

Introduction to the SharePoint Framework



Agenda

- Creating SPFX Projects using the Yeoman Generator
- Testing & Debugging Projects in SharePoint Workbench
- Creating Application Customizers
- Creating Field Customizers and Command Sets.
- Creating a Web Part with Custom Properties
- Managing Styles using SCSS Files and CSS Modules



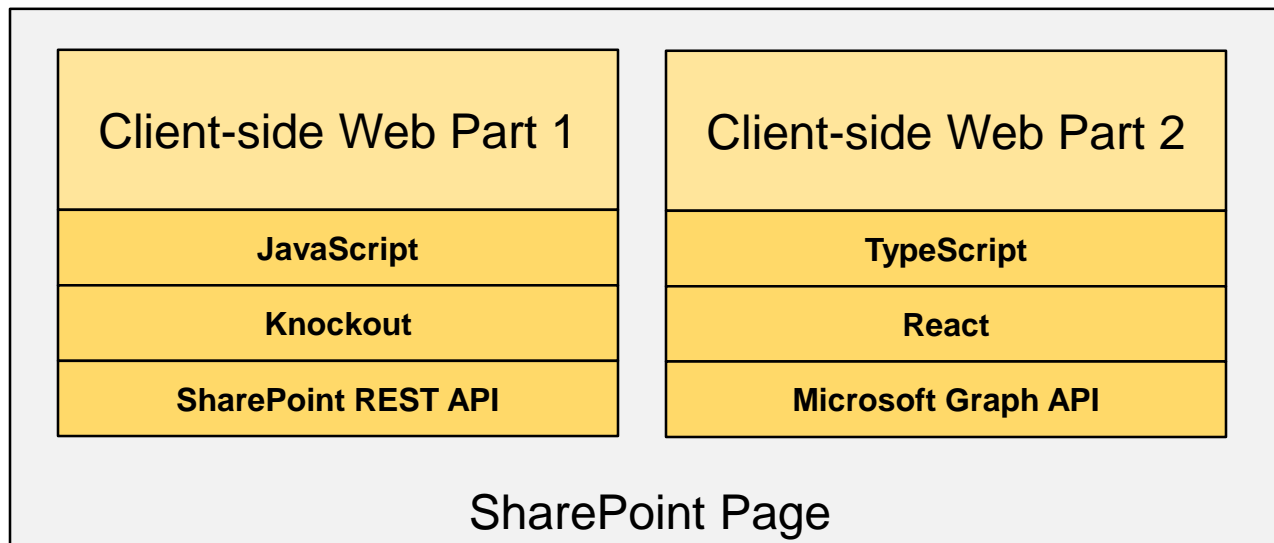
Evolution of the SharePoint Platform

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



What is SPFx?

- Development model based on pages and web parts
 - Based on client-side development with JavaScript or TypeScript
 - Code runs with authenticated identity of current user
 - Easy access to SharePoint and Office 365 content and data
 - Developer tools designed to support cross-platform development
 - Great support for targeting mobile devices



How Does SPFx Work?

- No more iFrames
 - Code runs the context of the current page
- Code runs with identity and permissions of user
 - Uses open browser connections for current user
- Supports lifecycle events
 - render, load, serialize, deserialize, etc.
- Use whatever JavaScript framework you want
 - React, Handlebars, Knockout, Angular1, Angular2, D3



Cross-platform Toolchain

- Node.js
- Node Package Manager (npm)
- TypeScript
- Yeoman
- Webpack
- Gulp
- git



Agenda

- ✓ Overview the SharePoint Framework (SPFx)
- Setting up an SPFx Development Environment
 - Creating Projects using the SPFx Templates
 - Deploying SPFx Projects using an Azure CDN



Install the SPFx Developer Toolchain

- Install Node.JS
 - Version 5.0 recommended - 4.0+ minimum
 - Installs Node Package Manage (npm)
- Install Visual Studio Code
 - Better environment for Development with Node.js
- Install Local self-signed certificate



Working with npm

- Windows Build Tools (*Visual C++ Build Tools 2015*)
npm install -g --production windows-build-tools
- Install Gulp
npm install -g gulp
- Install Yeoman
npm install -g yo
- Install Yeoman Template for SPFx
npm install -g @microsoft/generator-sharepoint



Agenda

- ✓ Overview the SharePoint Framework (SPFx)
- ✓ Setting up an SPFx Development Environment
- Creating Projects using the SPFx Templates
 - Debugging with the SharePoint Workbench
 - Developing SPFx Web Parts using React.js
 - Deploying SPFx Projects using an Azure CDN



Using the SPFx Yeoman Template

- SPFx projects created with Yeoman template
 - `yo @microsoft/sharepoint`
 - Takes 8-10 minutes to complete
 - Create a directory with over 200MB of source files

The image is a collage of terminal screenshots illustrating the process of creating an SPFx project using the Yeoman template. The main terminal window shows the following commands and output:

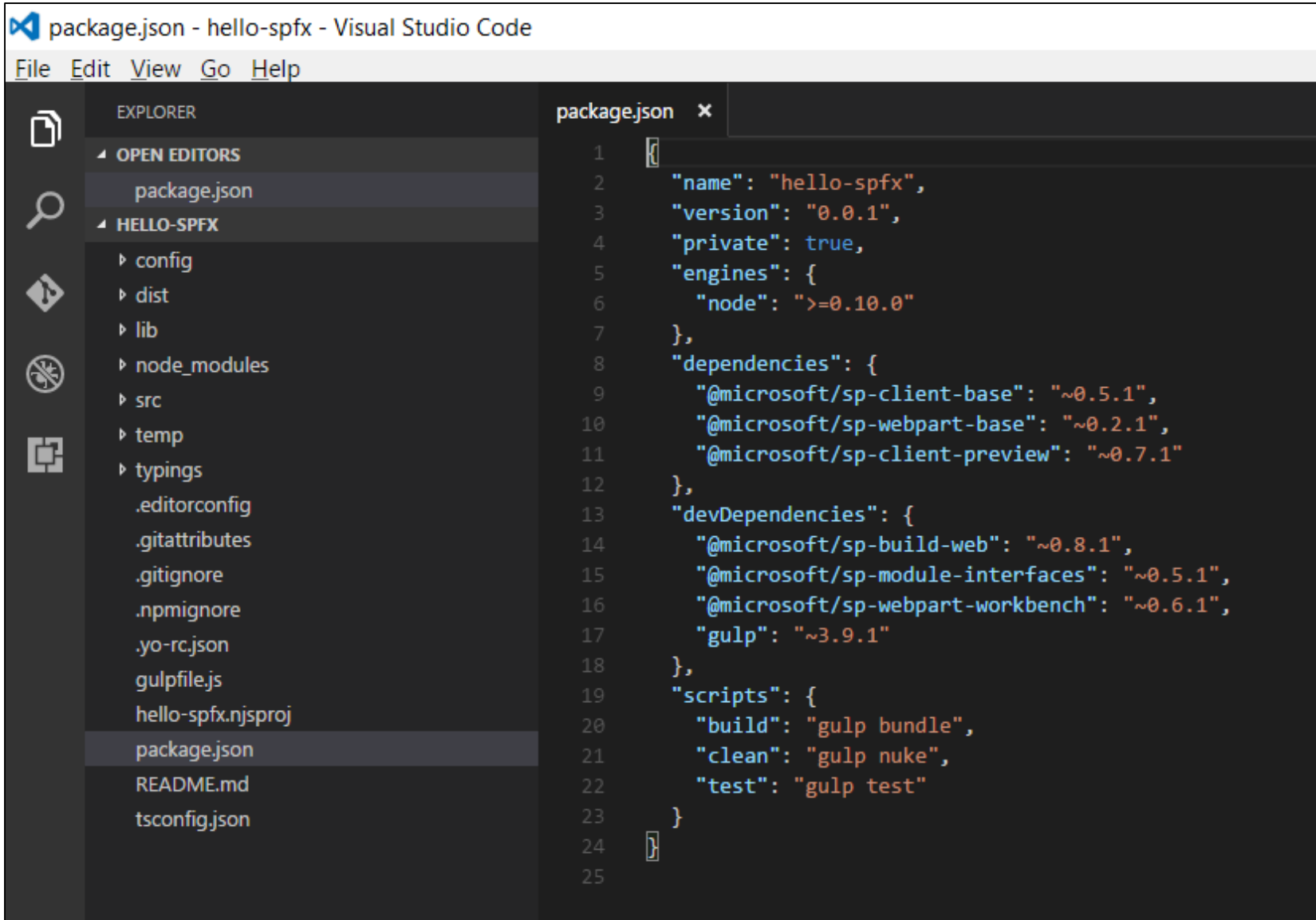
```
C:\Demos>md hello-spfx
C:\Demos>cd hello-spfx
C:\Demos\hello-spfx>yo @microsoft/sharepoint
```

The output of the `yo @microsoft/sharepoint` command includes a welcome message and a series of prompts to create a new SharePoint solution:

```
Let's create a new SharePoint solution.
? What is your solution name? hello-spfx
? Where do you want to place the files? Use ~ to place in current directory
? What is your webpart name? my-spfx-webpart
? What is your webpart description? absolute
? What framework would you like to start with? create
create package.json
create config\config.json
create config\deploy-azure-storage.json
create config\package-solution.json
create config\prepare-deploy.json
create config\serve.json
create config\tslint.json
create config\write-manifests.json
```

Overlaid on the main terminal are two smaller terminal windows. The top one shows the output of the `yo` command, including a welcome message and a link to the Yeoman SharePoint generator: <https://github.com/yeoman/generators-sharepoint>. The bottom one shows the output of the `gulp serve` command, which displays a congratulations message: "Congratulations! Solution hello-spfx is created. Run gulp serve to play with it!"

Package.json



The screenshot displays the Visual Studio Code interface with the 'package.json' file open in the editor. The Explorer sidebar on the left shows the project structure, including folders like 'config', 'dist', 'lib', 'node_modules', 'src', 'temp', and 'typings', as well as files like '.editorconfig', '.gitattributes', '.gitignore', '.npmignore', '.yo-rc.json', 'gulpfile.js', 'hello-spfx.njsproj', 'package.json', 'README.md', and 'tsconfig.json'. The 'package.json' file is selected and its content is shown in the editor window.

```
1 {  
2   "name": "hello-spfx",  
3   "version": "0.0.1",  
4   "private": true,  
5   "engines": {  
6     "node": ">=0.10.0"  
7   },  
8   "dependencies": {  
9     "@microsoft/sp-client-base": "~0.5.1",  
10    "@microsoft/sp-webpart-base": "~0.2.1",  
11    "@microsoft/sp-client-preview": "~0.7.1"  
12  },  
13  "devDependencies": {  
14    "@microsoft/sp-build-web": "~0.8.1",  
15    "@microsoft/sp-module-interfaces": "~0.5.1",  
16    "@microsoft/sp-webpart-workbench": "~0.6.1",  
17    "gulp": "~3.9.1"  
18  },  
19  "scripts": {  
20    "build": "gulp bundle",  
21    "clean": "gulp nuke",  
22    "test": "gulp test"  
23  }  
24 }  
25
```



Gulp as a Task Runner

- Gulp serves as a Task Runner
 - Compiles TypeScript files to JavaScript
 - Compiles SASS files to CSS
 - Bundles and minifies JavaScript and CSS files
- Create a self-signed certificate
gulp trust-dev-cert
- Start up the project for testing & debugging
gulp serve



Developing a SPFx Web Part?

- Create class that extends BaseClientSideWebPart
 - Override render() for minimal “hello world” functionality
 - Base class provides access to page context

HelloWebPart.ts ✕

```
1  import { BaseClientSideWebPart } from '@microsoft/sp-webpart-base';
2
3  import styles from './Hello.module.scss';
4
5  export interface IMyWebPartProps {}
6
7  export default class HelloWebPart extends BaseClientSideWebPart<IMyWebPartProps> {
8
9      public render(): void {
10         var styleName: string = styles.helloWebPart;
11         this.domElement.innerHTML = `<div class="${styleName}">Hello SPFx</div>`;
12     }
13
14 }
15
```



Working with SASS and .SCSS Files

- Sass: Syntactically Awesome Style Sheets
 - Compiles .scss files into .css files
 - Allows build process to use variables and nesting

```
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}
```

```
public render(): void {
  var styleName: string = styles.helloWebPart;
  this.domElement.innerHTML = `<div class="${styleName}">Hello SPFx</div>`;
}
```





DEMO

Hello World with SPFx

Adding a JavaScript Library (D3.js)

- Adding package for D3.js library

npm install d3 --save

- Add typings file to Intellisense and type checking

npm install @types/d3 --save-dev





DEMO

Using D3 with SPFx

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

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 panelPropertySettings()

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



Property Panel Settings

```
protected get propertyPaneSettings(): IPropertyPaneSettings {
    return {
        pages: [
            {
                header: { description: "Greeter Web Part" },
                groups: [
                    {
                        groupName: "General Properties",
                        groupFields: [
                            PropertyPaneTextField('greeting', { label: 'Greeting' }),
                        ]
                    },
                    {
                        groupName: "Cosmetic Properties",
                        groupFields: [
                            PropertyPaneToggle('largefont', {
                                label: 'Large Font',
                                onText: 'On',
                                offText: 'Off'
                            }),
                            PropertyPaneDropdown('color', {
                                label: 'Font Color',
                                options: [
                                    { key: 'green', text: 'Green' },
                                    { key: 'blue', text: 'Blue' },
                                    { key: 'red', text: 'Red' },
                                    { key: 'purple', text: 'Purple' }
                                ]
                            })
                        ]
                    }
                ]
            }
        ]
    }
}
```

Walmart Greeter

Greeter Web Part

General Properties

Greeting

Welcome to Walmart

Cosmetic Properties

Large Font

☒ On ☐ Off

Font Color

Blue



The background of the slide is an abstract digital pattern. It features a dense grid of small, glowing blue and white dots and lines, creating a sense of depth and movement. The pattern is reminiscent of a circuit board or a data visualization. The overall color scheme is dominated by blue and white, with a dark background.

DEMO

Web Part Properties

Calling the SharePoint REST API

```
private getListData(): Promise<ISPLists> {  
  
    var restUrl: string = this.context.pageContext.web.absoluteUrl +  
        `/_api/web/lists?$select=ID,Title,DefaultViewUrl&$filter=Hidden eq false`;  
  
    return this.context.httpClient.get(restUrl)  
        .then((response: Response) => {  
            return response.json();  
        });  
}
```

```
private renderList(items: ISPList[]): void {  
    var baseUrl = this.context.pageContext.web.absoluteUrl + "/";  
    let html: string = ``;  
    items.forEach((item: ISPList) => {  
        var DefaultViewUrl = baseUrl + item.DefaultViewUrl;  
        html += `  
<ul class="${styles.list}">  
    <li class="${styles.listItem}">  
        <a href='${DefaultViewUrl}'>${item.Title}</a>  
        <span class="ms-font-1"></span>  
    </li>  
</ul>`;   
    });  
  
    const listContainer: Element = this.domElement.querySelector('#spListContainer');  
    listContainer.innerHTML = html;  
}
```



The background of the slide is a close-up, low-angle shot of a server rack. The rack is filled with numerous server units, each featuring a grid of small, glowing blue lights. The perspective is looking up the length of the rack, creating a sense of depth and scale. The lighting is predominantly blue, giving it a high-tech, digital feel.

DEMO

Calling the SharePoint REST API

React and JSX

```
export default class Futurepart extends React.Component<any, any> {  
  
  constructor(props: any){  
    super(props);  
    this.state = { message: "Press the button when you can" };  
  }  
  
  public render(): JSX.Element {  
    return (  
      <div className={styles.futurepart}>  
        <div className={styles.container}>  
  
          <h3>Hello React and JSX/TSX</h3>  
  
          <div>  
            <input type="Button" onClick={e => this.onClickHandler(e) } value="Click me"  
          </div>  
  
          <div className={styles.message} >{this.state.message}</div>  
  
        </div>  
      </div>  
    );  
  }  
};
```



DEMO

Creating Web Parts with React.js

Agenda

- ✓ Overview the SharePoint Framework (SPFx)
- ✓ Setting up an SPFx Development Environment
- ✓ Creating Projects using the SPFx Templates
- Deploying SPFx Projects using an Azure CDN



Building a Deployment Package

```
Node.js command prompt

c:\Demos\charts>gulp package-solution_
```

Local Disk (C:) > Demos > charts > sharepoint > solution

Name	Date modified	Type
debug	12/8/2016 12:24 PM	File folder
charts.spapp	12/8/2016 12:24 PM	SPAPP File

Home

Apps for SharePoint ①

[New](#) [Upload](#) [Sync](#) [Share](#) [More](#) ▾

[All Apps](#) [Featured Apps](#) [Unavailable Apps](#) ...

✓	📄	Title	Name	App Version	Edit	Product ID	Metadata Language	Default Metadata Language
Product ID : {95073FC4-5FB6-40DA-8DF9-C064604A2228} (1)								
	📄	charts-client-side-solution	charts 🌿	...	1.0.0.0	📄 {95073FC4-5FB6-40DA-8DF9-C064604A2228}	English - 1033	Yes



Deploying to Azure

- Gulp commands to deploy to CDN
 1. **gulp --ship**
 2. **gulp deploy-azure-storage**
 3. **gulp bundle --ship**
 4. **gulp package-solution --ship**



Summary

- ✓ Overview the SharePoint Framework (SPFx)
- ✓ Setting up an SPFx Development Environment
- ✓ Creating Projects using the SPFx Templates
- ✓ Deploying SPFx Projects using an Azure CDN



Assets included in deployment packages

