## Developing for SharePoint Online using JavaScript Injection

**Lab Time**: 45 minutes

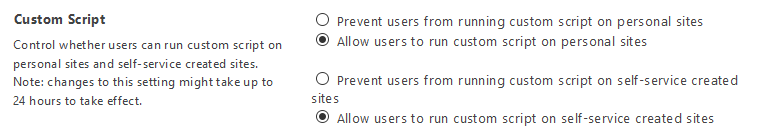
**Lab Folder**: C:\Student\Modules\JavaScriptInjection\Lab

**Lab Overview**: In this lab you will use 3 different techniques to inject JavaScript into a SharePoint site. First, you will use a simple Script Editor web part. Next, you will use a JS Link file as a vehicle for delivering code. Finally, you will use a script custom action injected through a SharePoint add-in. All of these techniques allow you to get client-side code into the SharePoint host web.

### Exercise 1: Create a New Test Site

In this exercise, you will create a new SharePoint to use with the subsequent exercises.

1. Enable scripting support for the SharePoint online site collection.
   1. In Office 365, select **Admin** from the **App Launcher**.
      1. On the Administration page, select **Admin>SharePoint**.
      2. In the SharePoint Admin Center, click **Settings**.
      3. In the Custom Script section, select the options to allow scripting.



### Exercise 2: Working with the CDN Manager

In this exercise you will work with the **CDN Manager** add-in. The CDN Manager provides centralized management of JavaScript library references for a SharePoint site.

1. Install the CDN Manager add-in in SharePoint online
   1. On the Administration page, select **Admin>SharePoint**.
   2. In the SharePoint Admin Center, click **Apps**.
   3. Click **App Catalog**.

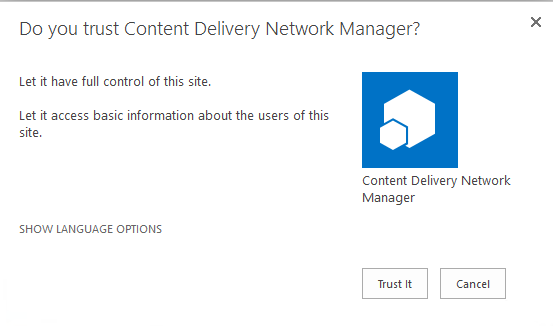
If the App Catalog was not created previously, then you will be prompted to create it now. If the App Catalog was created previously, you will be taken to the catalog site.

* 1. In the App catalog site, click **Apps for SharePoint**.
  2. Click **Upload**.
  3. In the **Add a Document** dialog, browse to the **cdnManager.app** file.
  4. Click **OK**.

1. Add the CDN Manager to the test site
   1. Navigate to the test site you created previously.
      1. Click **Site Contents**.
      2. Click **Add an App**

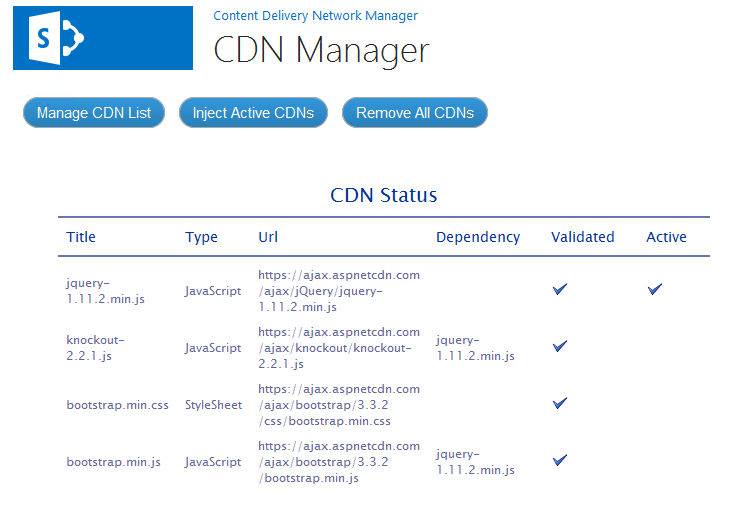


* + 1. Click **From Your Organization**.
    2. Click the **Content Delivery Manager** add-in.
    3. When prompted to trust the app, click **Trust It**.



* + 1. Wait for the add-in to finish installing.

1. Configure the CDN Manager
   1. On the Site Contents page, click the **Content Delivery Network Manager** add-in.
   2. In the CDN Manager, you will see references to jQuery, knockout, and bootstrap.



The CDN Manager allows you to select JavaScript and style sheet references to inject into the site. The CDN Manager maintains the following information:

**Title** field: The name of the JavaScript library or style sheet

**Type** field: “JavaScript” or “StyleSheet”

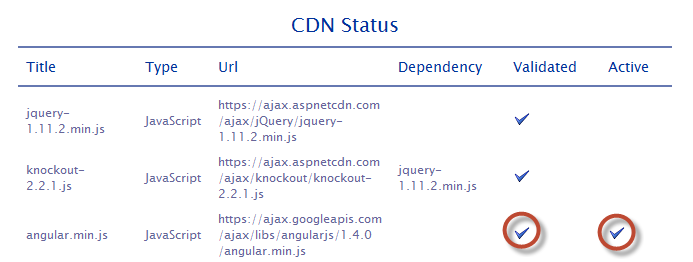
**Url** field: The protocol-relative URL to the associated content delivery network

**Dependency**: Libraries that must be loaded prior to loading this library

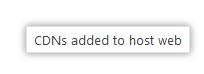
**Validated**: The CDN Manager tests the URL and validates the library is present at the other end

**Active**: The library is selected for injection into the current site. Active libraries are injected when the **Inject Active CDNs** button is clicked.

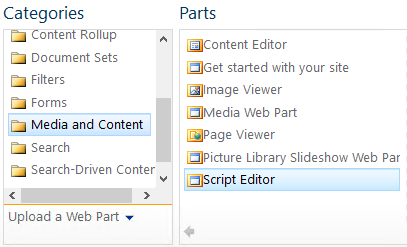
* 1. Add a reference to the Angular framework and inject it into the current site.
     1. Click **Manage CDN List**.
     2. Click **New Item**.
        1. Type **angular.min.js** in the **Title** field.
        2. Type **//ajax.googleapis.com/ajax/libs/angularjs/1.4.0/angular.min.js** in the **Url** field.
        3. Check the **Active** box.
        4. Enter **1** in the **Sequence** field to indicate the loading sequence.
        5. Select **JavaScript** in the **Type** dropdown.
        6. Leave the **Dependency** field blank.
        7. Enter **angular** in the Namespace field to indicate the name of the object that will be present when the library successfully loads.
        8. Click **Save**.
  2. Remove the jQuery reference.
     1. Select **Edit Item** using the item menu associated with the jQuery library.
     2. Uncheck the **Active** box.
     3. Click **Save**.
  3. Inject the active CDN into the site.
     1. Click the **Content Delivery Network Manager** link to return to the add-in home page.
     2. Ensure the Angular reference is active and validated.



* + 1. Click Inject Active CDNs.
    2. Verify that you see a success message.



1. Add a Script Editor web part to the page.
   1. On the home page of the test site you created earlier, select Edit Page from the settings menu.
   2. Click **Insert>Web Part**.
   3. In the **Media and Content** folder, select **Script Editor**.



* 1. Click **Add**.

1. Add Angular code to the web part
   1. Select **Edit Web Part** in the Script Editor.
   2. Click **Edit Snippet**.
   3. Add the following HTML JavaScript to the web part to define a simple Angular application.

<div data-ng-app='myApp'>

<div data-ng-controller='myCtrl'>

<div data-ng-show="showWelcome">Welcome, {{fullName}}</div>

<div data-ng-show="!showWelcome">Working on it...</div>

</div>

</div>

<script>

CDNManager.getScript('angular.min.js',function() {

angular.module('myApp',[]).controller('myCtrl', ['$scope', '$http', function myCtrl($scope, $http) {

$scope.fullName ="";

$scope.showWelcome = false;

$http({

url: "../\_api/web/currentUser",

params: {

"$select": "Title"

},

method: "GET",

headers: {

"accept": "application/json"

}

}).then( function (json) {

$scope.fullName = json.data.Title;

console.log(json.data.Title);

}).catch( function (err) {

alert(err.statusText);

}).finally( function () {

$scope.showWelcome = true

});;

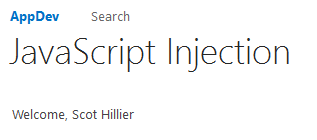
}]);

});

</script>

The CDN Manager allows you to write JavaScript without explicitly referencing a JavaScript library. By centralizing the references, the CDN Manager prevents multiple library references by different developers, references to different versions of the same library, and unnecessary references. To use the CDN Manager, developers imply place their code within the CDNManager.getScript() method. The getScript() method ensures libraries are loaded in the proper order and then executes the contained script.

* 1. Click **Insert**.
  2. Click **OK** in the web part properties dialog.
  3. Click **Save** in the ribbon to save the page. You should now see a welcome message from the web part.



### Exercise 3: Injecting Script through JSLink

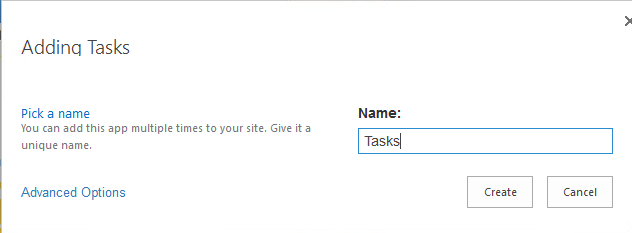
In this exercise you will use a JSLink file to inject an Angular service into the home page of the test site.

1. Activate the Publishing features.
   1. Activate the publishing infrastructure.
      1. Select **Site Settings** from the settings menu.
      2. Click **Go To Top Level Site Settings**.
      3. Click **Site Collection Features**.
      4. Activate the **SharePoint Server Publishing Infrastructure** feature if not already activated.
   2. Return to the home page of the test site.
      1. Select **Site Settings** from the settings menu.
      2. Click **Manage Site Features**.
      3. Activate **SharePoint Server Publishing**.
2. Configure the CDN Manager
   1. Click **Site Contents**.
   2. On the Site Contents page, click the **Content Delivery Network Manager** add-in.
   3. Click **Manage CDN List**.
   4. Click the **List** tab.
   5. Click **Quick Edit**.
   6. **Activate** all of the CDN entries.

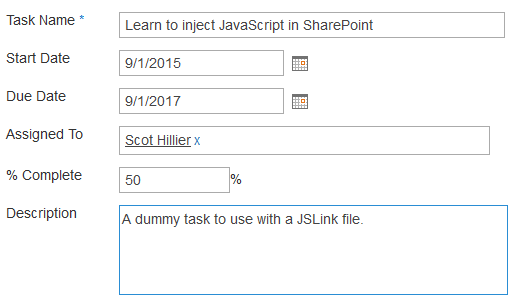


* 1. Click the **Browse** tab.
  2. Click **Stop Editing this List**.
  3. Click the **Content Delivery Network Manager** link to return to the add-in home page.
  4. Click **Inject Active CDNs**.
  5. Return to the home page of the test site.

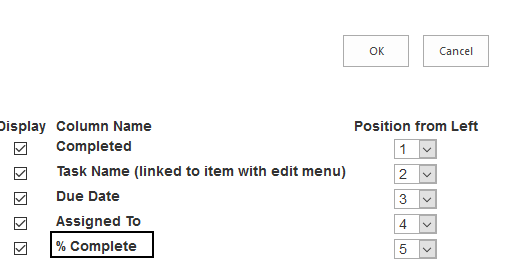
1. Inject an Angular service using a JSLink file attached to a Task list.
   1. Add a Task list to the site.
      1. Click **Site Contents**.
      2. On the Site Contents page, click **Add an App**.
      3. Click **Tasks**.
      4. Name the new Task list **Tasks** and click **Create**.



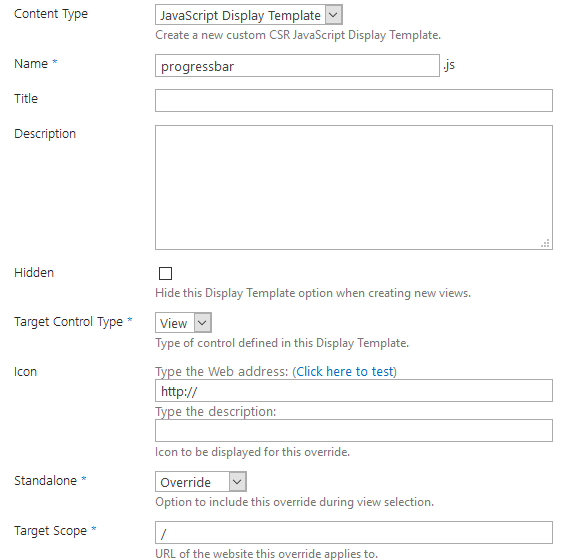
* + 1. Click the new **Task** list.
    2. Click **New Task**.
    3. Create a new task with any data you like as long as you set the **Percent Complete** field to **50%**



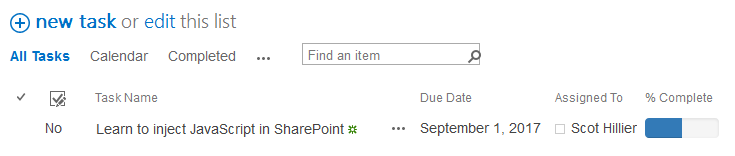
* + 1. Click **Save**.
  1. Modify the list view
     1. Click the ellipsis above the Tasks list and select **Modify this View**.
     2. Check the **% Complete** field.
     3. Click **OK**.



1. Add the JSLink file to the Master Page Gallery.
   1. Select **Site Settings** from the Settings menu.
   2. On the Site Settings page, click **Master Pages and Page Layouts**.
   3. Click the **Files** tab.
   4. Click **Upload** document.
   5. Browse to the **progressbar.js** file.
   6. Click **OK**.
   7. Update the file metadata
      1. Select **JavaScript Display Template** from the **Content Type** dropdown.
      2. Select **View** from the **Target Control Type** dropdown.
      3. Select **Override** from the **Standalone** dropdown.
      4. Enter a forward slash (**/**) in the **Target Scope** field.
      5. Click **Save**.

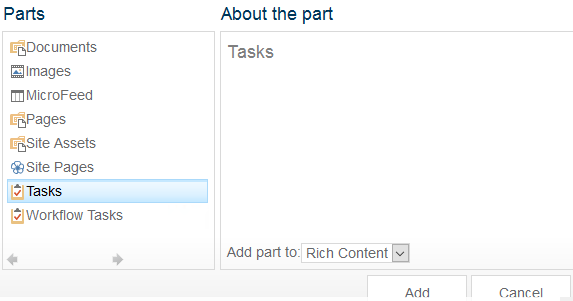


1. Add the JS Link reference to the Task list.
   1. Return to the home page of the test site.
   2. Click the link for the **Tasks** list.
   3. Select **Edit Page** from the settings menu.
   4. Select **Edit Web Part** from the Task list view web part.
   5. Expand the **Miscellaneous** category in the web part properties dialog.
   6. Type **~sitecollection/\_catalogs/masterpage/progressbar.js** in the JS Link field.
   7. Click **OK**.
   8. Click **Page>Stop Editing**.
   9. You should now see the graphic progress bar replacing the normal text field.



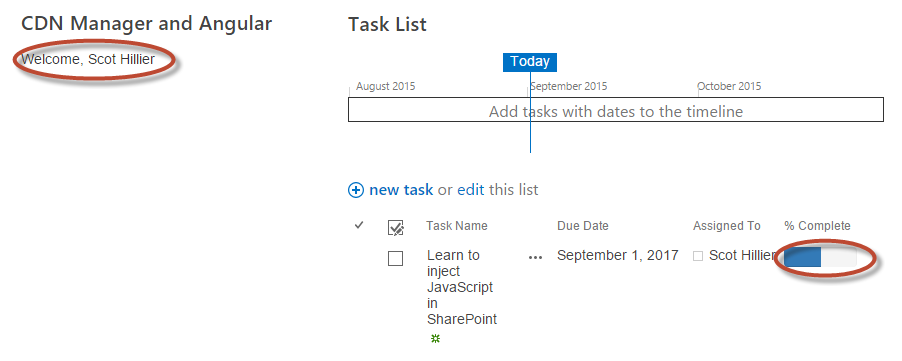
At this point in the exercise, the JSLink file has been used to modify the % Complete field display. However, the same file was also used as a “Trojan horse” to introduce an Angular controller. You can use this idea to introduce the controller to any page on the site where you can place a view of the Task list. Next, you will do that on the home page of the test site.

1. Add the Task list to the home page
   1. Return to the home page of the test site.
   2. Click **Page**.
   3. Click **Edit**.
   4. Select **Insert>Web Part**.
   5. Select the **Tasks** list and click **Add**.



* 1. Change the **Selected View** to **All Tasks**.
  2. Expand the **Miscellaneous** category in the web part properties dialog.
  3. Type **~sitecollection/\_catalogs/masterpage/progressbar.js** in the JS Link field.
  4. Click **OK**.

1. Update the Script Editor web part to call the injected service.
   * 1. Select **Edit Web Part** from the **Script Editor** web part.
     2. Click **Edit Snippet**.
     3. Remove all of the JavaScript code by deleting the <script> tags and everything within them.
     4. You should now see both the modified list view and the welcome message on the page.



### Exercise 4: Injecting Script through Custom Actions

In this exercise you will use a script custom action defined in a PowerShell script to inject JavaScript into the test site.

1. Install the SharePoint Online Management Shell.
   1. Go to <http://www.microsoft.com/en-us/download/details.aspx?id=35588> and download the SharePoint online Management shell.
   2. Run the download and follow the instructions.
2. Run the SharePoint Online Management shell as Administrator.
   1. Change directories to the location of the **AddScriptBlock.ps1** file.
   2. Execute the script by typing **./AddScriptBlock.ps1**.
   3. When prompted:
      1. Enter the **Site URL** for the test site.
      2. Enter your **username**.
      3. Enter your **password**.
3. Remove the Task list view from the home page
   1. Return to the home page of the test site.
   2. Select **Edit page** from the settings menu.
   3. **Delete** the Task list web part from the page.
   4. Note that the Script Editor still functions because the Powershell injected a ScriptLink block into the site.
4. Run the SharePoint Online Management shell as Administrator.
   1. Change directories to the location of the **RemoveScriptBlock.ps1** file.
   2. Execute the script by typing **./RemoveScriptBlock.ps1**.
   3. When prompted:
      1. Enter the **Site URL** for the test site.
      2. Enter your **username**.
      3. Enter your **password**.

Congratulations! You have finished using various techniques to inject JavaScript into a SharePoint site.