

Introduction to the SharePoint Framework



Agenda

- Introduction to the SharePoint Framework
- Creating SPFX Projects using the Yeoman Generator
- Testing & Debugging Webparts in SharePoint Workbench
- Managing Styles using SCSS Files and CSS Modules
- Creating a Web Part with Custom Properties
- Creating Application Customizers



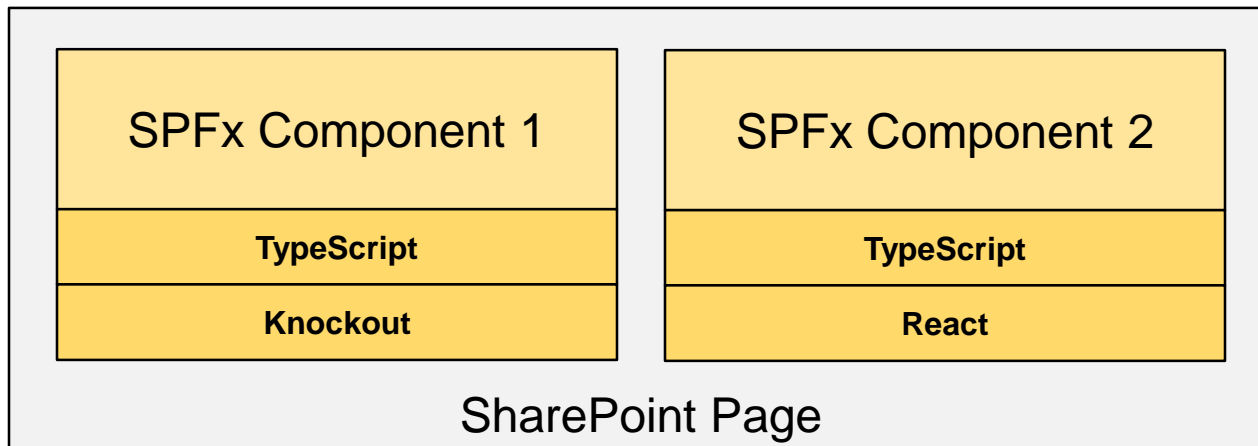
Evolution of the SharePoint Platform

- Farm Solutions
 - Server-side DLLs and XML-based Definitions
- ~~Sandboxed Solutions~~
- SharePoint ~~Apps~~ Add-ins
 - iFrames used to add in extra security dimension
 - Introduced complexity with 2 domains (app web vs host web)
- JavaScript Injection
 - Scripting can be disabled in SharePoint Online
 - No formal deployment model
- SharePoint Framework
 - A natural evolution and formalization of JavaScript Injection model



SharePoint Framework (SPFx)

- Getting Started with the SharePoint Framework
 - You write your logic using client-side TypeScript code
 - You can use any JavaScript libraries (e.g. React, Knockout, etc.)
 - Your code runs under the identity of the current user
 - You don't worry about authenticating the user - it's already done
 - You can leverage the APIs added by SharePoint Framework
 - You create components like webparts and application customizers



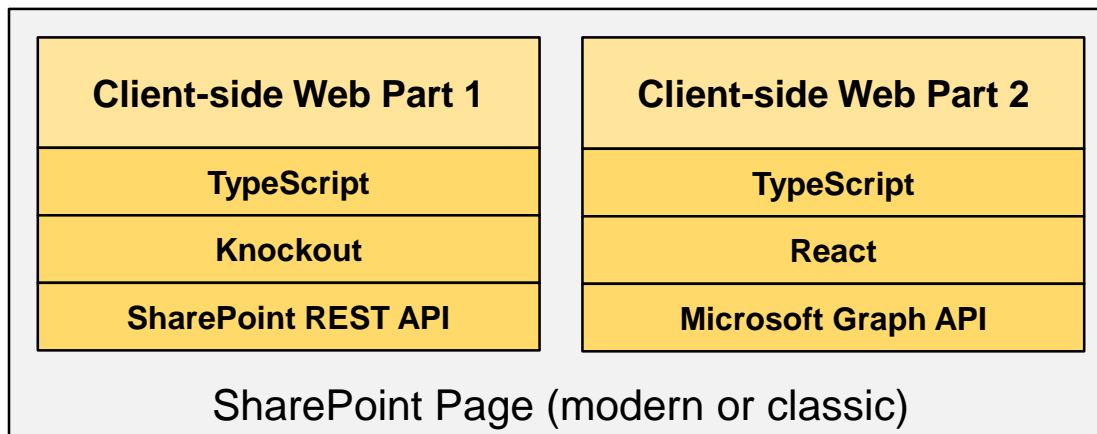
SPFx versus the SharePoint Add-in Model

- SPFx is quite different from SharePoint Add-in model
 - SPFx components hosted directly on page, not in iFrame
 - SPFx components rendered using DOM of hosting page
 - No more confusion over "host web" versus "app web"
- SPFx developer experience is completely different
 - SPFx uses modern tools (npm, Yeoman, gulp and webpack)
 - Requires move from Visual Studio to Node.js & Visual Studio Code
- Considerations for migrating to SharePoint Framework
 - SPFx is replacement for SharePoint-hosted add-in model
 - SPFx has nothing similar to provided-hosted add-in model



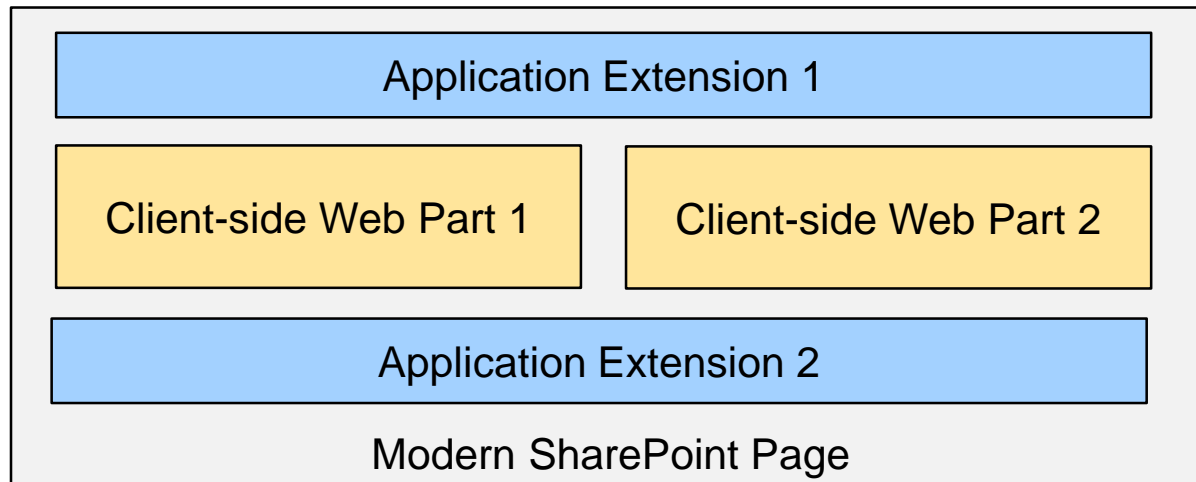
SharePoint Framework Webparts

- Webparts play central role in SharePoint Framework development
 - SPFx webparts designed to run on modern pages
 - SPFx webparts can also be added to classic pages
- Developing Webparts
 - You create webparts using classes defined in Typescript
 - Webpart class inherits from base class defined by SPFx APIs
 - SPFx support webpart lifecycle methods (e.g. render, load, serialize. etc.)
 - SPFx APIs provide easy access to content in SharePoint and Office 365



SharePoint Framework Component Types

- SPFx allows you to create several styles of webparts
 - Standard Webparts
 - React Webparts
- SPFx also provides several other Application Extensions
 - Application Customizer
 - Field Customizers
 - Command Sets



Installing Packages for SPFx Development

- Install Gulp (version 3)

```
npm install -g gulp@3
```

- Install Yeoman

```
npm install -g yo
```

- Install Yeoman Template for SPFx

```
npm install -g @microsoft/generator-sharepoint
```



Agenda

- ✓ Introduction to the SharePoint Framework
- Creating SPFX Projects using the Yeoman Generator
 - Testing & Debugging Webparts in SharePoint Workbench
 - Managing Styles using SCSS Files and CSS Modules
 - Creating a Web Part with Custom Properties
 - Creating Application Customizers

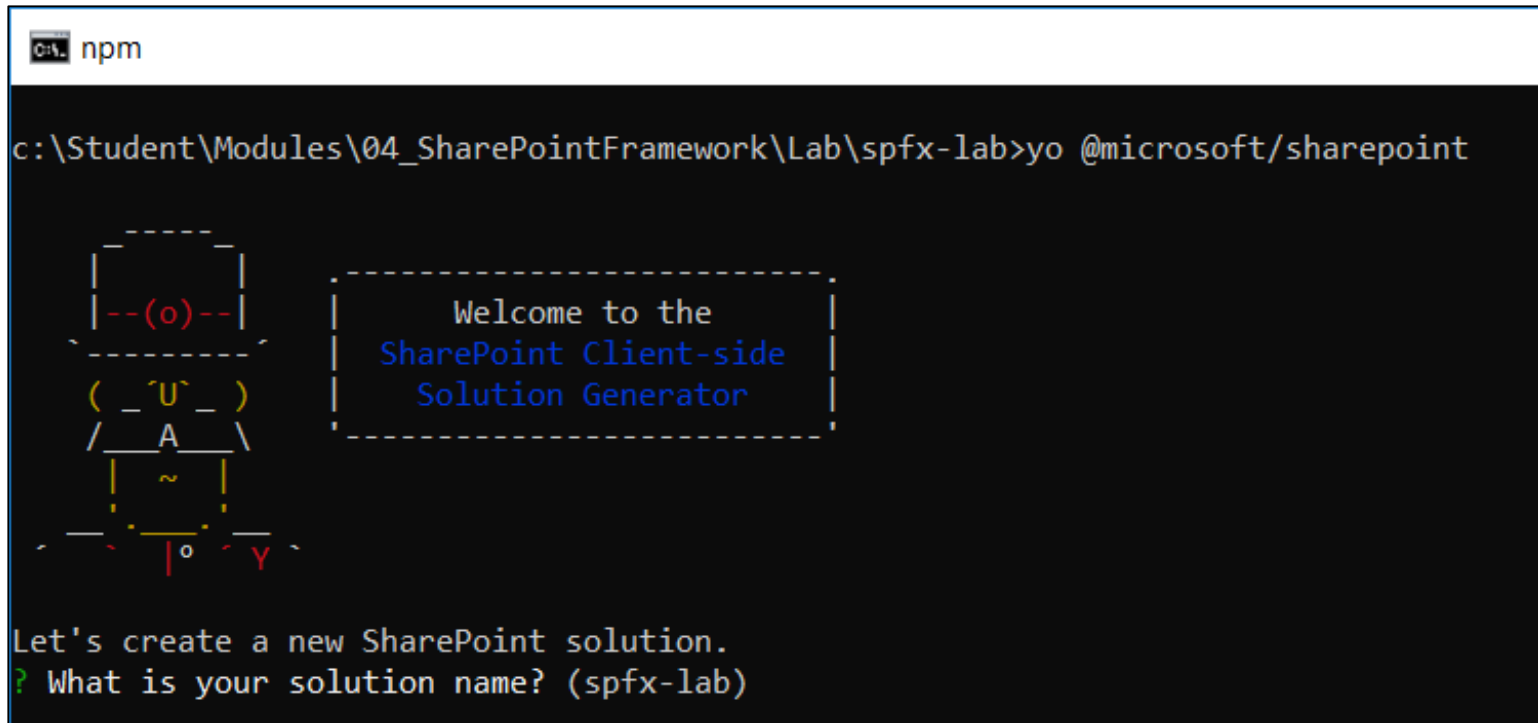


Using the SPFx Yeoman Template

- SPFx projects created with Yeoman template

```
yo @microsoft/sharepoint
```

- Template provides wizard-like experience when creating new project



The screenshot shows a terminal window with the title 'C:\ npm'. The command prompt shows the command `yo @microsoft/sharepoint` being executed in the directory `c:\Student\Modules\04_SharePointFramework\Lab\spfx-lab`. The output displays a ASCII art logo for the SharePoint Client-side Solution Generator, which includes a stylized 'U' and 'A' with a tilde '~' below them. To the right of the logo, a dashed box contains the text 'Welcome to the SharePoint Client-side Solution Generator'. Below the logo and text, the prompt 'Let's create a new SharePoint solution.' is followed by the question '? What is your solution name? (spfx-lab)'.

```
C:\ npm
c:\Student\Modules\04_SharePointFramework\Lab\spfx-lab>yo @microsoft/sharepoint

  --(o)--
  ( _U_ )
  /   A   \
  |   ~   |
  |   o   |
  |   Y   |

Welcome to the
SharePoint Client-side
Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? (spfx-lab)
```



Answering Questions about a New Project

- Do you want to support SharePoint On-premises?

```
Let's create a new SharePoint solution.  
? What is your solution name? spfx-lab  
? Which baseline packages do you want to target for your component(s)? (Use arrow keys)  
> SharePoint Online only (latest)  
   SharePoint 2016 onwards, including SharePoint Online
```

- Do you want to create a webpart or an SPFx extension

```
? Which type of client-side component to create? (Use arrow keys)  
> WebPart  
   Extension
```

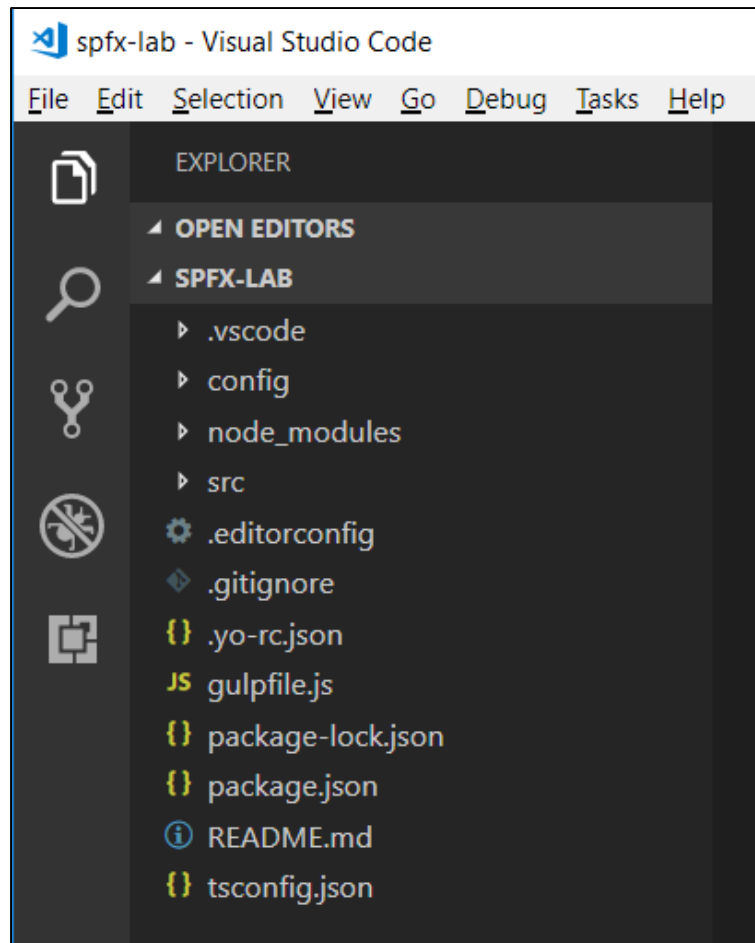
- Do you want to create a standard webpart or a React webpart

```
Add new Web part to solution spfx-lab.  
? What is your Web part name? WalmartGreeter  
? What is your Web part description? My first SPFX webpart  
? Which framework would you like to use? (Use arrow keys)  
> No JavaScript framework  
   React  
   Knockout
```



SharePoint Framework Project Structure

- Project created as Node.js project



SharePoint Framework Adds Gulp Tasks

- Run **gulp --tasks** to see SPFx gulp tasks added to project

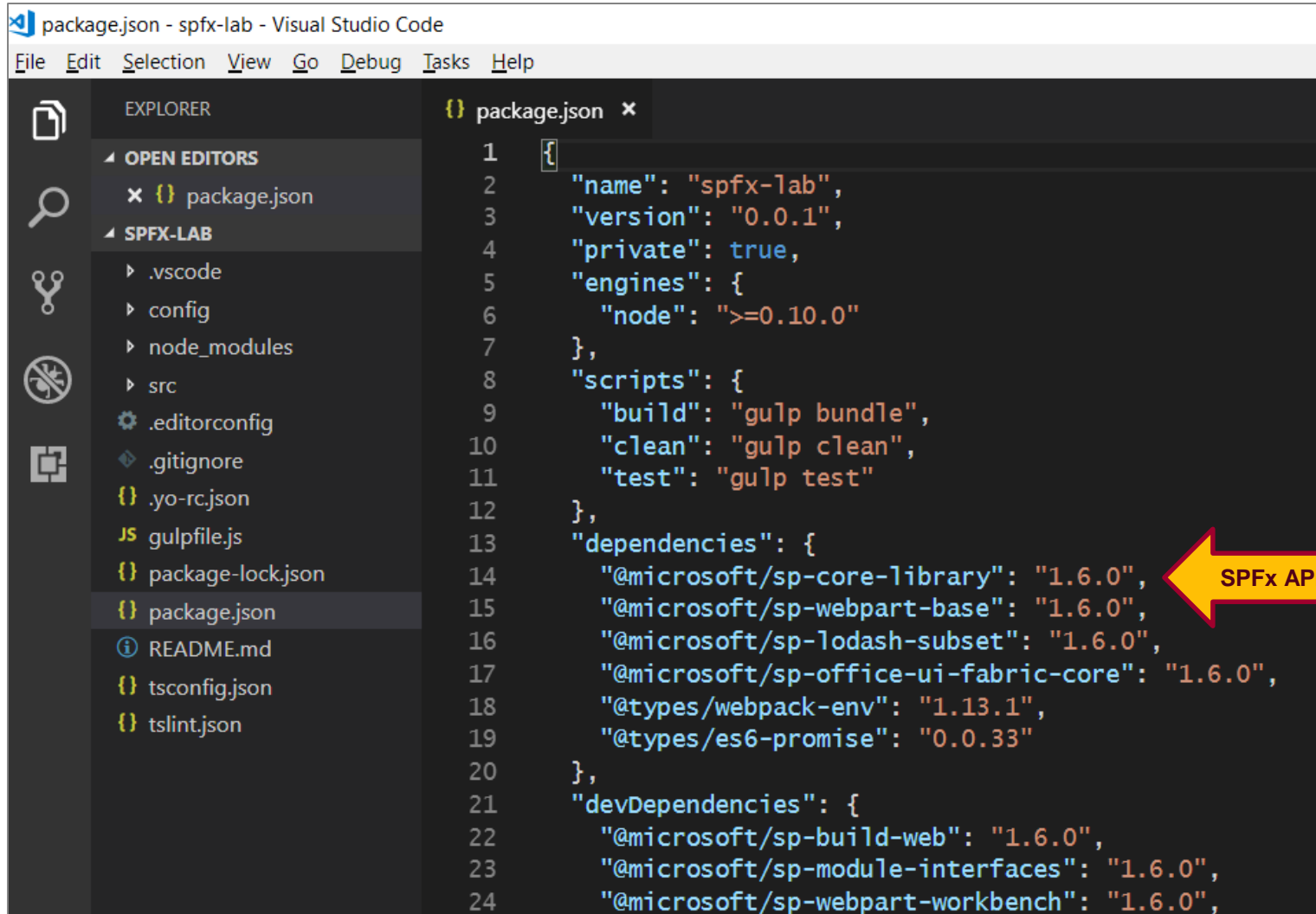
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab> gulp --tasks
[00:05:07] Using gulpfile C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab\gulpfile.js
[00:05:07] Tasks for C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab\gulpfile.js
[00:05:07] |— clean
[00:05:07] |— build
[00:05:07] |— default
[00:05:07] |— bundle
[00:05:07] |— dev-deploy
[00:05:07] |— deploy-azure-storage
[00:05:07] |— package-solution
[00:05:07] |— test
[00:05:07] |— serve
[00:05:07] |— trust-dev-cert
[00:05:07] |— untrust-dev-cert
PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab> █
```



Package.json



The screenshot shows the Visual Studio Code interface with the 'package.json' file open. The Explorer sidebar on the left lists the project files, including 'package.json'. The main editor area displays the JSON content of 'package.json' with line numbers 1 through 24. The JSON structure includes fields for 'name', 'version', 'private', 'engines', 'scripts', 'dependencies', and 'devDependencies'. A yellow arrow points from the text 'SPFx API Version Number' to the '1.6.0' version string in the '@microsoft/sp-core-library' dependency.

```
1  {
2    "name": "spfx-lab",
3    "version": "0.0.1",
4    "private": true,
5    "engines": {
6      "node": ">=0.10.0"
7    },
8    "scripts": {
9      "build": "gulp bundle",
10     "clean": "gulp clean",
11     "test": "gulp test"
12   },
13   "dependencies": {
14     "@microsoft/sp-core-library": "1.6.0",
15     "@microsoft/sp-webpart-base": "1.6.0",
16     "@microsoft/sp-lodash-subset": "1.6.0",
17     "@microsoft/sp-office-ui-fabric-core": "1.6.0",
18     "@types/webpack-env": "1.13.1",
19     "@types/es6-promise": "0.0.33"
20   },
21   "devDependencies": {
22     "@microsoft/sp-build-web": "1.6.0",
23     "@microsoft/sp-module-interfaces": "1.6.0",
24     "@microsoft/sp-webpart-workbench": "1.6.0",
```

SPFx API Version Number



Running gulp trust-dev-cert

- Testing SPFx code requires self-signed certificate
 - Certificate used serve pages with SSL at <https://localhost>
 - Certificate created and registered using **gulp trust-dev-cert**

```
PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab> gulp trust-dev-cert
Build target: DEBUG
[07:51:33] Using gulpfile C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab\gulpfile.js
[07:51:33] Starting gulp
[07:51:33] Starting 'trust-dev-cert'...
[07:51:33] Starting subtask 'configure-sp-build-rig'...
[07:51:33] Finished subtask 'configure-sp-build-rig' after 4.71 ms
[07:51:33] Starting subtask 'trust-cert'...
[07:51:33] Finished subtask 'trust-cert' after 67 ms
[07:51:33] Finished 'trust-dev-cert' after 73 ms
[07:51:33] =====[ Finished ]=====
[07:51:34] Project spfx-lab version:0.0.1
[07:51:34] Build tools version:3.7.4
[07:51:34] Node version:v8.11.4
[07:51:34] Total duration:3.28 s
PS C:\Student\Modules\04_SharePointFramework\Lab\spfx-lab>
```

- **gulp trust-dev-cert** must be run within project directory
 - However, you only have to run this command once
 - No need to run **gulp trust-dev-cert** on a per-project basis



Agenda

- ✓ Introduction to the SharePoint Framework
- ✓ Creating SPFX Projects using the Yeoman Generator
- Testing & Debugging Webparts in SharePoint Workbench
 - Managing Styles using SCSS Files and CSS Modules
 - Creating a Web Part with Custom Properties
 - Creating Application Customizers



The "Hello World" SPFx Webpart

- Webpart class must extend BaseClientSideWebPart
 - Override render() for minimal “hello world” functionality
 - Base class provides API though **context** and **pageContext**
 - Base class provides **domElement** to access hosting page DOM

```
TS WalmartGreeterWebPart.ts •
import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';

export default class WalmartGreeterWebPart extends BaseClientSideWebPart<any> {

  public render(): void {

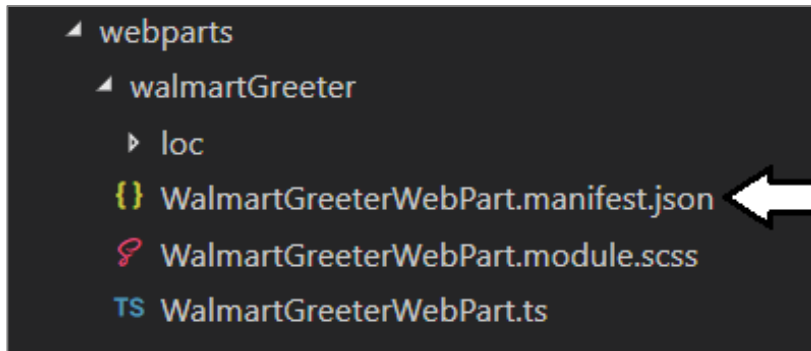
    let currentUser = this.context.pageContext.user.displayName;

    this.domElement.innerHTML = `
      <div>
        <h2>Hello ${currentUser}</h2>
      </div>`;
  }
}
```



Webpart Manifest

- Each webpart requires its own manifest file
 - Manifest file automatically added by SPFx Yeoman template



- Update manifest to set webpart title and icon

```
"preconfiguredEntries": [{  
  "groupId": "5c03119e-3074-46fd-976b-c60198311f70",  
  "group": { "default": "Other" },  
  "title": { "default": "Walmart Greeter" },  
  "description": { "default": "My first SPFX webpart" },  
  "officeFabricIconFontName": "Emoji2",  
  "properties": {  
    "description": "WalmartGreeter"  
  }  
}]  
}
```



Web Part Context

```
public render(): void {

    var container = jquery(this.domElement);
    container.append( jquery("<h2>").text("Web Part Context Demo") );

    var table: JQuery = this.CreateTable();
    this.AddTableRow(table, "site.id:", this.context.pageContext.site.id.toString());
    this.AddTableRow(table, "web.id:", this.context.pageContext.web.id.toString());
    this.AddTableRow(table, "web.title:", this.context.pageContext.web.title);
    this.AddTableRow(table, "web.absoluteUrl:", this.context.pageContext.web.absoluteUrl);
    this.AddTableRow(table, "web.serverRelativeUrl:", this.context.pageContext.web.serverRelativeUrl);
    this.AddTableRow(table, "web.templateName:", this.context.pageContext.web.templateName);
    this.AddTableRow(table, "web.currentCultureName:", this.context.pageContext.cultureInfo.currentCultureName);
    this.AddTableRow(table, "web.language:", this.context.pageContext.web.language.toString());
    this.AddTableRow(table, "user.displayName:", this.context.pageContext.user.displayName);
    this.AddTableRow(table, "user.loginName:", this.context.pageContext.user.loginName);
    this.AddTableRow(table, "user.email:", this.context.pageContext.user.email);
    this.AddTableRow(table, "this.diplyMode:", this.displayMode.toString());
    this.AddTableRow(table, "context.webPartTag:", this.context.webPartTag);
    container.append(table);
}
```

Property	Value
site.id:	a5aa0f03-16b6-4057-8704-daaa2f84494
web.id:	b68b2b24-63c2-42af-a10b-fabb37c034f3
web.title:	Labs for CBD365 Team Site
web.absoluteUrl:	https://labsforcbd365.sharepoint.com
web.serverRelativeUrl:	/
web.templateName:	1
web.currentCultureName:	en-US
web.language:	1033
user.displayName:	Ted Pattison
user.loginName:	student@labsforcbd365.onmicrosoft.com
user.email:	
this.diplyMode:	2
context.webPartTag:	WebPart.InspectorWebPart.eaf44355-2d45-4e1c-b8de-e8b3bce60279



DEMO

Hello World with SPFx

Agenda

- ✓ Introduction to the SharePoint Framework
- ✓ Creating SPFX Projects using the Yeoman Generator
- ✓ Testing & Debugging Webparts in SharePoint Workbench
- Managing Styles using SCSS Files and CSS Modules
 - Creating a Web Part with Custom Properties
 - Creating Application Customizers



Issues with CSS in Web Development

- CSS can be hard to manage in large applications
 - Global style names can conflict with one another
 - Component CSS should not affect other parts of page
 - Component CSS should be isolated
 - You should avoid using element IDs in CSS
 - Prefer using classes instead of IDs
 - You should create class names unique across page

Bad, Bad, Bad - do not use IDs in a webpart

```
public render(): void {  
  this.domElement.innerHTML = `  
    <div id="myWebPart" >  
      <div id="myTitle" >Hello World</div>  
    </div>`;   
}
```



Working with SASS and .SCSS Files

- SPFx uses Syntactically Awesome Style Sheets (SASS)
 - Styles maintained in .scss files instead of .css files
 - SASS is superset of CSS with variables, selector nesting & mixins
 - SASS compilation occurs when you build project using **gulp build**
 - Webpack compiles .scss files into .css files
- SASS compilation generates unique style names
 - **helloWebPart** renamed to **helloWebPart_0989818e**

```
Hello.module.scss x
1  $font-stack:    Helvetica, sans-serif;
2  $background-color: #ffffe0;
3  $font-size: 3.0em;
4  $padding: 18px;
5
6  .helloWebPart{
7    font: $font-stack;
8    font-size: $font-size;
9    background-color: $background-color;
10   border: 1px solid black;
11   border-radius: $padding;
12   padding: $padding;
13   text-align: center;
14 }
```

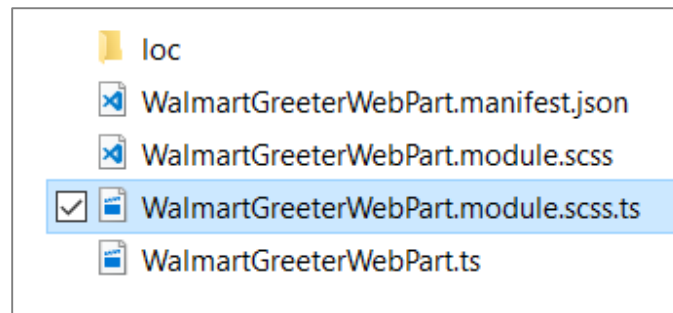
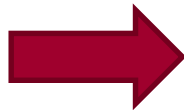
SASS

```
Hello.module.css •
1  .helloWebPart_0989818e{
2    font:Helvetica,sans-serif;
3    font-size:3em;
4    background-color:#ffffe0;
5    border:1px solid black;
6    border-radius:18px;
7    padding:18px;
8    text-align:center}
```

SCSS Compilation Generates TypeScript File

- SASS compilation also generates TypeScript file
 - Used to provide strongly-types style names in TypeScript code

```
WalmartGreeterWebPart.module.scss x
1  .walmartGreeter {
2
3  .container {
4    max-width: 700px;
5    margin: 0px auto;
6  }
7
8  .title {
9    font-size: 24px;
10   color: darkblue;
11 }
12
13 }
```



```
TS WalmartGreeterWebPart.module.scss.ts x
1  /* tslint:disable */
2  require('./WalmartGreeterWebPart.module.css');
3  const styles = {
4    walmartGreeter: 'walmartGreeter_d498b2d0',
5    container: 'container_d498b2d0',
6    title: 'title_d498b2d0',
7  };
8
9  export default styles;
10 /* tslint:enable */
```

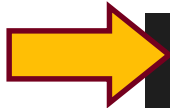
Importing Style Names in a TypeScript File

- You can run **gulp build** to force SASS compilation
 - **import** statement to .scss file displays error until you run **gulp build**

```
import styles from './WalmartGreeterWebPart.module.scss';
```



- Once compiled, SCSS styles names can be referenced in TypeScript



```
import styles from './WalmartGreeterWebPart.module.scss';

export default class WalmartGreeterWebPart extends BaseClientComponent {

  public render(): void {
    this.domElement.innerHTML = `
      <div class="${styles.container}" >
        <div class="${styles.title}" >Hello World</div>
      </div>`;
  }
}
```

A red dashed rectangular box highlights the `styles` object in the code, specifically around the `styles.container` and `styles.title` references.

- container
- title
- walmartGreeter



Agenda

- ✓ Introduction to the SharePoint Framework
- ✓ Creating SPFX Projects using the Yeoman Generator
- ✓ Testing & Debugging Webparts in SharePoint Workbench
- ✓ Managing Styles using SCSS Files and CSS Modules
- Creating a Web Part with Custom Properties
 - Creating Application Customizers



Web Part Properties

- Define interface with properties

```
IGreeterWebpartWebPartProps.ts •  
1  export interface IGreeterWebpartWebPartProps {  
2      greeting: string;  
3      largefont: boolean;  
4      color: string;  
5  }
```

- Add interface to web part class definition

```
class GreeterWebpartWebPart extends BaseClientSideWebPart<IGreeterWebpartWebPartProps> {
```

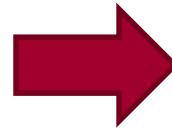
- Override getPropertyPaneConfiguration()

```
protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {  
    return {  
        pages: [  
            {  
                header: { description: "Greeter Web Part" },  
                groups: [  
                    {  
                        groupName: "General Properties",  
                        groupFields: [  
                            PropertyPaneTextField('greeting', { label: 'Greeting' }),  
                        ],  
                    },  
                ],  
            },  
        ],  
    };  
}
```



Property Panel Settings

```
protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
  return {
    pages: [ {
      header: { description: "Greeter Web Part" },
      groups: [ {
        groupName: "General Properties",
        groupFields: [
          PropertyPaneTextField('greeting', { label: 'Greeting' }),
        ]
      },
      {
        groupName: "Cosmetic Properties",
        groupFields: [
          PropertyPaneToggle('fontBold', {
            label: 'Font Bold',
            onText: 'On',
            offText: 'Off'
          }),
          PropertyPaneDropdown('fontType', {
            label: 'Font Type',
            options: [
              { key: 'Arial', text: 'Arial' },
              { key: 'Courier', text: 'Courier' },
              { key: 'Verdana', text: 'Verdana' }
            ]
          }),
          PropertyPaneSlider("fontSize", {
            label: "Font Size",
            min: 24,
            max: 64
          }),
        ]
      }
    ]
  }
}
```



Walmart Greeter

Greeter Web Part

General Properties

Greeting

Welcome to Walmart

Cosmetic Properties

Large Font

☐ Off

Font Color

Blue





DEMO

Web Part Properties

Agenda

- ✓ Introduction to the SharePoint Framework
- ✓ Creating SPFX Projects using the Yeoman Generator
- ✓ Testing & Debugging Webparts in SharePoint Workbench
- ✓ Managing Styles using SCSS Files and CSS Modules
- ✓ Creating a Web Part with Custom Properties
- Creating Application Customizers



SPFx Application Extensions

- Application Customizers
 - Used to add page header and/or page footer into modern pages
 - Application customizers not supported in classic pages
- Field Customizers
 - Used to add client-side behavior on top of site columns
 - Allows you to create custom field rendering experience
- Command Sets
 - Allows you to add custom commands into SharePoint UI
 - SPFx component for creating user custom actions
 - Users invoke commands which trigger your client-side code



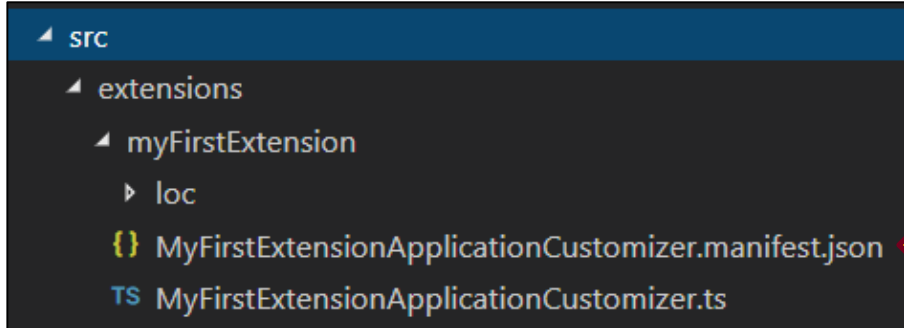
Creating an Application Customizer

- Yeoman templates support creating application extension
 - You can choose between the 3 types of application extensions

```
Let's create a new SharePoint solution.  
? What is your solution name? spfx-extension-lab  
? Which baseline packages do you want to target for your component(s)? SharePoint  
? Where do you want to place the files? Use the current folder  
Found npm version 5.6.0  
? Do you want to allow the tenant admin the choice of being able to deploy the s  
running any feature deployment or adding apps in sites? No  
? Which type of client-side component to create? Extension  
? Which type of client-side extension to create? Application Customizer  
Add new Application Customizer to solution spfx-extension-lab.  
? What is your Application Customizer name? MyFirstExtension  
? What is your Application Customizer description? My first extension_
```



The Application Customizer Manifest



```
{ MyFirstExtensionApplicationCustomizer.manifest.json •
1  {
2    "$schema": "https://developer.microsoft.com/json-schemas/spfx/client-side",
3    "id": "92700aa0-d156-4465-b4b3-aaf865e6e086",
4    "alias": "MyFirstExtensionApplicationCustomizer",
5    "componentType": "Extension",
6    "extensionType": "ApplicationCustomizer",
7    "version": "*",
8    "manifestVersion": 2,
9    "requiresCustomScript": false
10 }
```

ApplicationCustomizerInfo.txt - Notepad

File Edit Format View Help

Application Customizer ID:

92700aa0-d156-4465-b4b3-aaf865e6e086



Implementing an Application Customizer

TS MyFirstExtensionApplicationCustomizer.ts

```
import {
  BaseApplicationCustomizer,
  PlaceholderContent,
  PlaceholderName
} from '@microsoft/sp-application-base';

import styles from './MyApplicationCustomizerStyles.module.scss'

export default class MyFirstExtensionApplicationCustomizer
  extends BaseApplicationCustomizer<any> {

  private PageHeader: PlaceholderContent | undefined;
  private PageFooter: PlaceholderContent | undefined;

  @override
  public onInit(): Promise<void> {
    this.context.placeholderProvider.changedEvent.add(this, this.RenderPlaceHolders);
    this.RenderPlaceHolders();
    return Promise.resolve<void>();
  }

  private RenderPlaceHolders(): void { ...
  }

}
```



Rendering Content into Placeholders

```
private RenderPlaceholders(): void {

    if (!this.PageHeader) {
        this.PageHeader = this.context.placeholderProvider.tryCreateContent(PlaceholderName.Top);
        if (!this.PageHeader) {
            console.error('The expected placeholder (Top) was not found.');
```

```
            return;
        }
        this.PageHeader.domElement.innerHTML = `
<div class="${styles.app}">
  <div class="${styles.top}">
    <div>This is the page header</div>
  </div>
</div>`;
    }

    if (!this.PageFooter) {
        this.PageFooter = this.context.placeholderProvider.tryCreateContent(PlaceholderName.Bottom);
        if (!this.PageFooter) {
            console.error('The expected placeholder (Bottom) was not found.');
```

```
            return;
        }
        this.PageFooter.domElement.innerHTML = `
<div class="${styles.app}">
  <div class="${styles.bottom}">
    <div>This is the page footer</div>
  </div>
</div>`;
    }
}
```



Debugging an Application Customizer

- You must build debug URL with query string parameters
 - You must add GUID which is application customizer ID



A screenshot of a Notepad window titled "ApplicationCustomizerInfo.txt - Notepad". The window contains the following text:

```
File Edit Format View Help
Application Customizer ID: 92700aa0-d156-4465-b4b3-aaf865e6e086

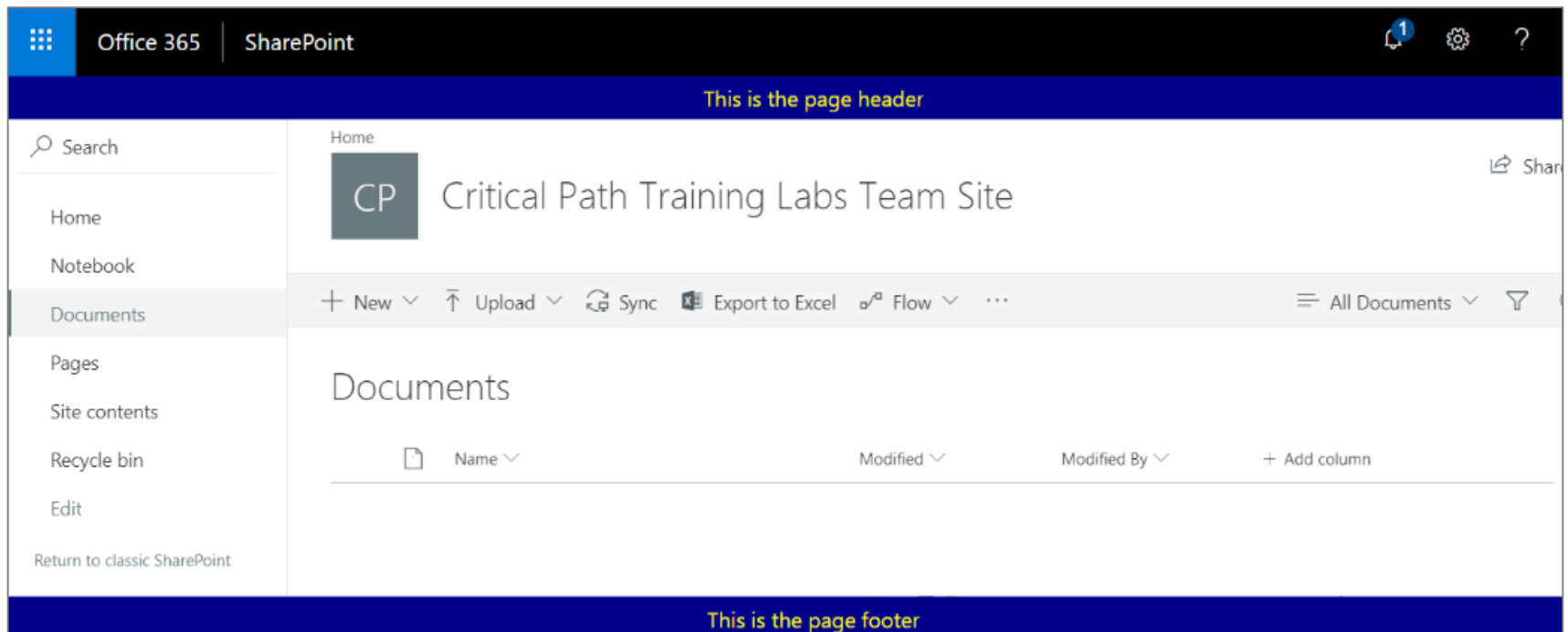
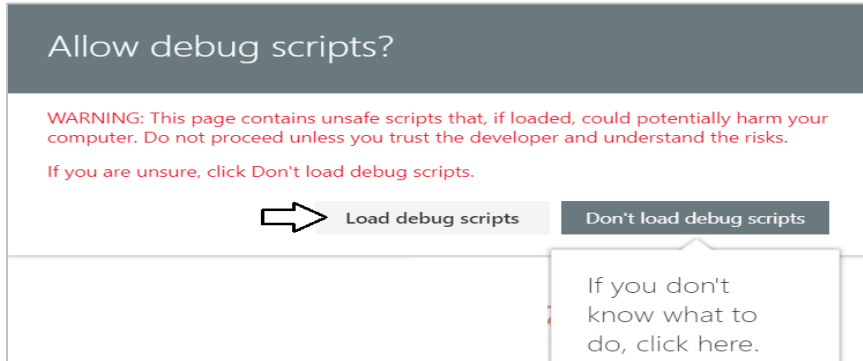
?loadSPFX=true
&debugManifestsFile=https://localhost:4321/temp/manifests.js
&customActions={"92700aa0-d156-4465-b4b3-aaf865e6e086":{
    "location":"ClientSideExtension.ApplicationCustomizer",
    "properties":{"testMessage":"Hello as property!"}}}
```

An arrow points to the GUID "92700aa0-d156-4465-b4b3-aaf865e6e086" in the customActions object.

- Next, you copy and paste URL into browser address bar



Testing the Application Customizer



Summary

- ✓ Introduction to the SharePoint Framework
- ✓ Creating SPFX Projects using the Yeoman Generator
- ✓ Testing & Debugging Webparts in SharePoint Workbench
- ✓ Managing Styles using SCSS Files and CSS Modules
- ✓ Creating a Web Part with Custom Properties
- ✓ Creating Application Customizers

