

Embedding Reports and Dashboards using SharePoint Framework



Agenda

- Creating SharePoint Framework Projects
- Quick Primer on SharePoint Framework Development
- Granting Tenant-level Permissions to Call to Power BI Service API
- Calling the Power BI Service API using AadHttpClient
- Installing the Power BI JavaScript API Package in an SPFx Project
- Using Typescript to Embed Reports & Dashboards in Web Parts



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



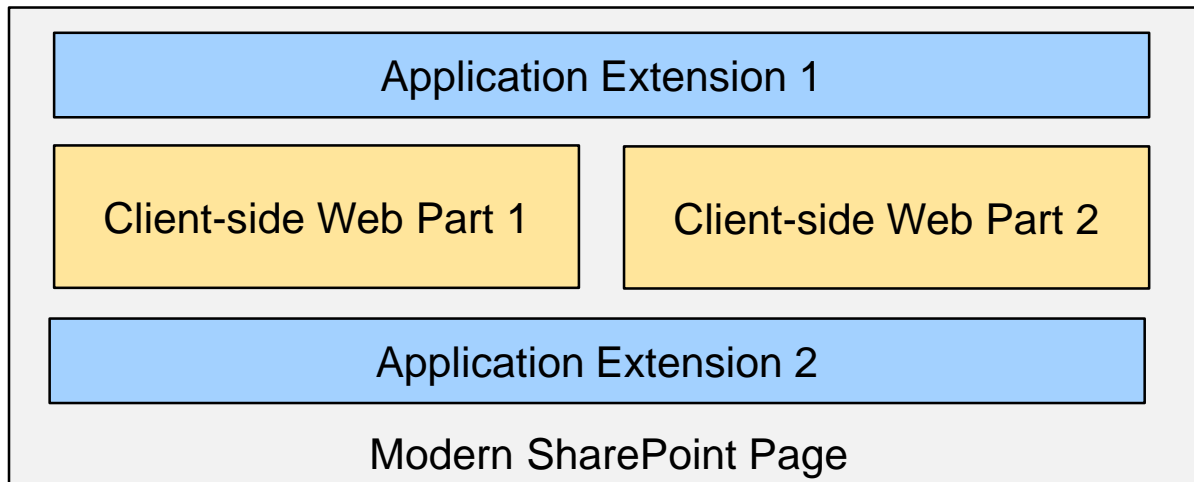
SPFx versus the SharePoint Add-in Model

- 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 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

```
npm install -g gulp
```

- Install Yeoman

```
npm install -g yo
```

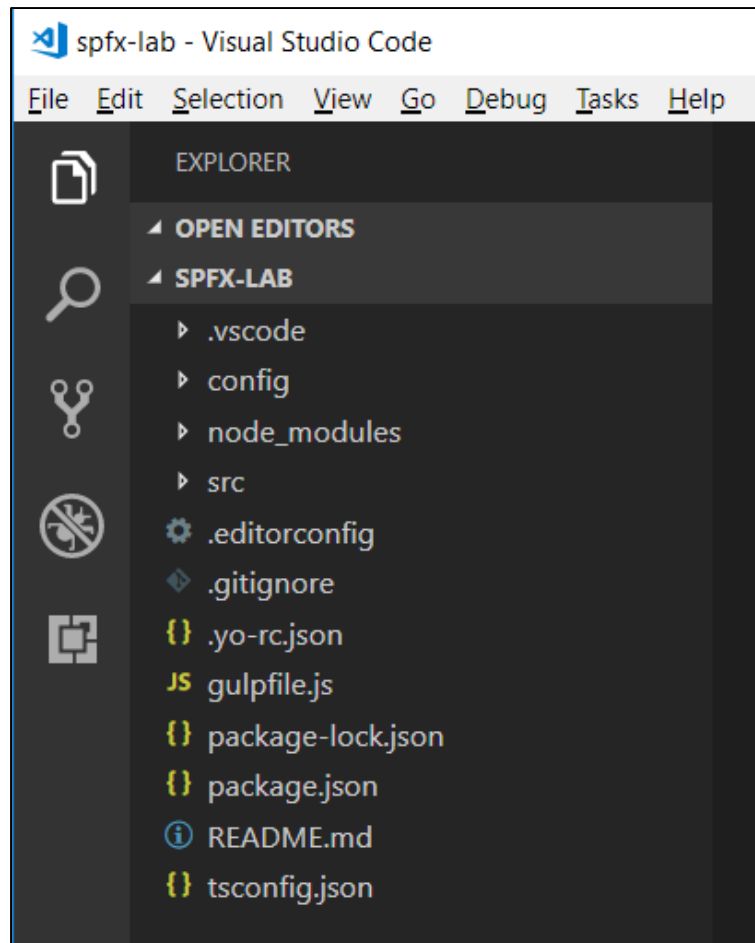
- Install Yeoman Template for SPFx

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



SharePoint Framework Project Structure

- Project created as Node.js project



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>`;
  }
}
```



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

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: lightyellow;
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 #000;
6    border-radius:18px;
7    padding:18px;
8    text-align:center}
```

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

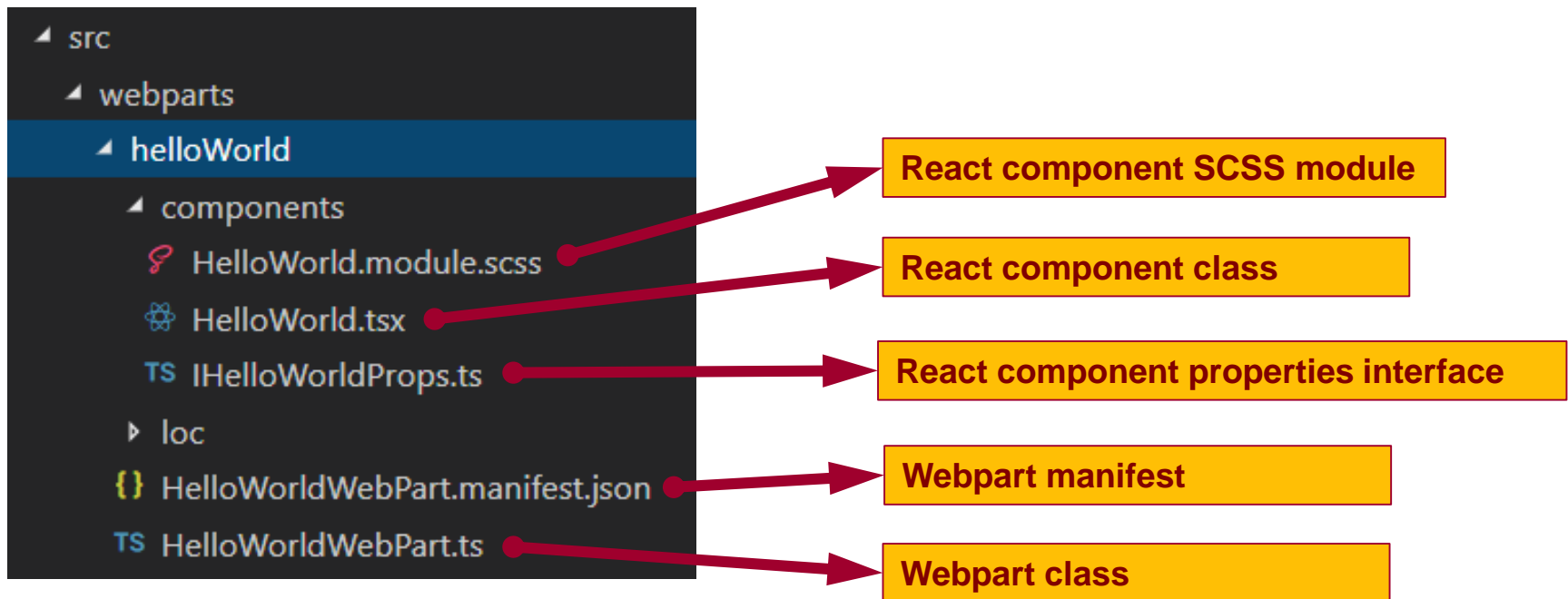
Font Color

Blue



Creating a React Webpart

- You can select React as framework for your webpart
 - You can create a React webpart when creating new project
 - You can add React webpart to existing project
 - React webpart made up of several different source files



React Webpart Architecture

```
export default class HelloWorldWebPart extends BaseClientSideWebPart<IHelloWorldWebPartProps> {  
  
    public render(): void {  
        const element: React.ReactElement<IHelloWorldProps> = React.createElement(  
            HelloWorld, { description: this.properties.description }  
        );  
        ReactDOM.render(element, this.domElement);  
    }  
}
```

```
export interface IHelloWorldProps {  
    description: string;  
}
```

```
import * as React from 'react';  
  
import { IHelloWorldProps } from './IHelloWorldProps';  
  
export default class HelloWorld extends React.Component<IHelloWorldProps, {}> {  
  
    public render(): React.ReactElement<IHelloWorldProps> {  
        return <div>{this.props.description}</div>;  
    }  
}
```

Webpart class
instance

React.CreateElement

description

React component
instance



React Webpart Styling

```
HelloWorld.module.scss •  
  
.helloWorld {  
  background-color: lightsalmon;  
  border: 4px solid purple;  
  border-radius: 12px;  
  
  .title {  
    padding: 8px;  
    font-size: 48px;  
  }  
}
```



```
HelloWorld.tsx ×  
  
import * as React from 'react';  
  
import { IHelloWorldProps } from './IHelloWorldProps';  
  
import styles from './HelloWorld.module.scss';  
  
export default class HelloWorld extends React.Component<IHelloWorldProps, {}> {  
  
  public render(): React.ReactElement<IHelloWorldProps> {  
    return (  
      <div className={styles.helloWorld}>  
        <div className={styles.title}>  
          {this.props.description}  
        </div>  
      </div>  
    );  
  }  
}
```



Passing SPHttpClient to the React Component

```
import { SPHttpClient } from '@microsoft/sp-http';

export interface ILeadTrackerProps {
  targetListDefault: string;
  siteUrl: string;
  spHttpClient: SPHttpClient | undefined;
}
```

```
public render(): void {

  const element: React.ReactElement<ILeadTrackerProps> = React.createElement(
    LeadTracker, {
      targetListDefault: this.properties.targetList,
      siteUrl: this.context.pageContext.web.absoluteUrl,
      spHttpClient: <SPHttpClient>this.context.spHttpClient
    }
  );

  this.leadTracker = <LeadTracker>ReactDOM.render(element, this.domElement);
}
```



Create a New SPFX Web Part Project

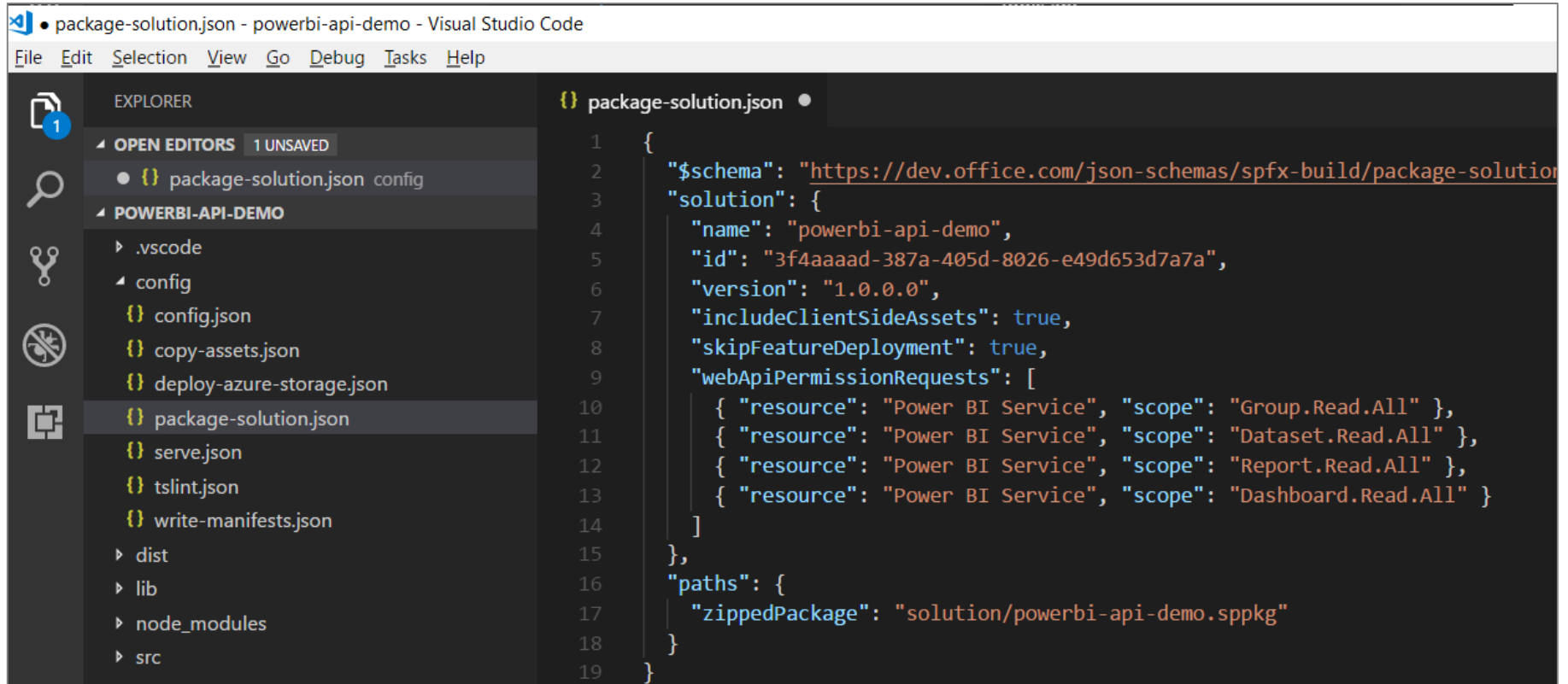
```
c:\Vegas\PBIESPF\powerbi-api-demo>yo @microsoft/sharepoint

  _--_
  |  (o)  |
  |__|_   |
  |  ^__^  |
  |  (oo)\_______
  |  (__)\       )\/\
  |_____|       (oo)/_____
  |  (__)\       )\/\
  |_____|       (oo)/_____

Welcome to the
SharePoint Client-side
Solution Generator

Let's create a new SharePoint solution.
? What is your solution name? powerbi-api-demo
? 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? Yes
? Which type of client-side component to create? WebPart
? What is your Web part name? reportlist
? What is your Web part description? A sample web part which calls the Power BI Service API
? Which framework would you like to use? (Use arrow keys)
> No JavaScript framework
  React
  Knockout
```

Configuring Web API Permissions

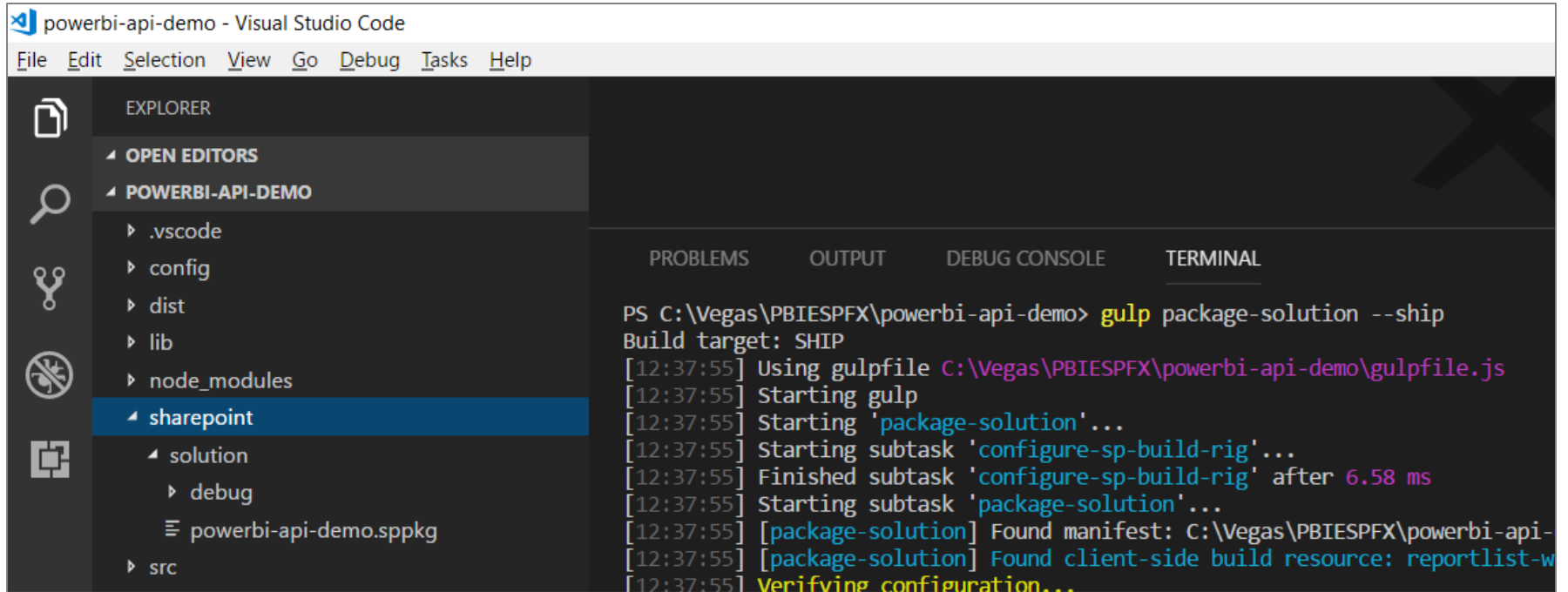


The screenshot shows the Visual Studio Code interface with the file `package-solution.json` open in the editor. The Explorer sidebar on the left shows the project structure, with `package-solution.json` selected under the `POWERBI-API-DEMO` folder. The editor displays the following JSON configuration:

```
1 {
2   "$schema": "https://dev.office.com/json-schemas/spfx-build/package-solution",
3   "solution": {
4     "name": "powerbi-api-demo",
5     "id": "3f4aaaad-387a-405d-8026-e49d653d7a7a",
6     "version": "1.0.0.0",
7     "includeClientSideAssets": true,
8     "skipFeatureDeployment": true,
9     "webApiPermissionRequests": [
10      { "resource": "Power BI Service", "scope": "Group.Read.All" },
11      { "resource": "Power BI Service", "scope": "Dataset.Read.All" },
12      { "resource": "Power BI Service", "scope": "Report.Read.All" },
13      { "resource": "Power BI Service", "scope": "Dashboard.Read.All" }
14    ]
15  },
16  "paths": {
17    "zippedPackage": "solution/powerbi-api-demo.sppkg"
18  }
19 }
```



Packaging Your SPFX Solution



```
powerbi-api-demo - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

EXPLORER
├─ OPEN EDITORS
├─ POWERBI-API-DEMO
│  ├── .vscode
│  ├── config
│  ├── dist
│  ├── lib
│  ├── node_modules
│  └─ sharepoint
│     ├── solution
│     │  ├── debug
│     │  └─ powerbi-api-demo.sppkg
│     └─ src
└─ ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Vegas\PBIESPFX\powerbi-api-demo> gulp package-solution --ship
Build target: SHIP
[12:37:55] Using gulpfile C:\Vegas\PBIESPFX\powerbi-api-demo\gulpfile.js
[12:37:55] Starting gulp
[12:37:55] Starting 'package-solution'...
[12:37:55] Starting subtask 'configure-sp-build-rig'...
[12:37:55] Finished subtask 'configure-sp-build-rig' after 6.58 ms
[12:37:55] Starting subtask 'package-solution'...
[12:37:55] [package-solution] Found manifest: C:\Vegas\PBIESPFX\powerbi-api-
[12:37:55] [package-solution] Found client-side build resource: reportlist-w
[12:37:55] Verifying configuration...
```



Deploy the Web Part to the App Gallery

The screenshot displays the SharePoint 'Apps for SharePoint' interface. At the top, there's a 'Home' link and the title 'Apps for SharePoint'. Below this, a navigation bar includes 'New', 'Upload', 'Sync', 'Share', and a 'More' dropdown. A search bar labeled 'Find a file' is also present. The main content area shows a table with columns for app management. Overlaid on this is a file explorer window titled 'solution' showing the contents of a 'sharepoint' folder, which includes a 'debug' folder and a 'powerbi-api-demo.sppkg' file.

Home

Apps for SharePoint ⓘ

New **Upload** **Sync** **Share** **More** ▼

All Apps **Featured Apps** **Unavailable Apps** ...

✓	📄	Title	Name	App Version	Edit	Product ID	Metadata Language	Default Metadata Language	Modified	Enabled	Valid App Package	Deployed
---	---	-------	------	-------------	------	------------	-------------------	---------------------------	----------	---------	-------------------	----------

File Explorer: solution

File Home Share View

« sharepoint » solution Search sol...

Quick access

- Desktop
- Downloads

Name

- debug
- powerbi-api-demo.sppkg



Configuring Trust

Do you trust powerbi-api-demo?

The client-side solution you are about to deploy contains full trust client side code. The components in the solution can, and usually do, run in full trust, and no resource usage restrictions are placed on them.

This client side solution will get content from the following domains:


SharePoint Online

☒ Make this solution available to all sites in the organization

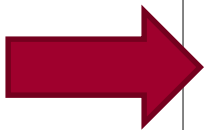
Please go to the Service Principal Permissions Management Page to approve pending permissions.

Deploy

Cancel



powerbi-api-demo



Granting Web API Permissions

API management

Control the third-party APIs that can be called by apps and custom script.

^ API name	Permission	Access
^ Pending approval (0)		
^ Approved (4)		
Power BI Service	Group.Read.All	Approved
Power BI Service	Group.Read.All Dataset.Read.All	Approved
Power BI Service	Group.Read.All Dataset.Read.All Report.	Approved
Power BI Service	Group.Read.All Dataset.Read.All Report.	Approved

[admin center](#) [feedback](#)



Calling Power BI API with AadHttpClient

```
import { AadHttpClient, HttpClientResponse } from '@microsoft/sp-http';

export default class ReportlistWebPart extends BaseClientSideWebPart<any> {

    private powerbiApiResourceId = "https://analysis.windows.net/powerbi/api";
    private pbiClient: AadHttpClient;

    protected onInit(): Promise<void> {
        this.pbiClient = new AadHttpClient(this.context.serviceScope, this.powerbiApiResourceId);
        return Promise.resolve();
    }

    public render(): void {
        var urlReports: string = "https://api.powerbi.com/v1.0/myorg/reports/";
        this.pbiClient.get(urlReports, AadHttpClient.configurations.v1)
            .then((res: HttpClientResponse): Promise<any> => {
                return res.json();
            })
            .then((reports: any): void => {
                this.domElement.innerHTML =
                    `<h2>Power BI Reports</h2>
                    <ul>
                        ${ reports.value.map(r => `<li><a href='${r.webUrl}' target='_blank' >${r.name}</a></li>`).join("") }
                    </ul>`;
            });
    }
}
```

Power BI Reports

- [Northwind Retro](#)
- [Wingtip Sales Analysis](#)

Installing the Power BI JavaScript API

- `npm install powerbi-client --save-dev`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Vegas\PBIESPFX\embed-report-demo> npm install powerbi-client --save
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4
)

+ powerbi-client@2.5.1
added 9 packages in 22.762s
```

```
› postcss-value-parser
› postcss-zindex
› powerbi-client
› powerbi-models
› powerbi-router
› prelude-ls
```

```
tsconfig.json x
1  {
2    "compilerOptions": {
3      "target": "es5",
4      "forceConsistentCasingInFileNames": true,
5      "module": "commonjs",
6      "jsx": "react",
7      "declaration": true,
8      "sourceMap": true,
9      "experimentalDecorators": true,
10     "skipLibCheck": true,
11     "typeRoots": [
12       "./node_modules/@types",
13       "./node_modules/@microsoft",
14       "./node_modules/powerbi-client",
15       "./node_modules/powerbi-models"
16     ],
17     "types": [
18       "es6-promise",
19       "webpack-env"
20     ],
21     "lib": [
22       "es5",
23       "dom",
24       "es2015.collection"
25     ]
26   }
27 }
28
```



```

public render(): void {
    let hostDiv: JQuery = $(this.domElement);
    let height: string = this.properties.reportHeight + "px";
    hostDiv.empty().css({ "margin": "0", "padding": "0", "height": height });

    var reqHeaders: HeadersInit = new Headers();
    reqHeaders.append("Accept", "*");
    this.pbiclient.get(this.reportUrl, AadHttpClient.configurations.v1, { headers: reqHeaders })
        .then((res: HttpClientResponse): Promise<any> => {
            return res.json();
        })
        .then((report: any): void => {
            console.log("begin embed...");
            var embedReportId: string = report.id;
            var embedUrl: string = report.embedUrl;
            var accessToken: string = window.sessionStorage["adal.access.token.keyhttps://analysis.windows.net/powerbi/api"];
            // Get models object to access enums for embed configuration
            var models = pbimodels;

            var config: any = {
                type: 'report',
                id: embedReportId,
                embedUrl: embedUrl,
                accessToken: accessToken,
                tokenType: models.TokenType.Aad,
                permissions: models.Permissions.All,
                viewMode: models.ViewMode.View,
                settings: {
                    filterPaneEnabled: false,
                    navContentPaneEnabled: this.properties.showPageTabs,
                }
            };
            window.powerbi.reset(this.domElement);
            window.powerbi.embed(this.domElement, config);
        });
}

```

Agenda

- Creating SharePoint Framework Projects
- Quick Primer on SharePoint Framework Development
- Granting Tenant-level Permissions to Call to Power BI Service API
- Calling the Power BI Service API using AadHttpClient
- Installing the Power BI JavaScript API Package in an SPFx Project
- Using Typescript to Embed Reports & Dashboards in Web Parts

