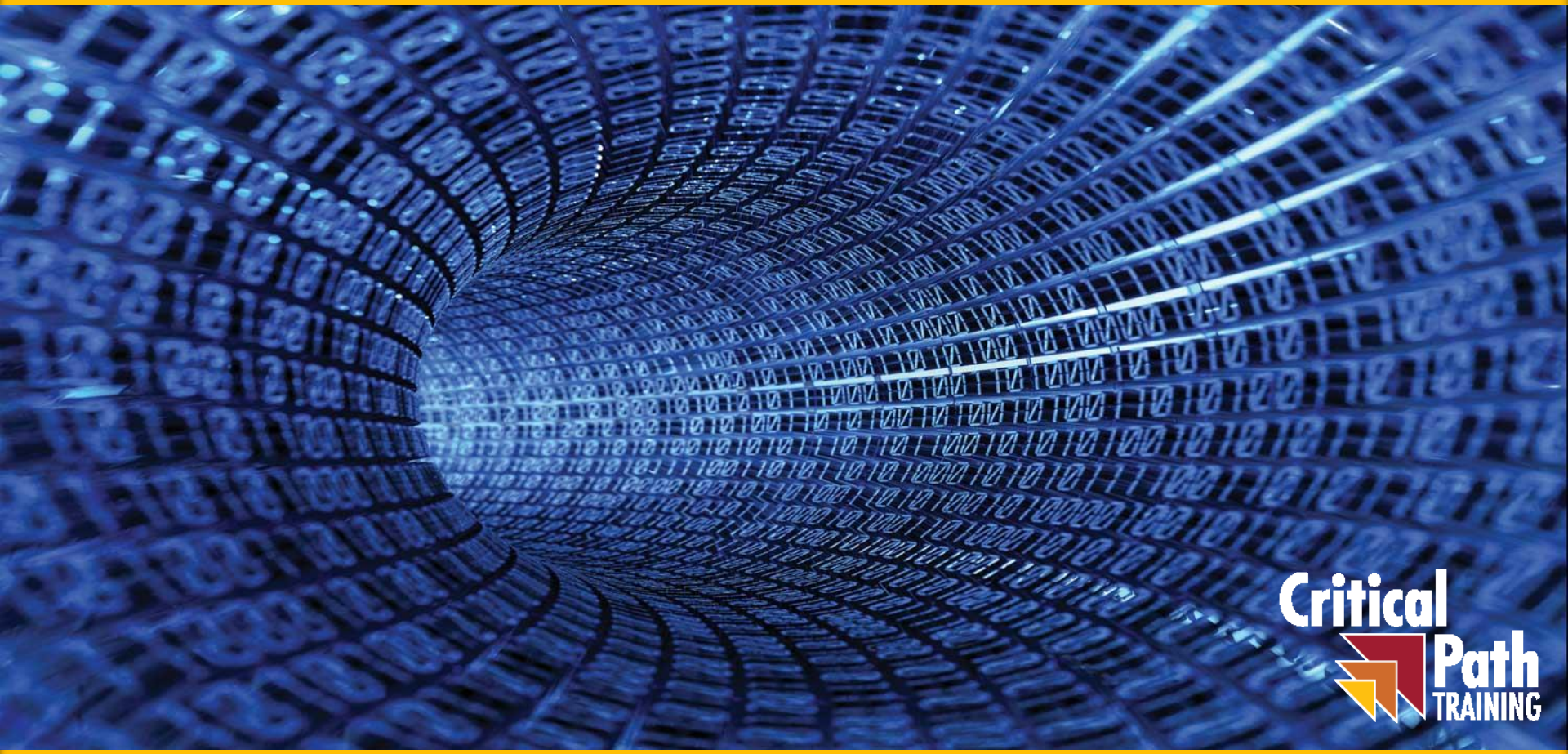
Developing with the SharePoint Framework



Agenda

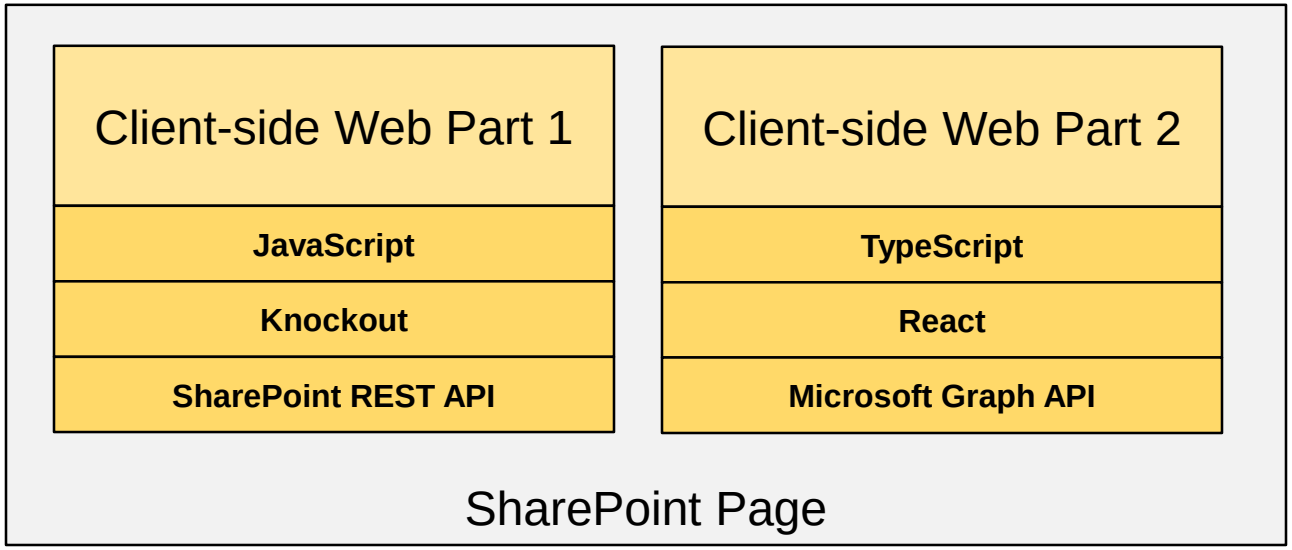
- SharePoint Framework Overview
- Creating new SPFx Projects
- Running SPFx Projects in SharePoint Workbench
- The SharePoint Framework Object Model
- Extending Web Parts with Custom Properties
- Programming with the SharePoint REST API

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



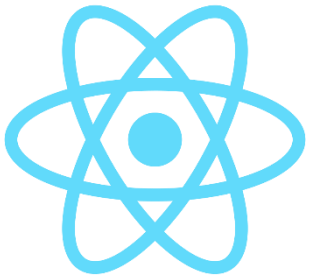
How Does SPFx Work?

- 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
- React

YEOMAN



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

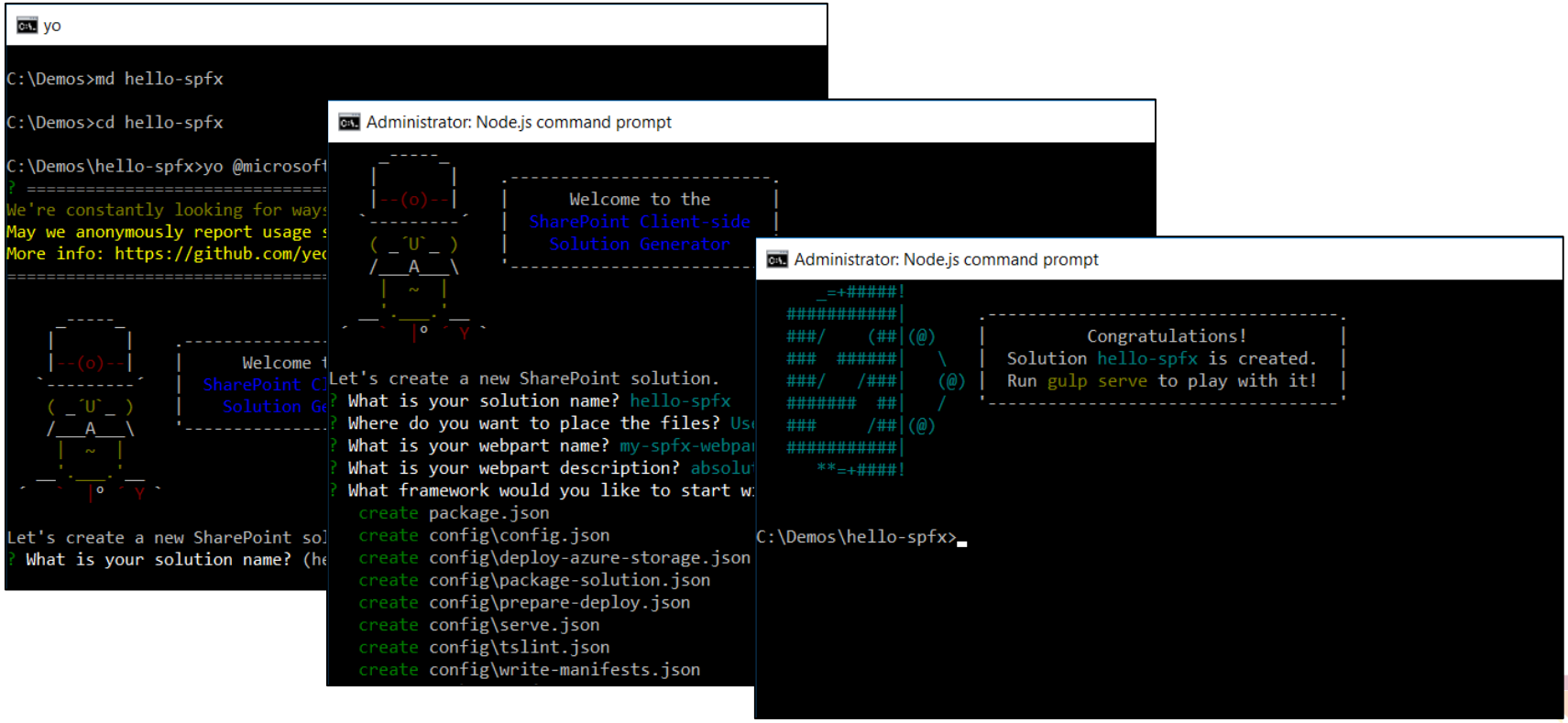
  npm install -g @microsoft/generator-sharepoint

Agenda

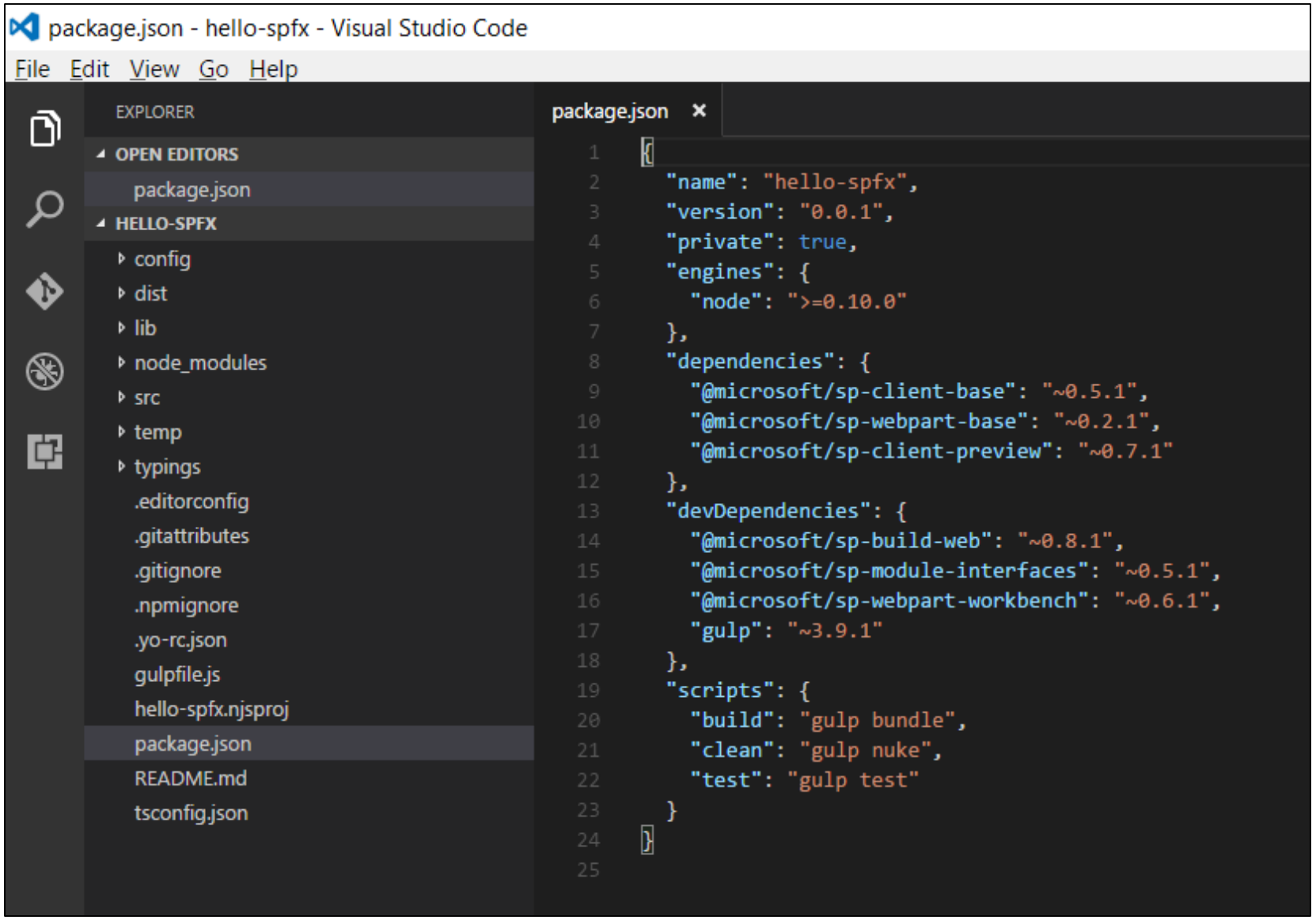
- ✓ Overview the SharePoint Framework (SPFx)
- ✓ Setting up an SPFx Development Environment
- ➢ Creating Projects using the SPFx Templates
- Debugging with the SharePoint Workbench
- Developing SPFx Web Parts using React.js
- Deploying SPFx Projects using an Azure CDN

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



Package.json



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

```scss
$font-stack:    Helvetica, sans-serif;
$background-color: lightyellow;
$foont-size: 3.0em;
$padding: 18px;

.helloWebPart{
  font: $font-stack;
  font-size: $foont-size;
  background-color: $background-color;
  border: 1px solid black;
  border-radius: $padding;
  padding: $padding;
  text-align: center;
}
```

SASS

Hello.module.css

```css
.helloWebPart_0989818e{
    font:Helvetica,sans-serif;
    font-size:3em;
    background-color:#ffffe0;
    border:1px solid #000;
    border-radius:18px;
    padding:18px;
    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 Intellisence 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.emal:", this.context.pageContext.user.email);
  this.AddTableRow(table, "this.diplayMode:", this.displayMode.toString());
  this.AddTableRow(table, "context.webPartTag:", this.context.webPartTag);
  container.append(table);
```

Property	Value
site.id:	a5aa0f03-16b6-4057-8704-daaea2f84494
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.emal:	
this.diplayMode:	2
context.webPartTag:	WebPart.InspectorWebPart.eaf44355-2d45-4e1c-b8de-e8b3bce60279

Web Part Properties

- Define interface with properties

IGreeterWebpartWebPartProps.ts

```ts
export interface IGreeterWebpartWebPartProps {
  greeting: string;
  largefont: boolean;
  color: string;
}
```

- Add interface to web part class definition

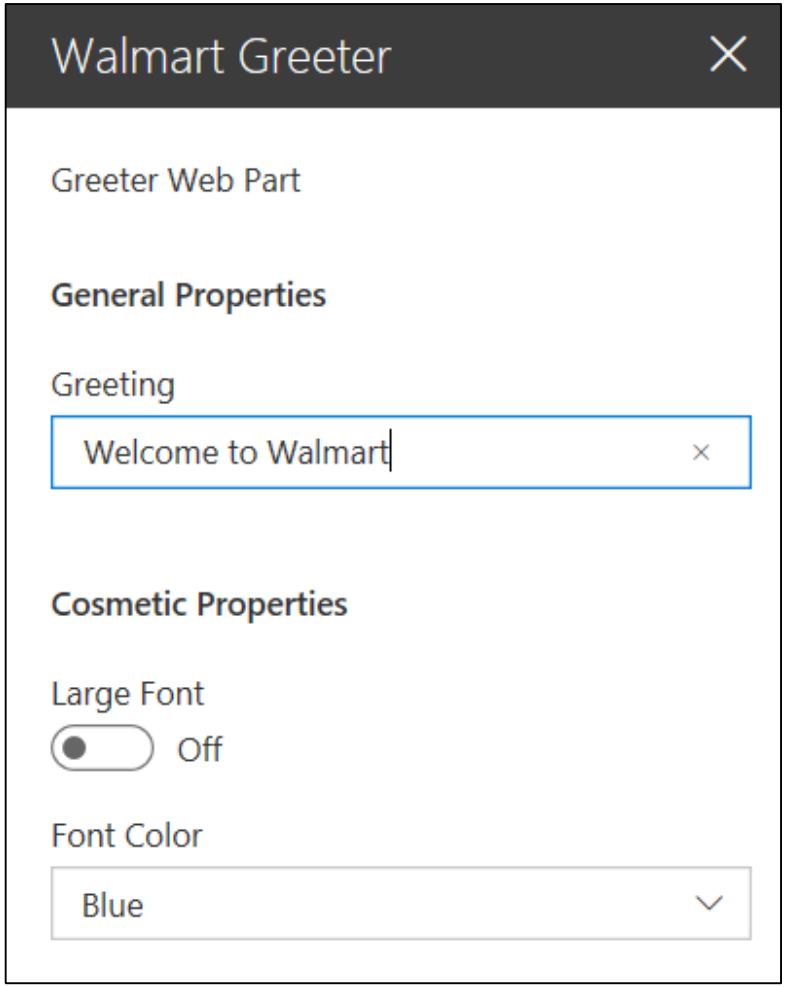
```ts
class GreeterWebpartWebPart extends BaseClientSideWebPart<IGreeterWebpartWebPartProps> {
```

- Override panelPropertySettings()

```ts
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' }
                ]
              })
            ]
```

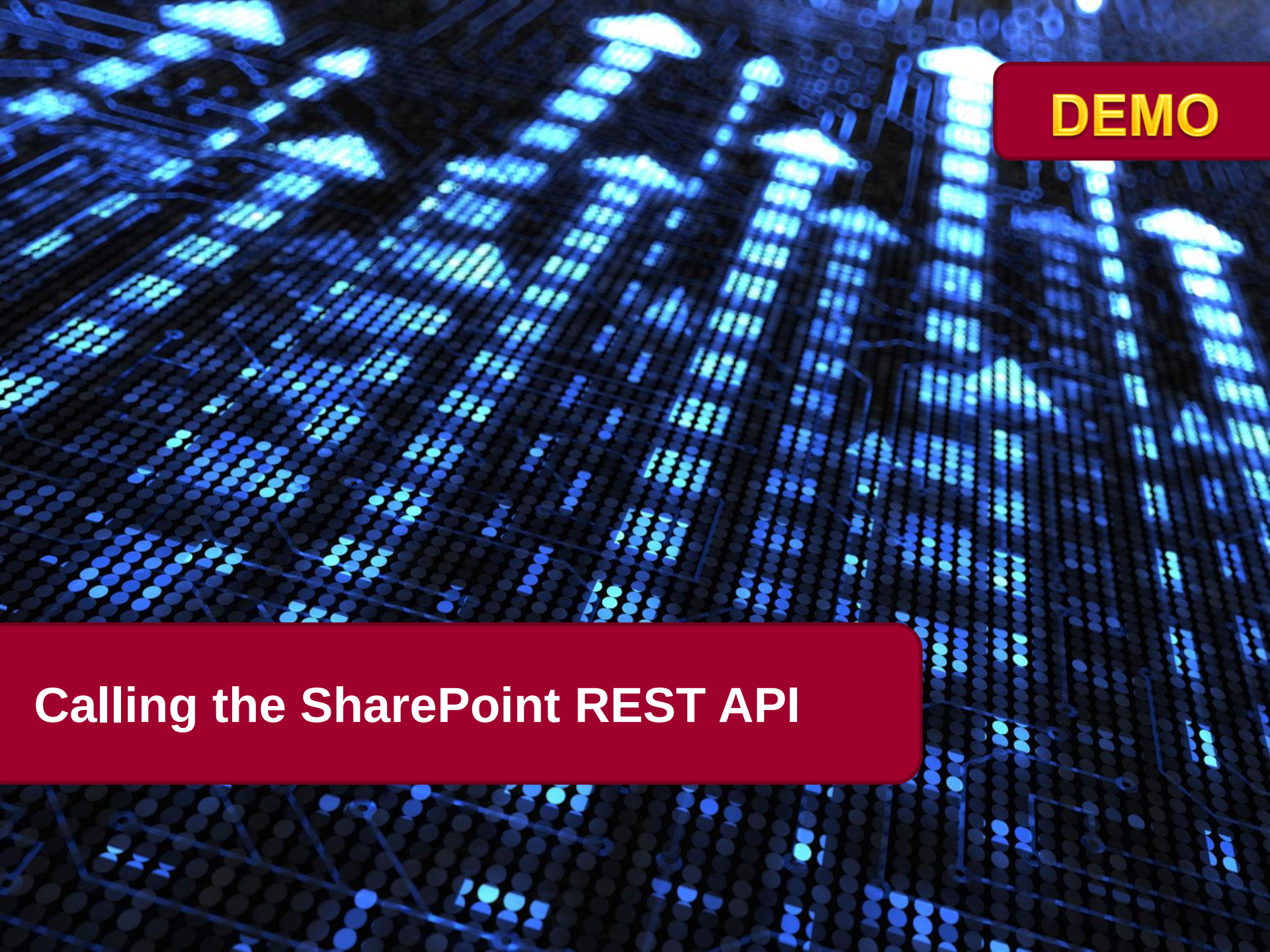



DEMO
Web Part Properties

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-l"></span>
          </li>
      </ul>`;
      });

      const listContainer: Element = this.domElement.querySelector('#spListContainer');
      listContainer.innerHTML = html;
  }
```


DEMO
Calling the SharePoint REST API

React and JSX

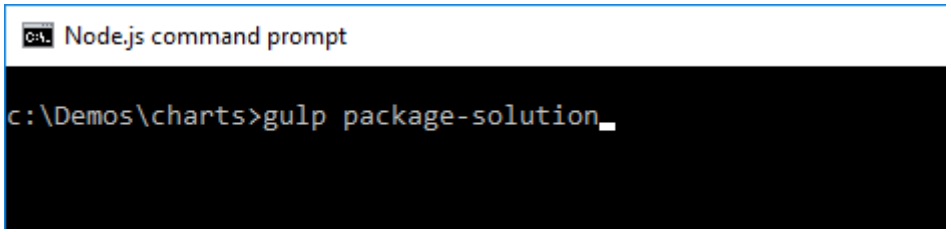
```
export default class Futurepart extends React.Component<any, any> {

  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>
    );
```

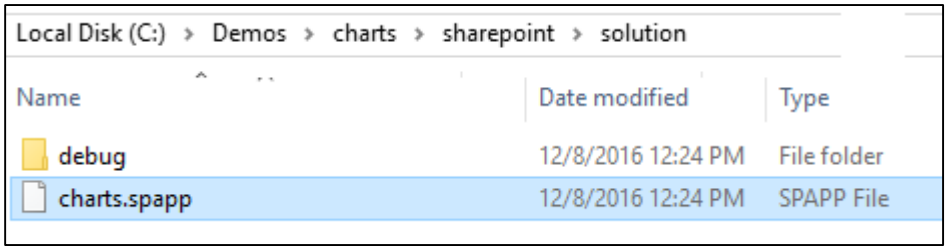

DEMO
Creating Web Parts with React.js

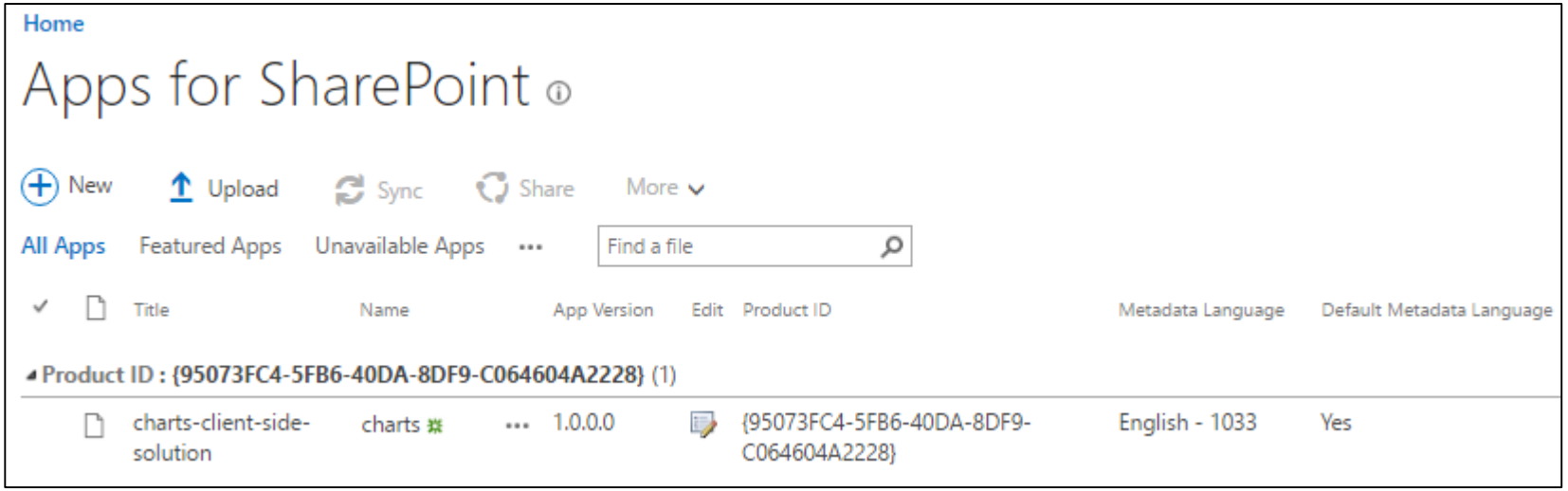
Agenda

- ✓ 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







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