look to a SharePoint site by navigating to **Site Settings > Look and Feel > Change the look.** You can customize can apply looks composed of colors, fonts, master page, and a background image using the **Change the look** wizard. If you want to change an image, you can click **Change**, click **Try It**, and if you like it, click **Keep It**.

The **Change the Look** wizard copies, transforms, and stores CSS in SharePoint’s content database. It also recolors images and stores them in the content database.

# The Theming engine

The SharePoint 2013 theming engine makes it possible to apply colors, fonts, and a background image to a site by associating those elements with a master page.

In SharePoint 2013 and SharePoint Online, a theme is a connected set of XML definition files, an image file, and an associated master page that you can use to apply custom CSS to a site. The XML files--.spcolor and .spfont—define *color slots* and *font slots* that define the details of specific colors and fonts as they’re applied to styles. You can create your own color and font files in your favorite text editor.

**Table 2. What’s in a composed look?**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component** | **Files** |  | **Where it’s stored** | **Required?** |
| Color palette | .spcolor |  | Theme Gallery\15 folder | x |
| Font scheme | .spfont |  | Theme Gallery\15 folder |  |
| Site layout | .master, .preview |  | Master Page Gallery | x |
| Background image | .jpg, .bmp, .png, .gif |  | Site assets |  |

Figure 1 shows the logical relationship between ways users can select and apply composed looks through the UI and the underlying theming engine files that define theme data and associate them with the master page. Users can select composed looks through the **Site Settings > Look and Feel >** **Change the Look**, the Getting Started UI, or directly in the site actions menu. When a user selects a composed look, the theming engine applies colors, fonts, background images, the associated .master page, and the .preview file associated with the .master page to the site.

The following sections explain the moving parts of the theming engine in more detail.

### Color palettes

The theming engine stores colors in color palettes defined by the .spcolor file. Color palettes are stored in the **Theme Gallery** of the root site. A *color palette* is an editable XML file made up of color palette definitions and *color slots*. Color palette metadata (<s:colorPalette>) defines three preview slots, an isInverted property that lets the palette designer specify whether the theme is inverted (dark background with light text), and the XML namespace associated with the theme. The three preview slots define what color slots to use in composed look previews.

Color slots are defined by two attributes—color name and value—which define a name for the color its RGB value. Color slots have semantic names such as BodyText or SuiteBarBackground that help you identify which slots correspond to a region of a SharePoint page:

<s:color name=”BodyText” value=”444444” />

**Figure 1. .spcolor file**



Line 2 of the .spcolor defines the XML namespace, preview slots, and whether colors are inverted (they have a light foreground on a dark background instead of a dark foreground on a light background).

The .spcolor file contains 89 color slots. You can use color slots to define richer aspects of color, including opacity, by using 8-digit hexadecimal values. For example, if green is RRGGBB 00FF00, a 70% opaque green is AARRGGBB 7F00FF00. If SharePoint uses a slot that you don’t define, any CSS that references it won’t change color. If a slot is defined that is never referenced in CSS, then the color is never shown in the UI.

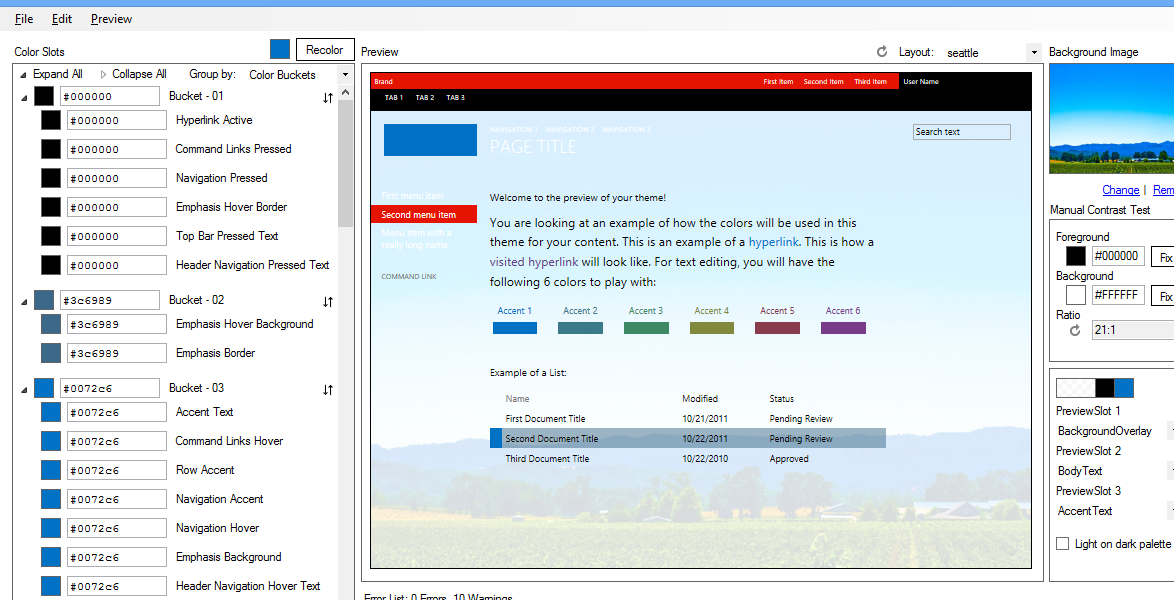
The .spcolor file is editable in Notepad; it is not editable in PowerPoint.

* + - 1. Color Palette Tool

The [color palette tool](http://www.microsoft.com/en-us/download/details.aspx?id=38182) (download) enables you to visualize theme colors and how they work together on the page. Use it to identify color information you can use in the color slots of the .spcolor file, and apply colors to a SharePoint site without changing any CSS as part of the process.

The tool displays the colors in hexadecimal format, so you can easily copy and paste the color value into the appropriate element in your .spcolor file. You can also use the color palette tool to fit a background image into a mockup and toggle between the seattle.master and oslo.master master pages.

**Figure 2. Color palette tool**



The .spcolor file is the only file that is required for a new theme, but you may need to add some custom font declarations depending on the depth of your design. To do that, you need to access the *.spfont* file.

### Font schemes

Just as color palettes define how colors are used in composed looks, *font schemes* define the fonts in composed looks.

Font schemes are defined in the .spfont file stored in the Theme Gallery. The .spfont file includes seven font slots that define the name, typeface, and script values of a composed look, as follows:

* Title
* Navigation
* Small-heading
* Heading
* Large-Heading
* Body
* Large-Body

Fonts are further scoped by script type (e.g., Latin, Arabic, Cyrillic).Web fonts support is included in four file types: embedded open type (EOT), web open font format file (WOFF), TrueType font (TTF), and scalable vector graphics (SVG).

The font scheme defines a large preview image and a small preview image. They are required only for web fonts.

Font schemes (.spfont) files are editable in Notepad, but are not editable in PowerPoint.

<?xml version=”1.0” encoding=”utf-8”?>

<s:fontScheme name=”Georgia” previewSlot1=”title” previewSlot2=”body”

xmlns:s=<http://schemas.microsoft.com/sharepoint/>>

<s:fontSlots>

<s:fontSlot name=”title”>

<s:latin typeface=”Georgia”/>

<s:font script=”Arab” typeface=”Calibri” />

<s:font script=”Deva” typeface=”Mangal” />

. . .

</s:fontSlot>

<s:fontSlot name=”navigation”>

<s:latin typeface=”Georgia”/>

<s:font script=”Arab” typeface=”Calibri” />

<s:font script=”Deva” typeface=”Mangal” />

. . .

</s:fontSlot>

</s:fontSlots>

</s:fontScheme>

### Site layout: .master pages and corresponding .preview files

The theming engine defines the site layout of a composed look based on the .master master page and its corresponding .preview file. For example, if the master page defined for the composed look is seattle.master, that master page defines the layout of the site.

The site layout is pulled from the Master Page Gallery of any master pages that have accompanying .preview files. A .preview file is required for a master page to be a selectable option in the **Change the Look** UI.

To make a master page available from the **Site Layout** drop-down menu, create a .preview file that corresponds to the .master page. The .preview file displays thumbnail images for the composed look and the preview section to the right of the **Change the Look** options on the designbuilder.aspx page.

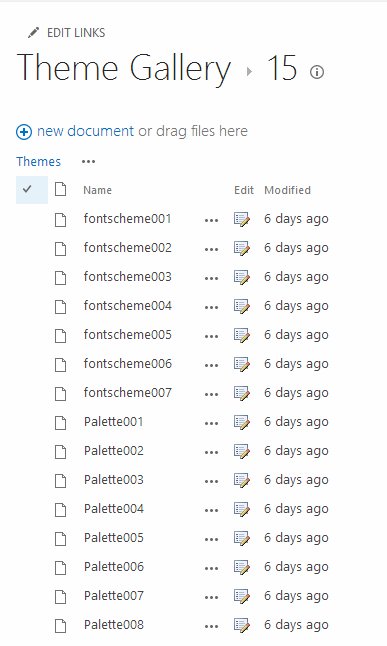
### Background image

You can change the background image of a Composed Look by clicking **Change**, which launches an upload dialog box that you can use to upload an image file. You can also drag and drop your own image onto the background preview.

### To create a custom theme

1. Navigate to **Site settings** and under the **Web Designer Galleries** heading, and select **Themes > 15**. A list of .spcolor and .spfont files is displayed.

**Figure 3. Theme Gallery**



1. Download a copy of one of the .spcolor files (e.g. Palette001.spcolor) and open it in a text editor.
2. Edit the copied .spcolor file to reflect your design guidelines. For instance, if you have a black font for main body text, edit the file so that the line **<s:color name=”BodyText” value=”444444” />** is **<s:color name=”BodyText” value=”000000” />.**
3. For each HTML element, add your color**.**
4. When you are done, upload the .spcolor file to the **Site settings > Theme > 15** folder.

**Note:** The file must be saved with a different name than any that appear on the page (e.g. custom\_palette1.spcolor)

Table 3 maps colors and page elements to their code in the .spcolor file. It is a subset of the mappings available in the .spcolor file.

**Table 3. Mapping page elements to colors as they’re defined in .spcolor**

|  |  |  |
| --- | --- | --- |
| **Element** | **Color** | **Code** |
| Body Text | Black | <s:color name="BodyText" value="000000" /> |
| Global navigation background | Blue | <s:color name="HeaderBackground" value="018dff" /> |
| Global navigation text | White | <s:color name="HeaderNavigationText" value="ffffff" /> |
| Current navigation background | Red | <s:color name="NavigationHoverBackground" value="e51400" /> |
| Current navigation text | White | <s:color name="Navigation" value="ffffff" /> |
| Title | White | <s:color name="SiteTitle" value="FFFFFF" /> |
| Footer background | Black | <s:color name="FooterBackground" value="000000" /> |

1. To customize .spfont, download a copy of a .spfont file and open it in a text editor. Notice the .spfont file is laid out a bit differently than .spcolor, but that both files share a similar structure.

**Figure 4. .spfont file**



1. Edit each **<s:fontSlot />** section to customize the font SharePoint applies the specified font slot on the page. For example, notice the first entry **<s:fontSlot name=”title”>.** This entry describes which font SharePoint uses to style the title of the page**.** This section also specifies which font will be used for different languages.

**Note**: You can upload custom fonts to SharePoint and point each entry to a custom .eot, .woff, .ttf, and .svg file.

1. Upload the file to the **Site settings > Theme > 15** folder.

**Note:** The file must be saved with a different name than any that appear on the page (e.g., custom\_font.spfont)

**Table 4.** **Mapping page elements to fonts as they’re defined in .spfont**

|  |  |  |
| --- | --- | --- |
| **Element** | **Font** | **Code** |
| Title | Open Sans | <s:cs typeface="Open Sans" /> |
| Navigation | Roboto | <s:cs typeface="Roboto" /> |
| Headings | Trajan Pro | <s:cs typeface="Trajan Pro" /> |
| Body | Open Sans | <s:cs typeface="Open Sans" /> |

Some custom fonts that aren’t usually available to a user’s browser. For example, if the headings refer to a Trajan Pro font, which is uncommon on most user’s computer, add these font declarations at the top of the **<s:fontSlot>** declaration. Adding this tag will ensure the correct font is displayed:

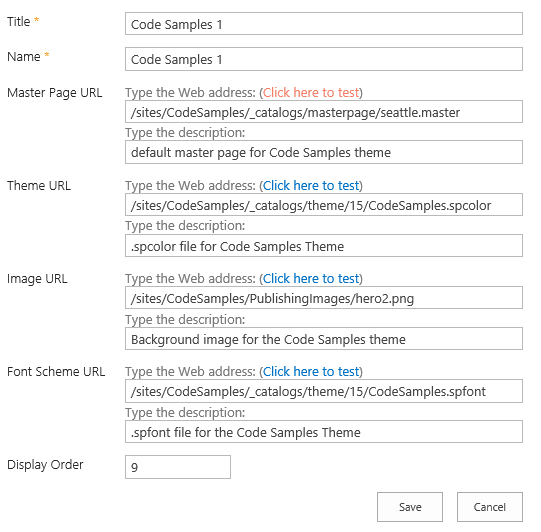
<s:latin typeface=” Trajan Pro” eotsrc=”/SiteAssets/Trajan Pro.eot” woffsrc=”/SiteAssets/Trajan Pro.woff” ttfsrc=”/SiteAssets/Trajan Pro.ttf” svgsrc=”/SiteAssets/Trajan Pro.svg” />

### To add a custom theme to SharePoint

After master page, .spcolor and .spfont customizations are done, add them to the Composed Looks directory so that SharePoint can access them.

1. Navigate to **Site settings** and under **Web Designer Galleries**, and select **Composed Looks.**
2. Click the **new item** link at the top left.

**Figure 5. Composed Looks**



1. Add a title and a name for your composed look.
2. In the Master Page URL field, add the URL of the master page you would like the theme to use.
3. The Theme URL field is the URL of the .spcolor file.
4. The Image URL field is the URL of an image you want to use as a background. This is not required if your design doesn’t call for a background image.
5. The Font Scheme URL is the URL of the .spfont file.

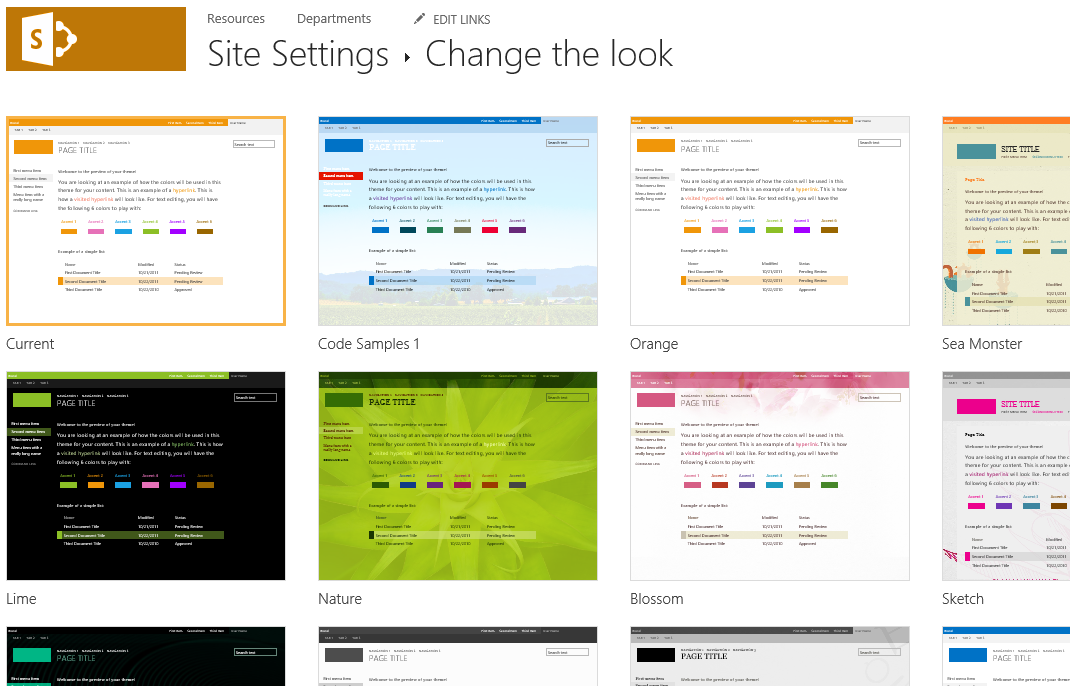
The display order tells SharePoint in what order the composed looks should be displayed.

1. Click **Save**.

You should see your theme entry in the **Composed Looks** list.

Once you’ve successfully added your custom theme to SharePoint as a composed look, SharePoint users can now access the theme and apply it to a site by navigating to **Site settings** and clicking **Change the look** in the **Look and Feel** section.

**Figure 6. Change the look**



## What does the theming engine do when a user applies a composed look?

When a user applies a composed look, SharePoint copies, transforms, and stores CSS in the content database. It also recolors and stores images in the content database. As part of the process of applying a theme to a site, the theming engine pulls color and font values from the specified color palette and font scheme found in the Theme Gallery of the root site. To apply .master page and the master page .preview file (a.k.a. the site layout), the theming engine pulls master pages in the Master Page Gallery that have a corresponding .preview file.

When applying a composed look, the engine maps the settings specified by specific *CSS comments* that the theming engine defines. Under the hood, the theming engine saves the background image to **Site Assets**, scales and compresses JPG and BMP images, and limits the size of GIF and PNG images.

When a composed look is applied to a SharePoint site, SharePoint finds and replaces CSS comment tokens by injecting a value derived from the composed look in the next line in the CSS file after the token. This new value is applied to the SharePoint site.

**Table 5. Theme tokens**

|  |  |  |
| --- | --- | --- |
| **Token** | **Description** | **Corresponding ApplyTheme parameter** |
| /\* ReplaceBGImage \*/ | Swaps current background image with the image in the assigned composed look image URL. | backgroundImageUrl |
| /\* ReplaceFont \*/ | Swaps the current font with one of the fonts found in the font scheme URL of the assigned composed look. | fontSchemeUrl |
| /\* ReplaceColor \*/ | Swaps the current color with one of the colors specified in a color slot in the color palette URL of the assigned composed look. | colorPaletteUrl |
| /\* RecolorImage \*/ | Recolors images using tinting or filling. |  |

# Use remote provisioning to apply themes and get information about themes

The **Change the Look** wizard is powered by the [**Web.ApplyTheme**](http://msdn.microsoft.com/library/office/microsoft.sharepoint.spweb.applytheme.aspx) method, which applies a Composed Look with the specified components to a SharePoint site. Themes are applied on a site-by-site basis. The **ApplyTheme** method applies most theme elements that the **Change the Look** wizard applies.

### ApplyTheme

The [SPTheme](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.utilities.sptheme.aspx) class and its methods and properties are exposed in the server-side object model; however, when using the remote provisioning pattern to apply themes, use a client-side **ApplyTheme** method in your code.

public void ApplyTheme(

string colorPaletteUrl,

string fontSchemeUrl,

string backgroundImageUrl,

bool shareGenerated

)

* colorPaletteUrl is a server-relative URL of the color palette file (i.e., .spcolor).
* fontSchemeUrl is the server-relative URL of the font scheme file (i.e., .spfont).
* backgroundImageUrl is the server-relative URL of the background image; a null reference if there is no background image
* shareGenerated If the generated theme files should be applied to the root web then **true**; **false** to store them in this web.

**Note** The shareGenerated determines whether the themed output files are stored in a web-specific location or a location that is accessible across the site collection. Microsoft recommends not modifying the default value for your site type.

### The ThemeInfo class

You can also use CSOM code to get information about the composed looks applied to a site with the [ThemeInfo](http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.client.themeinfo.aspx) class. **ThemeInfo** gets the context associated with the themes.

public ThemeInfo ThemeInfo { get; }

You can use the **ThemeInfo** class to get details about themes applied to a site, including descriptions, context, object data, colors and fonts for the specified name (and fonts for the specified language code), and the URI for the background image defined for the composed look.

### Theming APIs by programming model

Table 7 indexes theming engine object model members by the object models in which they are exposed. A blank in a column indicates that the member is not exposed in that programming model or implementation.

**Table 6. Theming engine API index**

|  |  |  |  |
| --- | --- | --- | --- |
| **SSOM** | **CSOM** | **JSOM** | **REST** |
| [ApplyTheme](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.spweb.applytheme.aspx) | [ApplyTheme](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.web.applytheme.aspx) | [ApplyTheme](http://msdn.microsoft.com/en-us/library/office/jj838432.aspx) |  |
| [SPTheme](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.sptheme.aspx) |  |  |  |
| [ThemeColor](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.themecolor.aspx) |  |  |  |
| [ThemeFont](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.themefont.aspx) |  |  |  |
| [ThemeFontFace](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.themefontface.aspx) |  |  |  |
| [ThemeType](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.themetype.aspx) |  |  |  |
| [ThmxTheme](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.utilities.thmxtheme.aspx) |  |  |  |
| [ThemeInfo](http://msdn.microsoft.com/library/office/microsoft.sharepoint.spweb.themeinfo.aspx) | [ThemeInfo](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.themeinfo.aspx) | [ThemeInfo](http://msdn.microsoft.com/en-us/library/office/jj246235.aspx) |  |

### Using ApplyTheme and ThemeInfo in CSOM code

The following code example demonstrates how to use **ApplyTheme** and **ThemeInfo** in CSOM code. You can use this code in the remote provisioning pattern. For example, you may decide to programmatically create composed looks as specified by a designer, and provision them to sites in your web application.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Text;

using System.IO;

using Microsoft.SharePoint.Client;

namespace ApplyThemeAppWeb.Pages

{

public partial class Default : System.Web.UI.Page

{

public string \_ContextToken

{

get

{

if (ViewState["ContextToken"] == null)

return null;

return ViewState["ContextToken"].ToString();

}

set

{

ViewState["ContextToken"] = value;

}

}

public string \_HostWeb

{

get

{

if (ViewState["HostWeb"] == null)

return null;

return ViewState["HostWeb"].ToString();

}

set

{

ViewState["HostWeb"] = value;

}

}

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

\_ContextToken = TokenHelper.GetContextTokenFromRequest(Page.Request);

\_HostWeb = Page.Request["SPHostUrl"];

}

StatusMessage.Text = string.Empty;

}

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

{

try

{

StatusMessage.Text += GetThemeInfo();

}

catch (Exception ex)

{

StatusMessage.Text += Environment.NewLine + ex.ToString();

}

}

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

{

try

{

ApplyTheme();

StatusMessage.Text += "Theme applied. Click Get Theme Info to see changes." + Environment.NewLine;

}

catch (Exception ex)

{

StatusMessage.Text += Environment.NewLine + ex.ToString();

}

}

private string GetThemeInfo()

{

using (var clientContext = TokenHelper.GetClientContextWithContextToken(\_HostWeb, \_ContextToken, Request.Url.Authority))

{

Web hostWebObj = clientContext.Web;

ThemeInfo initialThemeInfo = hostWebObj.ThemeInfo;

// Get the initial theme info for the web. No need to load the entire web object.

clientContext.Load(hostWebObj, w => w.ThemeInfo, w => w.CustomMasterUrl);

// Theme component info is available via a method call that must be executed.

var linkShade = initialThemeInfo.GetThemeShadeByName("Hyperlink");

var titleFont = initialThemeInfo.GetThemeFontByName("title", 1033);

// Execute

clientContext.ExecuteQuery();

// Use ThemeInfo to show some data about the theme currently applied to the web.

StringBuilder initialInfo = new StringBuilder();

initialInfo.AppendFormat("Current master page: {0}\r\n", hostWebObj.CustomMasterUrl);

initialInfo.AppendFormat("Background Image: {0}\r\n", initialThemeInfo.ThemeBackgroundImageUri);

initialInfo.AppendFormat("The \"Hyperlink\" Color for this theme is: {0}\r\n", linkShade.Value);

initialInfo.AppendFormat("The \"title\" Font for this theme is: {0}\r\n", titleFont.Value);

return initialInfo.ToString();

}

}

protected void ApplyTheme()

{

using (var clientContext = TokenHelper.GetClientContextWithContextToken(\_HostWeb, \_ContextToken, Request.Url.Authority))

{

// Apply our new theme.

// First, copy theme files to a temporary location (the web's Site Assets library).

Web hostWebObj = clientContext.Web;

Site hostSiteObj = clientContext.Site;

Web hostRootWebObj = hostSiteObj.RootWeb;

// Get the necessary lists and libraries.

List themeLibrary = hostRootWebObj.Lists.GetByTitle("Theme Gallery");

Folder themeFolder = themeLibrary.RootFolder.Folders.GetByUrl("15");

List looksGallery = hostRootWebObj.Lists.GetByTitle("Composed Looks");

List masterLibrary = hostRootWebObj.Lists.GetByTitle("Master Page Gallery");

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

clientContext.Load(themeFolder, f => f.ServerRelativeUrl);

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

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

// First, upload the theme files to the Theme Gallery.

DirectoryInfo themeDir = new DirectoryInfo(Server.MapPath("/Theme"));

foreach (var themeFile in themeDir.EnumerateFiles())

{

FileCreationInformation newFile = new FileCreationInformation();

newFile.Content = System.IO.File.ReadAllBytes(themeFile.FullName);

newFile.Url = themeFile.Name;

newFile.Overwrite = true;

// Sort by file extension into the correct library.

switch (themeFile.Extension)

{

case ".spcolor":

case ".spfont":

Microsoft.SharePoint.Client.File uploadTheme = themeFolder.Files.Add(newFile);

clientContext.Load(uploadTheme);

break;

case ".master":

case ".html":

Microsoft.SharePoint.Client.File updloadMaster = masterLibrary.RootFolder.Files.Add(newFile);

clientContext.Load(updloadMaster);

break;

default:

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

clientContext.Load(uploadAsset);

break;

}

}

// Execute the file upload.

clientContext.ExecuteQuery();

// Create a new composed look for our theme.

string themeFolderUrl = themeFolder.ServerRelativeUrl;

string masterFolderUrl = masterLibrary.RootFolder.ServerRelativeUrl;

// Optional: Use to make the custom theme available for selection in the UI. For

// example, for OneDrive for Business sites, you don’t need this code because

// the ability to set a theme is hidden from OneDrive for Business sites.

ListItemCreationInformation newLook = new ListItemCreationInformation();

Microsoft.SharePoint.Client.ListItem newLookItem = looksGallery.AddItem(newLook);

newLookItem["Title"] = "Theme Sample Look";

newLookItem["Name"] = "Theme Sample Look";

FieldUrlValue masterFieldValue = new FieldUrlValue();

masterFieldValue.Url = masterFolderUrl + "/seattle.master";

newLookItem["MasterPageUrl"] = masterFieldValue;

FieldUrlValue colorFieldValue = new FieldUrlValue();

colorFieldValue.Url = themeFolderUrl + "/ThemeSample.spcolor";

newLookItem["ThemeUrl"] = colorFieldValue;

FieldUrlValue fontFieldValue = new FieldUrlValue();

fontFieldValue.Url = themeFolderUrl + "/ThemeSample.spfont";

newLookItem["FontSchemeUrl"] = fontFieldValue;

newLookItem.Update();

// Apply the master page.

hostWebObj.CustomMasterUrl = masterFieldValue.Url;

// Update between the last and next steps. ApplyTheme throws errors if the theme

// and master page are updated in the same query.

hostWebObj.Update();

clientContext.ExecuteQuery();

// Apply the theme.

hostWebObj.ApplyTheme(

colorFieldValue.Url, // URL of the color palette (.spcolor) file,

fontFieldValue.Url, // URL to the font scheme (.spfont) file (optional)

null, // Background Image URL (optional, null here),

false // false stores the composed look files in this web only. True would share with the site collection (to which we are not currently granted permissions)

);

// Need to call update to apply the change to the host web.

hostWebObj.Update();

// Execute the Update method.

clientContext.ExecuteQuery();

}

}

}

}

[ApplyThemeApp](http://code.msdn.microsoft.com/SharePoint-2013-Use-an-app-d90a49e3) sample

[Themes overview on TechNet](http://technet.microsoft.com/en-us/library/ee424397.aspx)

[Themes content in SharePoint 2013 MSDN](http://msdn.microsoft.com/en-us/library/jj927174.aspx)

[How to: Deploy a custom theme in SharePoint 2013](http://msdn.microsoft.com/en-us/library/jj927175.aspx)

# CSS in SharePoint

CSS plays a large role in SharePoint branding. To successfully customize the site design in SharePoint 2013 and SharePoint Online, it’s useful to be familiar with CSS and how SharePoint uses CSS.

This section presumes a basic understanding of CSS, and links to specific CSS concepts or functions as needed.

## A brief introduction to CSS

This section introduces some basic information about CSS, describes ways you can apply CSS to web pages, and shows how to use alternate CSS and **CSSRegistration** to associate external CSS files with a SharePoint master page.

Table 7 presents a quick reference of basic CSS concepts and syntax.

**Table 7. CSS quick reference**

|  |  |  |
| --- | --- | --- |
| **Concept** | **Description** | **Syntax** |
| Style sheet | A set of statements that specify the presentation of a document.  Style sheets may have three origins: author, user, or user agent (such as a web browser). User agents apply a default style sheet or must behave as if they did. |  |
| CSS rule | A two-part statement that selects an HTML element and declares how to style it. Also called a *style rule*.  Normally, style is applied to structural elements based on their place in the document tree. | H3 {color:green} |
| CSS cascade | Assigns a weight to each style. When more than one style rule applies, the one with the most weight is applied.  To learn more about the CSS cascade and precedence, see [Assigning property values, Cascading, and Inheritance](http://www.w3.org/TR/CSS2/cascade.html#preshint) (W3C). |  |
| Selector | Corresponds to an HTML tag. | “H3” in this statement:  H3 {color:green} |
| Declaration | The two-part definition of the CSS property: property name and property value. | {color:green}  “color” is the property name.  “green” is the property value. |
| Specifying CSS in HTML |  | <HEAD>  <STYLE type=”text/css”>  </HEAD> |
| External style sheets | Are the most flexible way to apply CSS because they can be shared across documents. | <HEAD>  <LINK rel=”stylesheet” href=”contoso.css” type=”text/css”>>  </HEAD> |
| LINK element | Specifies the type of link, its location (via the href attribute), and the type of style sheet linked. |  |
| CSS property definition | The first part of a CSS declaration.  Defines the legal values, syntax, initial property value, HTML or XML elements the property applies to, whether the property is inherited, how percentage values are interpreted, the kinds of media the property applies to, and how to compute a computed value. | [Complete list of CSS properties in CSS 2.1](http://www.w3.org/TR/CSS2/propidx.html) (W3C) |
| CSS property values | The second part of a CSS declaration.  Defines the values of a names property (the “property name”) including keyword values, data types, and value ranges for types with and without the same name as a property. |  |
| [Canvas](http://www.w3.org/TR/CSS21/intro.html) (W3C) |  |  |
| Pseudo-classes and pseudo-elements | Pseudo-classes and pseudo-elements account for cases when applying style to structural elements based on their place in the document tree is not sufficient.  :hover, :active, and :focus are examples of dynamic pseudo-classes that are applied when a user takes certain actions.  Pseudo-elements do not appear anywhere in the document tree or document structure. For example, pseudo-elements may specify styles that are applied to the first line of a paragraph or the first letter in a line. |  |
| Attribute selectors | Rules are allowed to match HTML elements that have specific attributes defined. For example, an attribute selector could match a specific type of *input* element. | An attribute selector could match a specific type of *input* element. |
| CSS specificity | Determines which rule “wins” when more than one can apply, so that the right style is applied. The most specific rule takes precedence. |  |
| CSS inheritance | Some CSS properties (e.g., font, color) pass on values to child elements automatically: inheritance. You don’t have to call out simple settings for every element in an HTML file-for example if you specify colors for an area of a SharePoint page, they can be passed to all child elements automatically. |  |
| [Media queries](http://www.w3.org/TR/css3-mediaqueries/) (W3C) | Media queries specify a media type and can also include expressions that check the conditions of specific media. |  |

### Applying CSS to web pages

SharePoint supports three ways of applying CSS to web pages: inline styles, internal style sheets, and external style sheets.

* Inline styles are defined in HTML. While Microsoft supports this approach, it should be avoided unless otherwise specified.
* Internal style sheets are CSS directly added to HTML files via the <style> tag. While Microsoft supports this approach, CSS added to pages this way are hard to reuse and maintain, and may make your site more difficult to support over time.
* External style sheets are CSS stored in independent files and loaded in HTML with a <link> tag in the <head> section of the page HTML. Microsoft supports and recommends this approach. The guidance in this document assumes that you’re using external style sheets.

## CSS in SharePoint branding

When you create a SharePoint site collection, SharePoint creates a Master Page Gallery (\_catalogs/masterpage) where most branding assets, including .master, .css, image, HTML, and JavaScript files are stored.

In SharePoint 2013, SharePoint uses the seattle.master page by default for Team sites, Publishing sites, and Team sites with Publishing enabled. It uses mysite15.master for OneDrive for Business sites. Each .master file refers to the corev15.css file, which is built from a variety of .css files delivered in SharePoint out-of-the-box.

All default master pages use corev15.css when processing styles, and rely on the CSS cascade and CSS file sharing to resolve which styles are applied to specific controls and elements in regions of a page. Multiple .css files are also combined to build the oslo.css file, which is used with the oslo.master .master page.

* + - 1. CSS in master pages

The <SharePoint:CssRegistration> content placeholder, which corresponds to the [CssRegistration](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.webcontrols.cssregistration.aspx) class in the server-side object model, defines the CSS file. Like a reference to a master page, SharePoint replaces the tokens in the master page when the page is processed. The [CssLink](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.webcontrols.csslink.aspx) class reads the registration from the master page and inserts a LINK tag in the resulting HTML that the system generates and renders.

Consider the following example:

<SharePoint:CssRegistration name="<% $SPUrl:~SiteCollection/Style Library/~language/Core Styles/contoso.css%>" runat="server"/>

At runtime, this code is rendered as:

<link rel="stylesheet" type="text/css" href="/sites/nopub/Style%20Library/en-US/Core%20Styles/contoso.css">

**CSSLink** renders all style sheets when the page is rendered. If you define styles in a custom .css file that are also defined in corev15.css, they are overwritten.

## Corev15.css

A lot of CSS is applied to SharePoint by default. The main source of styles in SharePoint is the **corev15.css** file. This file is stored in the SharePoint 15 folder on the server at *\TEMPLATE\LAYOUTS\1033\STYLES\Themable\COREV15.CSS* (not the Master Page Gallery of the root site and homepage).

To see how SharePoint uses CSS out-of-the-box, look at the **corev15.css** file using developer tools in your web browser. In Internet Explorer, use the Internet Explorer Developer Toolbar (access it by pressing F12 in IE) and click the **CSS** tab to view corev15.css.

Styles defined in **corev15.css** use the .ms- , and .s4- prefixes, which indicate styles that were created by Microsoft. **Corev15.css** also uses a lot of child and [descendent selectors](http://www.w3.org/TR/CSS2/selector.html#descendant-selectors).

When viewing the file, you’ll notice many comments in this format:

/\* [ReplaceFont ( themeFont:”body”)] \*/

SharePoint reads these comments when a composed look is applied. CSS comments tell SharePoint to change the attribute of the CSS that immediately follows the comment. Applying a composed look may change many of the default colors, fonts, and background images that are applied, and subsequently update the settings in corev15.css.

**Note** Selecting the corev15.css file this way loads the rendered CSS rather than the saved CSS—sometimes you may find discrepancies. User agents such as browsers can also change rendering in response to user actions.

**Important** Do not log on to the server and edit or customize core SharePoint CSS files in the SharePoint root. Doing so will negatively impact support and upgrade. Never edit the corev15.css directly; always create a copy, rename it, and edit it instead.

**Caution** Directly editing corev15.css applies branding to all web applications on the server.

##### To create a custom style sheet for SharePoint

1. Make a copy of corev15.css and name it contosov15.css.
2. Open contosov15.css and modify the **CssRegistration** line to point to contosov15.css from corev15.css.

<SharePoint:CssRegistration Name=”Themable/contoso.css” runat=”server” />

1. Below the **CssRegistration** line, add this line:

<SharePoint:CssRegistration name=”/\_catalogs/masterpage/contoso/contoso.css”

runat=”server”>

## Composed looks in custom branding

You can use composed looks in custom branding when CSS is called from a master page. The CSS file is stored in the SharePoint file system in one of the following locations, which are engineered to support themes:

* *15\TEMPLATE\LAYOUTS\{LCID}\STYLES\Themable*
* */Style Library/Themable/*
* */Style Library/{culture}/Themable/*

You can place custom branding files in /*Style Library/Themable/ and /Style Library/{culture}/Themable/*, but *15\TEMPLATE\LAYOUTS\{LCID}\STYLES\Themable* is not editable, so you can’t store custom files in that location.

**Note** If these locations don’t exist by default, you can create them manually and SharePoint will recognize them as themable.

# Applying custom CSS to a SharePoint page

You can add custom CSS to rich text fields and Web Part Zones. To add CSS to a rich text field, put the page in edit mode and choose **Insert > Embed Code** from the ribbon.

For Web Part zones, use the Script Editor Web Part (SEWP) to add HTML, scripts, or an internal style sheet. You can use this approach to hide the Quick Launch navigation in the default SharePoint UI. If you are using SharePoint Online and the NoScript feature, NoScript will disable SEWP.

Apply CSS to a SharePoint via an external style sheet and include a reference to that style sheet in the <link> tag in the <head> of a SharePoint page.

The following code example demonstrates how to upload custom CSS to the Asset Library, apply a reference to the CSS URL with a custom action, and create a custom action to build a link to the new CSS file. It’s an excerpt from the [ProvisionCustomCSS](http://code.msdn.microsoft.com/SharePoint-2013-Provision-bf1d878a) sample.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using Microsoft.SharePoint.Client;

using Microsoft.SharePoint.Client.EventReceivers;

namespace AlternateCSSAppAutohostedWeb.Services

{

public class AppEventReceiver : IRemoteEventService

{

public SPRemoteEventResult ProcessEvent(SPRemoteEventProperties properties)

{

SPRemoteEventResult result = new SPRemoteEventResult();

try

{

using (ClientContext clientContext = TokenHelper.CreateAppEventClientContext(properties, false))

{

if (clientContext != null)

{

Web hostWebObj = clientContext.Web;

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

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

// First, upload the CSS files to the Asset library.

DirectoryInfo themeDir = new DirectoryInfo(System.Web.Hosting.HostingEnvironment.ApplicationPhysicalPath + "CSS");

foreach (var themeFile in themeDir.EnumerateFiles())

{

FileCreationInformation newFile = new FileCreationInformation();

newFile.Content = System.IO.File.ReadAllBytes(themeFile.FullName);

newFile.Url = themeFile.Name;

newFile.Overwrite = true;

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

clientContext.Load(uploadAsset);

break;

}

string actionName = "SampleCSSLink";

// Now, apply a reference to the CSS URL via a custom action.

// Clean up existing actions that we may have deployed.

var existingActions = hostWebObj.UserCustomActions;

clientContext.Load(existingActions);

// Execute uploads and initialize the existing Actions collection.

clientContext.ExecuteQuery();

// Clean up existing actions.

foreach (var existingAction in existingActions)

{

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

existingAction.DeleteObject();

}

clientContext.ExecuteQuery();

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

UserCustomAction cssAction = hostWebObj.UserCustomActions.Add();

cssAction.Location = "ScriptLink";

cssAction.Sequence = 100;

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

cssAction.Name = actionName;

// Apply

cssAction.Update();

clientContext.ExecuteQuery();

}

result.Status = SPRemoteEventServiceStatus.Continue;

return result;

}

}

catch (Exception ex)

{

result.Status = SPRemoteEventServiceStatus.CancelWithError;

result.ErrorMessage = ex.Message;

return result;

}

}

public void ProcessOneWayEvent(SPRemoteEventProperties properties)

{

// This method is not used by app events.

}

}

}

# Customizing regions of a SharePoint page

You can use a combination of remote provisioning and custom CSS to customize the regions of a SharePoint page.

### Mapping regions of a page to CSS, JavaScript, and CSOM elements

Table 8 maps regions of a SharePoint page to associated files and functionality to the regions of the page.

**Table 8. Regions of a SharePoint page, associated files, and example customizations**

|  |  |  |
| --- | --- | --- |
| **Name** | **Associated files and functionality** | **Additional information and example customizations** |
| The Ribbon | Any default .master page  Can be hidden by **Focus On Content**  Corresponding CSS   * Main Body: body #s4-workspace * Suite Bar – Left: #suiteBarLeft * Ribbon Container: #globalNavBox * Suite Bar – Right: #suiteBarRight |  |
| Suite Navigation | Any default .master page  Can be hidden by **Focus On Content**. |  |
| Suite Links | Can be hidden by **Focus On Content**. |  |
| Welcome Menu | .master  Can be hidden by **Focus On Content**. |  |
| Settings Menu or Gear | .master | [Customize the site settings link using the CustomAction API in CSOM (Vesa Juvonen)](http://blogs.msdn.com/b/vesku/archive/2013/10/02/ftc-to-cam-custom-actions-and-property-bag-entries.aspx) |
| Help | .master |  |
| Ribbon Contextual Tabs | Any default .master page | [Using custom code from SharePoint 2010 can cause duplicate contextual tabs to appear in SharePoint 2013](http://social.msdn.microsoft.com/Forums/sharepoint/en-US/df1e4e32-ef58-4b51-8ac8-a8c3690e648b/sharepoint-2013-duplicate-contextual-tabs?forum=sharepointdevelopment) (Microsoft forums)  [Hide a contextual ribbon tab](http://social.msdn.microsoft.com/Forums/sharepoint/en-US/a3640d58-afe1-41d0-ac83-bd7886c37355/hide-a-contextual-ribbon-tab?forum=crmdevelopment) (Microsoft forums)  [Show/hide on click event of subgrid](http://social.msdn.microsoft.com/Forums/sharepoint/en-US/201306cf-5874-4778-b773-f870c67cee94/hideshow-contextual-tab-on-click-event-of-subgrid?forum=crmdevelopment) (Microsoft forums) |
| Quick Access Toolbar | .master  Can be hidden by **Focus On Content**. |  |
| Site Logo | .master  Corresponding CSS:  .ms-siteIcon-img |  |
| Top Navigation | Nav CSOM/JSOM  .master  Corresponding CSS (not in Edit Mode):   * New Item selected: .ms-core-listMenu-horizontalBox li.static > .ms-core.listMenu-selected * New Item Hover: .mscore-listMenu-horizontalBox li.static > a.ms-core-listMenu-item:hover * Flyout Arrow: .ms-core-listMenu-horizontalBox .dynamic-children.additional-background * Nav Item (corresponding to top-level menu items): .ms-core-listMenu-horizontalBox li.static > .ms-core-listMenu-item * Flyout Item: ul.dynamic .ms-core-listMenu-item * Flyout Container: ul.dynamic * Edit Links: .ms-navedit-editLinksText > span> .ms-metadata   Corresponding CSS (in Edit Mode):   * Nav Edit Mode Link: .ms-core-listMenu-horizontalBox .ms-core-listMenuEdit > tr> .msnavedit-linkCell > .ms-core-listMenu-item * Add Link: .ms-core-listMenu-horizontalBox a.ms-navedit-addNewLink | [How to: Brand snippets by using CSS in SharePoint 2013](http://msdn.microsoft.com/en-us/library/dn205275.aspx) (MSDN)  [Structured navigation (2010) (MSDN](http://msdn.microsoft.com/en-us/library/office/ms558975(v=office.14).aspx))  [Managed navigation (2013)](http://msdn.microsoft.com/en-us/library/jj163978.aspx) (MSDN)  [Managed metadata and navigation in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj163949.aspx) (MSDN) |
| Page Title | Corresponding CSS:   * Page Title and Page Title with Link: .ms-core-pageTitle, .ms-core-pageTitle a * Description button: #ms-pageDescriptionDiv * Description box: .js-callout-mainElement * Description box arrow: .js-callout-beak * Description text: .js-callout-body |  |
| Search Box | Nav CSOM/JSOM  .master  Corresponding CSS:   * Search Box Border: .ms-srch-sb-border * Search Box Border Hover: .ms-srch-sb-border: hover * Search Box Border when clicked: .ms-srch-sb-borderFocused * Search Box Input Text Box: .ms-srch-sb-borderFocused * Search Box Body: .ms-srch-sb * Search Box Input Text Box: .ms-srch-sb-searching * Search |  |
| Left Navigation | Nav CSOM/JSOM  .master | [Structured navigation (2010) (MSDN](http://msdn.microsoft.com/en-us/library/office/ms558975(v=office.14).aspx))  [Managed navigation (2013)](http://msdn.microsoft.com/en-us/library/jj163978.aspx) (MSDN)  [Structured navigation via code](http://social.msdn.microsoft.com/forums/sharepoint/en-US/dbeb857a-74fc-4ef1-b6ec-d359dd3db8b1/how-do-i-set-the-global-nav-to-structural-navigation-via-code) (MSDN forums)  [Custom navigation in SharePoint 2010](http://msdn.microsoft.com/en-us/library/office/bb897739(v=office.14).aspx) (MSDN forums) |
| Tree View | .master | [Customize tree view](http://social.msdn.microsoft.com/Forums/sharepoint/en-US/dd4d49fd-e107-469d-b326-d37c86ff66b8/how-to-customize-the-builtin-treeview-navigator-?forum=sharepointcustomizationprevious) (MSDN forums) |
| Page Content | Page Layout/Content Pages  Web Part Zone/Web Parts  Corresponding CSS (Web Part Zone and Web Part):   * Web Part Zone: .ms-webpart-zone * Web Part Holder: .ms-webpartzone-cell * Web Part Title: .ms-webpart-titleText * Web Part Title with Link: .ms-webpart-titleText > a * Web Part Body: .ms-WPBody | [How to: Create a page layout in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj822368.aspx) (MSDN) |

[AlternateCSSAppAutohosted](http://code.msdn.microsoft.com/SharePoint-2013-Use-an-app-c5d78f25) sample

[ApplyThemeApp](http://code.msdn.microsoft.com/SharePoint-2013-Use-an-app-d90a49e3) sample

[ProvisionCustomCSS](http://code.msdn.microsoft.com/SharePoint-2013-Provision-bf1d878a) sample

### Required “minimal” content placeholders in default SharePoint master pages

SharePoint .master pages require the following content placeholders, which render the basic content and structural elements that a SharePoint page requires during the page lifecycle. Parenthetical references to region in Table 9 refer to page regions itemized in Table 10.

**Table 9. Minimum required content placeholders for a SharePoint master page**

|  |  |
| --- | --- |
| **Content placeholder** | **Holds content for** |
| PlaceHolderAdditionalPageHead | Additional items in the <head> section of a page |
| PlaceHolderBodyAreaClass | Additional styles in the page header |
| PlaceHolderBodyLeftBorder | The left border element for the body of the page. |
| PlaceHolderBodyRightBorder | The right border element for the body of the page |
| PlaceHolderCalendarNavigator | A date picker for navigating in a calendar when a calendar is visible on a page |
| PlaceHolderFormDigest | The “form digest” security control |
| PlaceHolderGlobalNavigation | The global navigation breadcrumb (Top Navigation) |
| PlaceHolderGlobalNavigationSiteMap | The site map in the global navigation (Top Navigation) |
| PlaceHolderHorizontalNav | The top navigation menu for a page (Top Navigation) |
| PlaceHolderLeftActions | The bottom left navigation area (Left Navigation) |
| PlaceHolderLeftNavBar | The left navigation area (Left Navigation) |
| PlaceHolderLeftNavBarDataSource | The data source for the left navigation menu (Left Navigation) |
| PlaceHolderLeftNavBarTop | The top left navigation area (Left Navigation) |
| PlaceHolderMain | The main content of the page (Page Content) |
| PlaceHolderMiniConsole | Page-level commands, such as Edit Page, History, and Incoming Links on an enterprise wiki page. |
| PlaceHolderNavSpacer | The width of the left navigation area (Left Navigation) |
| PlaceHolderPageDescription | Description of the page contents |
| PlaceHolderPageImage | Page icon in the upper left corner of the page (The Ribbon) |
| PlaceHolderPageTitle | The page title (<title>) (Page Title) displayed in the title bar of the browser page |
| PlaceHolderPageTitleInTitle | The page title (Page Title) shown immediately below the breadcrumb |
| PlaceHolderQuickLaunchBottom | The bottom of the Quick Launch navigation (Top Navigation) |
| PlaceHolderQuickLaunchTop | The top of the Quick Launch navigation (Top Navigation) |
| PlaceHolderSearchArea | The area where the search box control appears (Search Box) |
| PlaceHolderSiteName | The name of the site (Suite Navigation) |
| PlaceHolderTitleAreaClass | The title area of the page (Suite Navigation) |
| PlaceHolderTitleAreaSeparator | Shadows in the title area (Suite Navigation) |
| PlaceHolderTitleBreadCrumb | The title breadcrumb content area |
| PlaceHolderTitleLeftBorder | The left border of the title area (Suite Navigation) |
| PlaceHolderTitleRightMargin | The right margin of the title area (Suite Navigation) |
| PlaceHolderTopNavBar | The top navigation area (Top Navigation) |
| PlaceHolderUtilities | Extra content that must appear at the bottom of the page (Page Content) |
| SPNavigation | Wraps navigation-related operations. |
| WSSDesignConsole | The page editing controls when the page is in **Edit** mode |

If you remove one of the content placeholders listed in Table 11 from a SharePoint .master page, SharePoint will throw an error. You can add a content placeholder with hidden visibility, which hides the content from end users.

The content placeholders required in a SharePoint master page have been consistent for several versions. This resource—which describes the [content placeholders in default master pages](http://msdn.microsoft.com/en-us/library/office/ms467402(v=office.12).aspx) in Windows SharePoint Services 3, still largely applies.

[Working with content placeholder controls (Office.com – applies to SharePoint 2010)](http://office.microsoft.com/en-us/sharepoint-designer-help/working-with-content-placeholder-controls-HA102265026.aspx)

* + - 1. Beyond the minimum content placeholders

Default SharePoint master pages such as seattle.master and oslo.master include many more content placeholders than SharePoint requires. For example, these master pages include the <SharePoint:Themes runat=”server”> and <SharePoint.CssRegistration runat=”server”> controls.

Both the Themes and CssRegistration controls run during the page lifecycle. Using the remote provisioning pattern, you can use a custom action to add a server control to register custom CSS.

* + - 1. Hiding content placeholders

Content placeholders that are not visible still work as expected, but any content they generate will be omitted from the HTML source that browsers recognize. A content placeholder with Visible=”false” is hidden.

**Important**  Do not create custom placeholders in custom master pages that exist in out-of-the-box .master pages such as Seattle.master and Oslo.master. Doing so will cause catastrophic exceptions.

### The SharePoint Online Suite Navigation Control

SharePoint Online introduces new master page markup for the Suite Navigation control, which renders the top navigation. The default markup for the Suite Navigation control differs depending on whether the site is an intranet site or a public-facing site. To learn more about the Suite Navigation control and see code examples for both intranet site and public-facing sites that you can add to your master pages, see [SharePoint Online Suite Navigation Control](http://msdn.microsoft.com/en-us/library/office/dn614990(v=office.15).aspx).

### Using CSOM to customize the regions of a SharePoint page

Generally, Microsoft recommends using the [UserCustomAction](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.usercustomaction.aspx) class to add or remove links and other elements. This is a variant of using the [CustomAction element](http://msdn.microsoft.com/en-us/library/ms460194.aspx), which developers who are familiar with SharePoint’s full-trust code model may recognize as tied to the feature framework. While the **CustomAction** element and feature framework provisioning pattern are not specifically supported in CSOM, the same location identifiers available to the **CustomAction** element can be used in CSOM code.

<CustomAction

RequiredAdmin = "Delegated | Farm | Machine"

ControlAssembly = "Text"

ControlClass = "Text"

ControlSrc = "Text"

Description = "Text"

FeatureId = "Text"

GroupId = "Text"

Id = "Text"

ImageUrl = "Text"

Location = "Text"

RegistrationId = "Text"

RegistrationType = "Text"

RequireSiteAdministrator = "TRUE" | "FALSE"

Rights = "Text"

RootWebOnly = "TRUE" | "FALSE"

ScriptSrc = "Text"

ScriptBlock = "Text"

Sequence = "Integer"

ShowInLists = "TRUE" | "FALSE"

ShowInReadOnlyContentTypes = "TRUE" | "FALSE"

ShowInSealedContentTypes = "TRUE" | "FALSE"

Title = "Text"

UIVersion = "Integer">

</CustomAction>

### To customize the SharePoint ribbon

When you customize the ribbon, the HTML that the server renders are assigned to the class name that you assign to a custom style.

Navigate to the **Style Library** and create a new .css file for each style that you want to add to the ribbon. You can add custom styles to two sections of the ribbon: Page Elements and Text Styles. Use the following syntax for the styles you add:

* For the Page Elements section, use this syntax: *span.ms-rteElement-<yourowndefinedname>.* Alternatively, you can use H1, H2, H3, or H4 which will be wrapped around the text that the style will be added to.
* For the Text Styles section, use this syntax: *.ms-rteEStyle-<yourowndefinedname>.*

In your CSS class definition, add this line:

*-ms-name:"The name visual in the ribbon for your style";*

Finish defining the details of your custom CSS class in your custom .css file.

### To customize Suite Navigation on a Web Part page

In SharePoint 2013, the SharePoint user interface has a modern look-and-file based on page tiles. For example, Live Tiles appear on the default SharePoint 2013 page by default. The top navigation makes it easy for users to see and access social content, OneDrive for Business, and so on. You can customize the look and feel of these areas using a mix of CSS and JavaScript.

After you create a Web Part page, add a Script Editor Web Part (SEWP) to an available Web Part zone. You can use this Web Part to add JavaScript to your page.

You can add a snippet of JavaScript code to the SEWP that identifies the top navigation bar by its **ElementId**, and then hide it by setting its visibility property to hidden.

### To customize the Settings menu or gear

You can use [user custom actions](http://msdn.microsoft.com/en-us/library/office/microsoft.sharepoint.client.usercustomaction.aspx) and property bag entries to customize the settings menu of any SharePoint site.

public void AddCustomActions(ClientContext clientContext)

{

// Add a site settings link if it doesn't already exist.

if (!CustomActionAlreadyExists(clientContext, "Sample\_CustomSiteSetting"))

{

// Add a site settings link.

UserCustomAction siteSettingLink = clientContext.Web.UserCustomActions.Add();

siteSettingLink.Group = "SiteTasks";

siteSettingLink.Location = "Microsoft.SharePoint.SiteSettings";

siteSettingLink.Name = "Sample\_CustomSiteSetting";

siteSettingLink.Sequence = 1000;

siteSettingLink.Url = string.Format(DeployManager.appUrl, clientContext.Url);

siteSettingLink.Title = "Modify Site Metadata";

siteSettingLink.Update();

clientContext.ExecuteQuery();

}

// Add a site actions link, if it doesn't already exist.

if (!CustomActionAlreadyExists(clientContext, "Sample\_CustomAction"))

{

UserCustomAction siteAction = clientContext.Web.UserCustomActions.Add();

siteAction.Group = "SiteActions";

siteAction.Location = "Microsoft.SharePoint.StandardMenu";

siteAction.Name = "Sample\_CustomAction";

siteAction.Sequence = 1000;

siteAction.Url = string.Format(DeployManager.appUrl, clientContext.Url); ;

siteAction.Title = "Modify Site Metadata";

siteAction.Update();

clientContext.ExecuteQuery();

}

}

### To customize the tree view

To modify the width of the tree view, add a <DIV> tag around the tree tag in the .master page and assign a CSS class with a style width attribute to the <DIV>.

You can increase the width of the Quick Launch navigation with this style:

.ms-nav {

width: 220 px

}

### To customize the page content

Requirements for customizing page content depends on the content you’re including in your page.

As is the case with customizing the Site Actions menu, you can use a **UserCustomAction** in CSOM to provision branding to Web Parts.

If you are building a Publishing site, see [How to: Create a page layout in SharePoint 2013](http://msdn.microsoft.com/en-us/library/office/jj822368.aspx) to learn the basics. Page layouts depend on the availability of the **ContentTypeId** in CSOM. As is the case with other members that aren’t available in CSOM, you can use a Windows Communication Foundation (WCF) service to work with **ContentTypeId** as a temporary workaround.

See the **ServicesSiteManager** sample in this solution pack to learn more.

# OneDrive for Business site branding

You can use remote provisioning APIs to brand OneDrive for Business sites. The sample [OneDriveForBusinessSiteBranding](http://code.msdn.microsoft.com/SharePoint-2013-Brand-a-6da627cb) demonstrates the approach.

By using a direct client context rather than the apps for SharePoint authorization pattern, the site provisioning code is executed only in the context of a particular user. This enables us to ignore app permissions completely.

using Microsoft.SharePoint.Client;

using Microsoft.SharePoint.Client.UserProfiles;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace PersonalSiteBrandingEditWeb.Pages

{

public partial class ModifyPersonalMySite : Page

{

protected void Page\_Load(object sender, EventArgs e)

{

Uri hostWeb = new Uri(Request.QueryString["SPHostUrl"]);

// This code uses direct client context, not the apps for SharePoint

// authorization pattern. This approach executes the code

// only in the context of the specific user and ignores app permissions completely.

using (ClientContext clientContext = new ClientContext(hostWeb))

{

// Get user profile

ProfileLoader loader = Microsoft.SharePoint.Client.UserProfiles.ProfileLoader.GetProfileLoader(clientContext);

UserProfile profile = loader.GetUserProfile(); ;

Microsoft.SharePoint.Client.Site personalSite = profile.PersonalSite;

clientContext.Load(personalSite);

clientContext.ExecuteQuery();

// Let's see whether the site already exists.

if (personalSite.ServerObjectIsNull.Value)

{

// Let's queue the OneDrive for Business site creation using an out-of-the box

// timer job. Use async mode, since the user could go away from browser.

// You could also use an out-of-the-box Web Part for this.

profile.CreatePersonalSiteEnque(true);

clientContext.ExecuteQuery();

Response.Write("No OneDrive for Business site exists. Currently provisioning...");

}

else

{

// The site already exists; modify branding by applying a theme.

// You could also upload the master page and set it. You can also modify

// this code to change the branding later, and updates would appear when

// the user visits the OneDrive for Business site host, or any other

// location where this app part is located. You could apply the app part

// to the front page of the intranet to ensure that it is applied.

using (ClientContext subContext = new ClientContext(personalSite.Url))

{

// Let's update the theme colors of the OneDrive for Business site.

Microsoft.SharePoint.Client.Web rootWeb = subContext.Web;

subContext.Load(rootWeb);

subContext.ExecuteQuery();

rootWeb.ApplyTheme(URLCombine(rootWeb.ServerRelativeUrl, "/\_catalogs/theme/15/palette008.spcolor"),

URLCombine(rootWeb.ServerRelativeUrl, "/\_catalogs/theme/15/fontscheme003.spfont"),

null, false);

subContext.ExecuteQuery();

// Output status

Response.Write("OneDrive for Business site exists: " + personalSite.Url + " - web title - " + subContext.Web.Title);

}

}

}

}

private string URLCombine(string baseUrl, string relativeUrl)

{

if (baseUrl.Length == 0)

return relativeUrl;

if (relativeUrl.Length == 0)

return baseUrl;

return string.Format("{0}/{1}", baseUrl.TrimEnd(new char[] { '/', '\\' }), relativeUrl.TrimStart(new char[] { '/', '\\' }));

}

}

}

# SharePoint Online NoScript and branding considerations

Branding SharePoint Online sites is very similar to branding on-premises and provider-hosted SharePoint sites in many respects. For example, SharePoint Online uses the same page model that SharePoint Team, Publishing, and OneDrive for Business sites use to render pages. However, there are a few differences that designers and developers working on site branding should keep in mind.

* The branding customization options available to you depend largely on your SharePoint Online SKU.
* SharePoint Online provides a different set of default .master pages: Kyoto.master, Berlin.master, and Lyon.master.
* Every user of a SharePoint Online site can also modify its site collection.
* NoScript, an administrative setting that prevents site collection administrators from running scripts and modifying settings on others’ SharePoint Online sites, blocks the provisioning of most branding elements to SharePoint Online sites when it’s turned on.

## NoScript and branding

Because every user of a SharePoint Online site can also modify its site collection, we need a way to protect users in collaborative environments, such as students in online schools, from modifying settings and running scripts on other users’ SharePoint Online sites. NoScript was designed with that in mind.

If NoScript is off, users can access Save Site as Template, Solution Gallery, Theme Gallery, Help Settings, and Web Parts as they normally would. When NoScript is on, affected features either no longer appear in Site Settings or selecting an affected Web Part returns an error when a user tries to add it.

**Table 10. Effects of NoScript on SharePoint Online features**

|  |  |
| --- | --- |
| **Feature** | **Behavior when NoScript is enabled** |
| Save Site as Template | No longer appears in Site Settings.  You can still create sites from templates that were deployed before NoScript was enabled. |
| Solution Gallery | No longer appears in Site Settings.  You can still use solutions that were deployed before NoScript was enabled. |
| Theme Gallery | No longer appears in Site Settings.  You can still use themes that were deployed before NoScript was enabled. |
| Help Settings | No longer appears in Site Settings.  You can still access help files that were deployed before NoScript was enabled. |
| Web Parts | When a user selects a web part to add to a SharePoint Online page and NoScript is enabled, SharePoint Online returns an error message that prevents the user from adding the Web Part to the page.  The following types of Web Parts are affected:   * Blog: Blog archives, notifications, and tools * Business Data: All parts except the filter (business data- -actions, -item, -item builder, -list, -related list, Excel Web Access, indicator details, status list, Visio Web Access) * Community: About this community, join, my membership, tools, what's happening * Content Rollup parts: Categories, Project Summary, Relevant Documents, RSS Viewer, Site Aggregator, Sites in Category, Term Property, Timeline, WSRP Viewer, XML Viewer * Document Sets:  Document set contents and properties * Forms: HTML Form Web Part * Media and Content: Content Editor, Script editor, Silverlight Web Part * Search: all parts: Refinement, Search Box, Search Navigation, Search Results * Search-driven Content: Category-Item reuse * Social Collaboration: all except Site users (Contact Details, Note Board, Organization Browser, Site Feed, Tag Cloud, User Tasks |

SharePoint administrators can enable and disable NoScript in SharePoint Central Administration or via a PowerShell cmdlet.

NoScript does not affect the Master Page Gallery.

# Conclusion

In Module 4, you learned about composed looks, themes, custom SharePoint CSS, and how to use a combination of remote provisioning CSOM code, CSS, and scripting tactics to customize the regions of a SharePoint page.

In Module 5, we’ll take closer look at the remote provisioning patterns.