# Exercise 5: Creating a field customizer extension

## Task 1: Create your project

1. From the PowerShell command prompt, change to the C:/LabFiles/SharePoint directory by executing the following command: cd c:/LabFiles/SharePoint
2. Make a new directory for your SharePoint project files by executing the following command: md SPFxFieldCustomizer
3. Navigate to the newly created SharePoint directory by executing the following command: cd SPFxFieldCustomizer
4. Run the SharePoint Yeoman generator by executing the following command: yo @microsoft/sharepoint
5. Use the following to complete the prompt that is displayed:
   * **What is your solution name?**: SPFxFieldCustomizer
   * **Which baseline packages do you want to target for your component(s)?**: SharePoint Online only (latest)
   * **Where do you want to place the files?**: Use the current folder
   * **Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately without running any feature deployment or adding apps in sites?**: No
   * **Will the components in the solution require permissions to access web APIs that are unique and not shared with other components in the tenant?**: No
   * **Which type of client-side component to create?**: Extension
   * **What type of client-side extension to create?**: Field Customizer
   * **What is your Field Customizer name?**: HelloFieldCustomizer
   * **What is your Field Customizer description?**: HelloFieldCustomizer description
   * **Which framework would you like to use?**: No JavaScript Framework
6. After provisioning the folders required for the project, the generator will install all the dependency packages using NPM.

## Task 2: Updating the SCSS styles for the field customizer

1. Locate and open the **./src/extensions/helloFieldCustomizer/HelloFieldCustomizerFieldCustomizer.module.scss** file.
2. Replace the contents of the file with the following styles:

* .HelloFieldCustomizer {  
   .cell {  
   display: 'inline-block';  
   }  
   .filledBackground {  
   background-color: #cccccc;  
   width: 100px;  
   }

## Task 3: Update the code for the field customizer

1. Locate and open the **./src/extensions/helloFieldCustomizer/HelloFieldCustomizerFieldCustomizer.ts** file.
2. Locate the interface **IHelloFieldCustomizerFieldCustomizerProperties** and update its properties to the following: greenMinLimit?: string;yellowMinLimit?: string;
3. Locate the method **onRenderCell()** and update the contents to match the following code. This code looks at the existing value in the field and builds the relevant colored bars based on the value entered in the completed percentage value.

* event.domElement.classList.add(styles.cell);  
   event.domElement.innerHTML = `  
   <div class='${styles.HelloWorld}'>  
   <div class='${styles.full}'>  
   <div style='width: ${event.fieldValue}px; background:#0094ff; color:#c0c0c0'>  
   &nbsp; ${event.fieldValue}  
   </div>  
   </div>  
   </div>`;

## Task 4: Update the deployment code for the field customizer

Field customizers, when deployed to production, are implemented by creating a new site column who’s rendering is tied to the custom script defined in the field customizer’s bundle file.

1. Locate and open the **./sharepoint/assets/elements.xml** file.
2. Add the following property to the element, setting the values on the public properties on the field customizer:

* ClientSideComponentProperties="{&quot;greenMinLimit&quot;:&quot;85&quot;,&quot;yellowMinLimit&quot;:&quot;70&quot;}"

## Task 5: Testing your field customizer

1. In a browser, navigate to a SharePoint Online modern site collection where you want to test the field customizer.
2. Select the **Site contents** link in the left navigation pane.
3. Create a new SharePoint list:
   1. Select **New > List** in the toolbar.
   2. Set the list name to **Work Status** and select **Create**.
4. When the list loads, select the **Add column** > **Number** to create a new column.
5. When prompted for the name of the column, enter **PercentComplete**.
6. Add a few items to the list, with different numbers in the percent field.
7. Update the properties for the serve configuration used to test and debug the extension:
   1. Locate and open the **./config/serve.json** file.
   2. Copy in the full URL (including **AllItems.aspx**) of the list you just created into the **serveConfigurations.default.pageUrl** property.
   3. Locate the **serveConfigurations.default.properties** object.
   4. Change the name of the property **serveConfigurations.default.fieldCustomizers.InternalFieldName** to **serveConfigurations.default.fieldCustomizers.PercentComplete**. This tells the SharePoint Framework which existing field to associate the field customizer with.
   5. Change the value of the properties object to the following: "properties": { "greenMinLimit": "85", "yellowMinLimit": "70"}
   6. The JSON code for the default serve configuration should look something like the following:
   * "default": { "pageUrl": "https://contoso.sharepoint.com/sites/mySite/Lists/Work%20Status/AllItems.aspx", "fieldCustomizers": { "PercentComplete": { "id": "6a1b8997-00d5-4bc7-a472-41d6ac27cd83", "properties": { "greenMinLimit": "85", "yellowMinLimit": "70" } } }}
8. When prompted, select the **Load debug scripts** button.
9. Stop the local web server by pressing CTRL+C in the console/terminal window.

## Review

In this exercise, you learned how to create and deploy a command set extension. Command set extensions are used for adding additional commands into the command bar at the top of a page.