# Exercise 1: Introduction to SharePoint Framework (SPFx)

## Task 1: Set up your SharePoint Framework development environment

### Ensure Node.js is installed and is the correct supported version

1. Open the PowerShell command prompt execute the following command: node -v
2. Ensure the version is Node.js **v10.x** and is NOT 9.x nor 11.x (these two are not supported with the SharePoint Framework).

### Install Yeoman and gulp

1. From the command prompt execute the following command: npm install -g yo gulp
2. From the command prompt execute the following command: npm install -g @microsoft/generator-sharepoint

## Task 2: Creating a SharePoint Framework client-side web part

1. From the PowerShell command prompt, change to the C:/LabFiles directory by executing the following command: cd c:/LabFiles
2. Make a new directory for your SharePoint project files by executing the following command: md SharePoint
3. Navigate to the newly created SharePoint directory by executing the following command: cd SharePoint
4. Make a new directory for your SharePoint project files by executing the following command: md HelloWorld
5. Navigate to the newly created SharePoint directory by executing the following command: cd HelloWorld
6. Run the SharePoint Yeoman generator by executing the following command: yo @microsoft/sharepoint
7. Use the following to complete the prompt that is displayed:
   * **What is your solution name?**: HelloWorld
   * **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?**: WebPart
   * **What is your Web part name?**: HelloWorld
   * **What is your Web part description?**: HelloWorld description
   * **Which framework would you like to use?**: No JavaScript framework

### Trusting the self-signed developer certificate

1. From the PowerShell command prompt execute the following command: gulp trust-dev-cert
2. After provisioning the folders required for the project, the generator will install all the dependency packages using NPM for Node.js. When NPM completes downloading all dependencies, run the project by executing the following command: gulp serve
3. The SharePoint Framework’s gulp serve task will build the project, start a local web server, and launch a browser open to the SharePoint Workbench.
4. Select the + web part icon button to open the list of available web parts.
5. Select the **HelloWorld** web part.
6. Edit the web part’s properties by selecting the pencil (edit) icon in the toolbar to the left of the web part.
7. In the property pane that opens, change the value of the **Description** field. Notice how the web part updates as you make changes to the text.
8. Go back to PowerShell and stop the running process by executing **Ctrl + C**
9. When prompted to terminate type Y and hit the enter key.

## Task 3: Test with the local and hosted SharePoint Workbench

In this exercise you will work with two different versions of the SharePoint Workbench, the local and hosted workbench, as well as the different modes of the built-in gulp serve task.

1. Open the project in Visual Studio Code by executing the following command: code .
2. From the Visual Studio Code ribbon, select **Terminal > New Terminal**.
3. Run the project by executing the following command: gulp serve
4. Leave the project running. Go back to Visual Studio Code.
5. Edit the HTML in the web part’s **render()** method, located in the **src/webparts/helloWorld/HelloWorldWebPart.ts** file.
   1. Locate the style with the class styles.title and update change the title to “**Welcome to Microsoft Learning for SharePoint!**”
   2. Save the project and then navigate back to the browser and notice your web part title value changes.
6. Next, in the browser navigate to one of your SharePoint Online sites and append the following to the end of the root site’s URL: \*\*/\_layouts/workbench.aspx\*\*. This is the SharePoint Online hosted workbench.
7. Notice that when you add a web part, many more web parts will appear beyond the one you created; and was the only one showing in the toolbox on the local workbench. This is because you are now testing the web part in a working SharePoint site.

* **Note**: The difference between the local and hosted workbench is significant. Consider that a local workbench is not a working version of SharePoint; rather, it’s just a single page that can let you test web parts. This means you won’t have access to a real SharePoint context, lists, libraries, or real users when you are testing in the local workbench.

1. Let’s see another difference with the local versus hosted workbench. Go back to the web part and make a change to the HTML.
2. Notice that, after saving the file, while the console displays a lot of commands, the browser that is displaying the hosted workbench does not automatically reload. This is expected. You can still refresh the page to see the updated web part, but the local web server cannot cause the hosted workbench to refresh. Close both the local and hosted workbench and stop the local web server by pressing CTRL+C in the command prompt.

## Review

In this exercise, you learned how to create a SharePoint Framework web part and test the web part using the SharePoint Framework Workbench and SharePoint site.