

SharePoint Framework Deep Dive – Building Web Parts

Rob Windsor
rob@robwindsor.com
@robwindsor

About Me



Rob Windsor

.NET/Microsoft 365 developer, trainer, author
Microsoft MVP | Microsoft 365 Development



Twitter: <https://twitter.com/robwindsor>

LinkedIn: <https://www.linkedin.com/in/rwindsor>

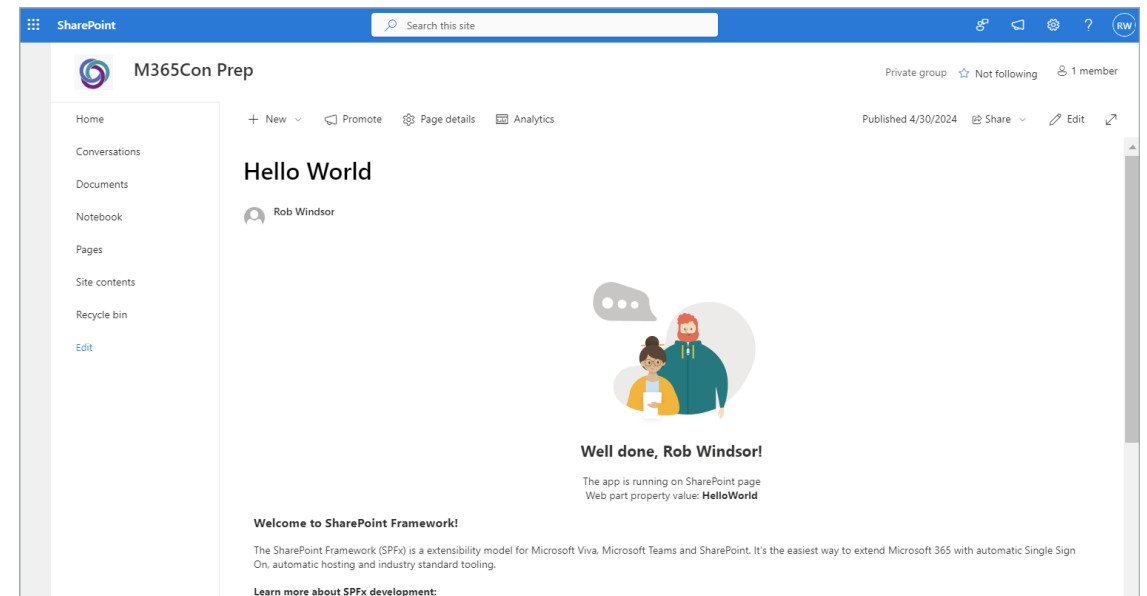
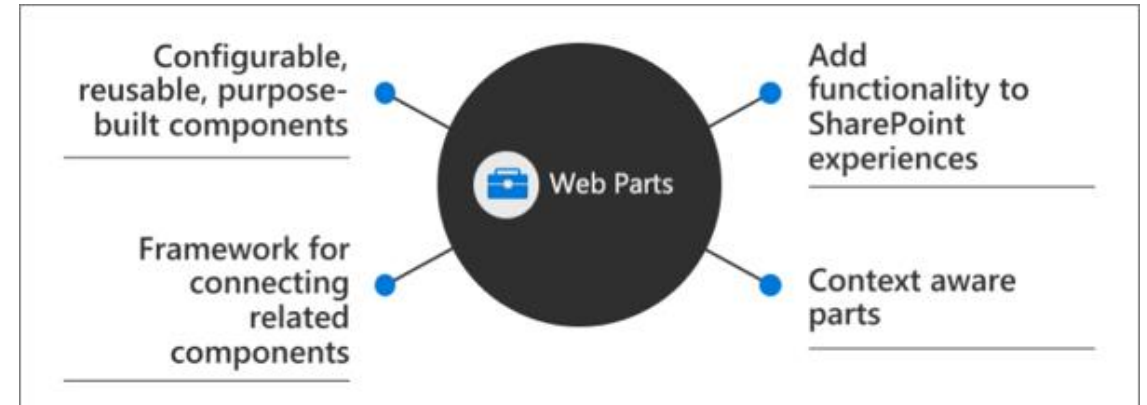
Blog: <https://robwindsor.hashnode.dev>

YouTube: <https://www.youtube.com/@RobWindsor>

GitHub: <https://github.com/rob-windsor>

Client-side Web Parts


- Reusable components that can be used in SharePoint pages
 - Can also be used in Teams tabs
 - Can also be used as personal apps in Teams, Outlook, and or the Microsoft 365 app
- Execute client-side in the browser
- Context aware: can get information about the current user and current host application (SharePoint, Teams, Outlook, or Microsoft 365 app)
- Configurable via the Web Part Property Pane



Creating a New Solution (Project)

- Also known as scaffolding a new solution
- Navigate to a folder where you want the solution to be created
- Run Yeoman SharePoint generator by executing **yo @microsoft/sharepoint**
- Generator will ask questions about the name of the solution, the type of solution you wish to create, and the web framework you wish to use
- The solution files will be scaffolded and **npm install** will be run after you've answered the questions

```
C:\Dev\Community\M365Con\Prep>yo @microsoft/sharepoint
```



```
Welcome to the Microsoft
365 SPFx Yeoman
Generator@1.18.2
```

See <https://aka.ms/spfx-yeoman-info> for more information on how to use this generator.

Let's create a new Microsoft 365 solution.

? What is your solution name? HelloWorld

? Which type of client-side component to create? WebPart

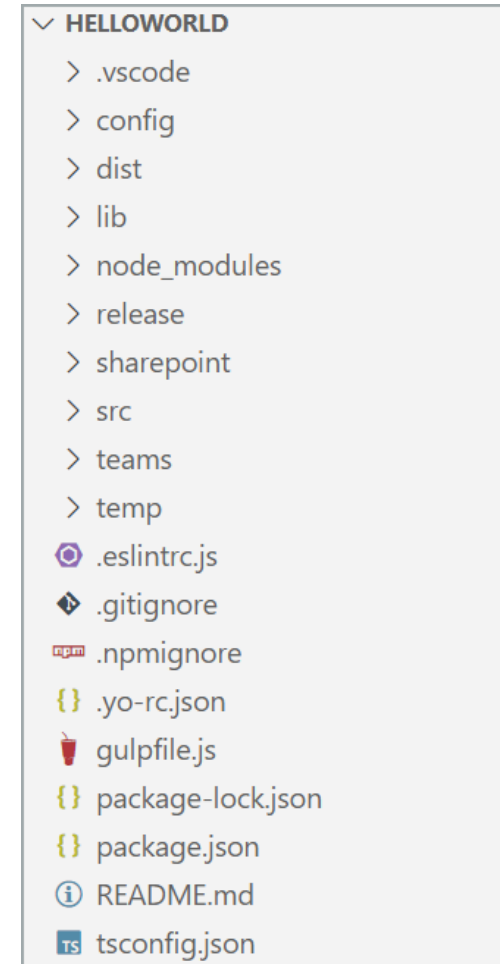
Add new Web part to solution hello-world.

? What is your Web part name? HelloWorld

? Which template would you like to use? React

Solution Structure

- **config:** configuration files used by the project's various tasks.
- **dist:** files generated when you bundle your project, regardless of which switch you use. Unminified JavaScript files and source maps contained in this folder are used when you run in debug mode.
- **lib:** temporary files generated from the compilation and transpilation of TypeScript to JavaScript and SCSS to CSS files.
- **release:** contains a subfolder named assets that contains the files generated when you use the ship switch when you bundle. These files are deployed to the CDN. Folder also contains two more subfolders that contain manifest files.
- **src:** the source code for your project.
- **temp:** files used by the local development web server.



Web Part Class

- Defines the main entry point for the web part
- Must extend **BaseClientSideWebPart**
- References interface that defines the non-standard public properties of the web part

```
export interface IHelloWorldWebPartProps {  
  · description: string;  
}  
  
export default class HelloWorldWebPart ·  
  · extends BaseClientSideWebPart<IHelloWorldWebPartProps> { ... }
```

Web Part Manifest

```
{
  "$schema": "https://developer.microsoft.com/json-schemas/spfx/client-side",
  "id": "6a76bf6f-a26e-401a-a56d-e4e2ba2f7331",
  "alias": "HelloWorldWebPart",
  "componentType": "WebPart",

  "version": "*",
  "manifestVersion": 2,

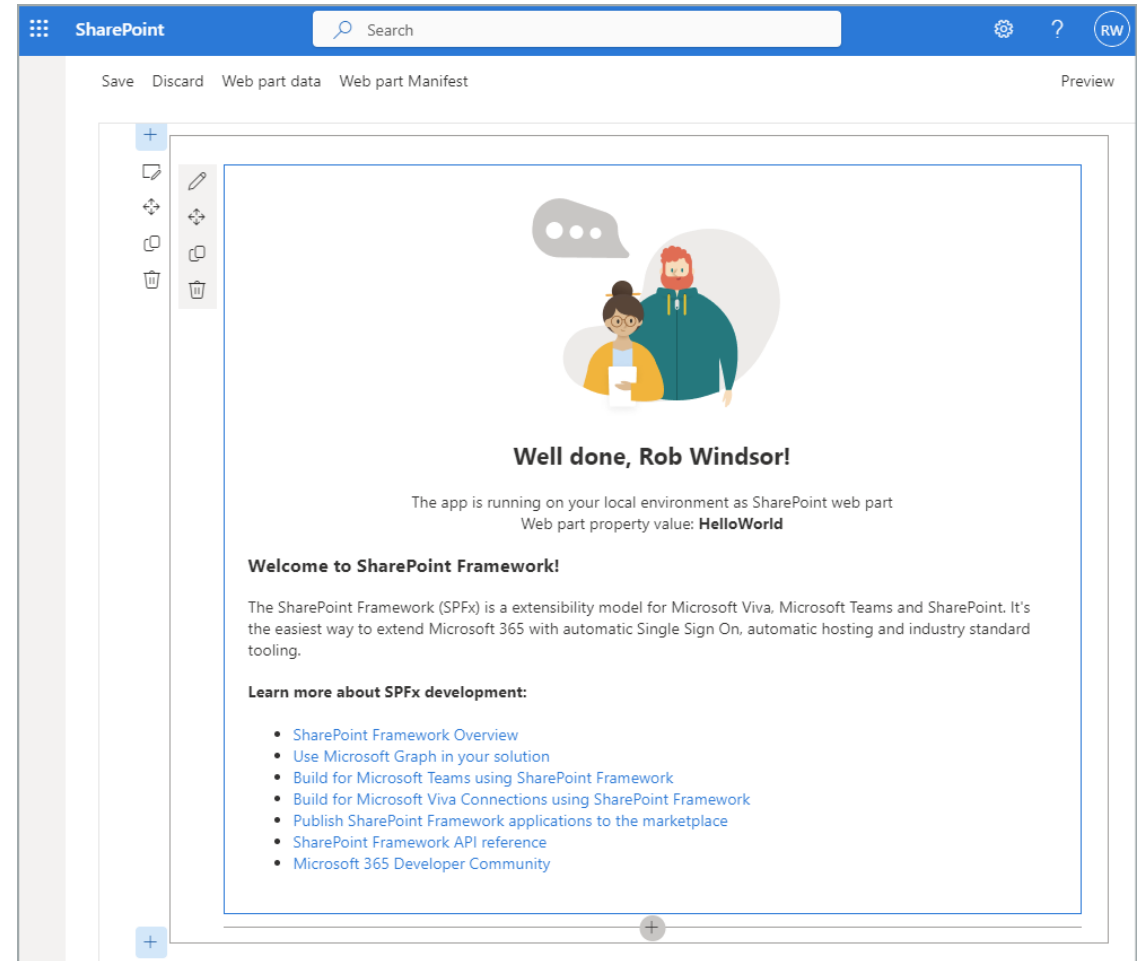
  "requiresCustomScript": false,
  "supportedHosts": ["SharePointWebPart", "TeamsPersonalApp", "TeamsTab",
    "SharePointFullPage"],
  "supportsThemeVariants": true,

  "preconfiguredEntries": [{
    "groupId": "5c03119e-3074-46fd-976b-c60198311f70", // other
    "group": { "default": "Other" },
    "title": { "default": "Hello SPFx" },
    "description": { "default": "My first SPFx web part" },
    "officeFabricIconFontName": "BirthdayCake",
    "properties": {
      "description": "HelloWorld"
    }
  }]
}
```

- Defines the web part metadata

SharePoint Framework Workbench

- Each SharePoint site contains a workbench page
 - **`https://{site-url}/_layouts/workbench.aspx`**
- The workbench is a special page that contains a single canvas that can be used to test web parts that have been deployed and/or web parts running locally

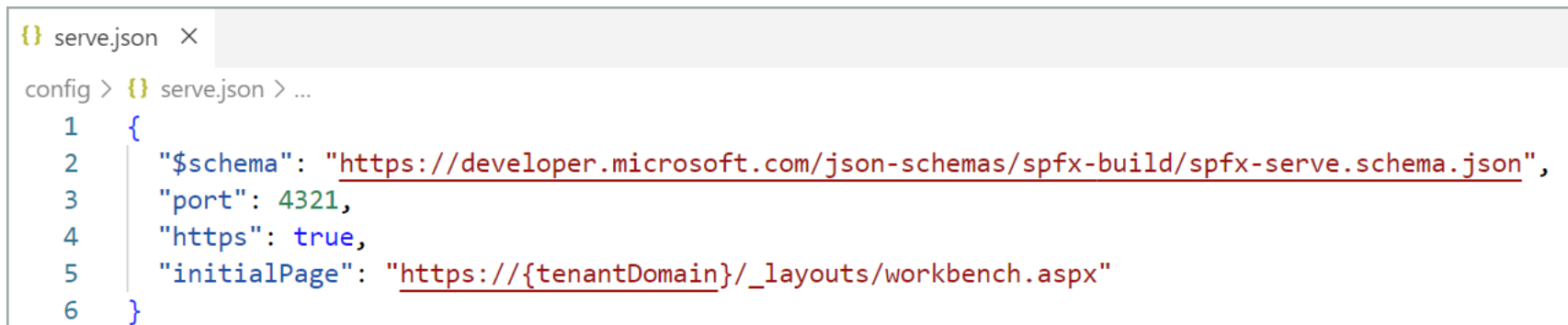


Testing a Web Part in the Workbench

Use the **gulp serve** task to start a debugging session. This will run the **build** and **bundle** tasks, start the local web server, launch the default browser, and load the workbench page at the URL configured in the **initialPage** property of the object stored in **config/serve.json**.

The default value of the **initialPage** property (shown in the image below) contains a **tenantDomain** token that will be replaced by the site URL contained in the **SPFX_SERVE_TENANT_DOMAIN** environment variable which you would need to add on your development machine. This is a good option when you commonly want to test different projects in the context of the same site.

The alternative is to set the value of the **initialPage** property to the URL of the workbench page in a specific site. This is a good option when you commonly want to test different projects in the context of different sites.

A screenshot of a code editor window titled 'serve.json'. The editor shows a JSON configuration object with the following properties: '\$schema' pointing to a Microsoft JSON schema URL, 'port' set to 4321, 'https' set to true, and 'initialPage' set to a URL template 'https://{tenantDomain}/_layouts/workbench.aspx'. The code is syntax-highlighted with colors for strings, numbers, and keywords.

```
{  
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/spfx-serve.schema.json",  
  "port": 4321,  
  "https": true,  
  "initialPage": "https://{tenantDomain}/_layouts/workbench.aspx"  
}
```

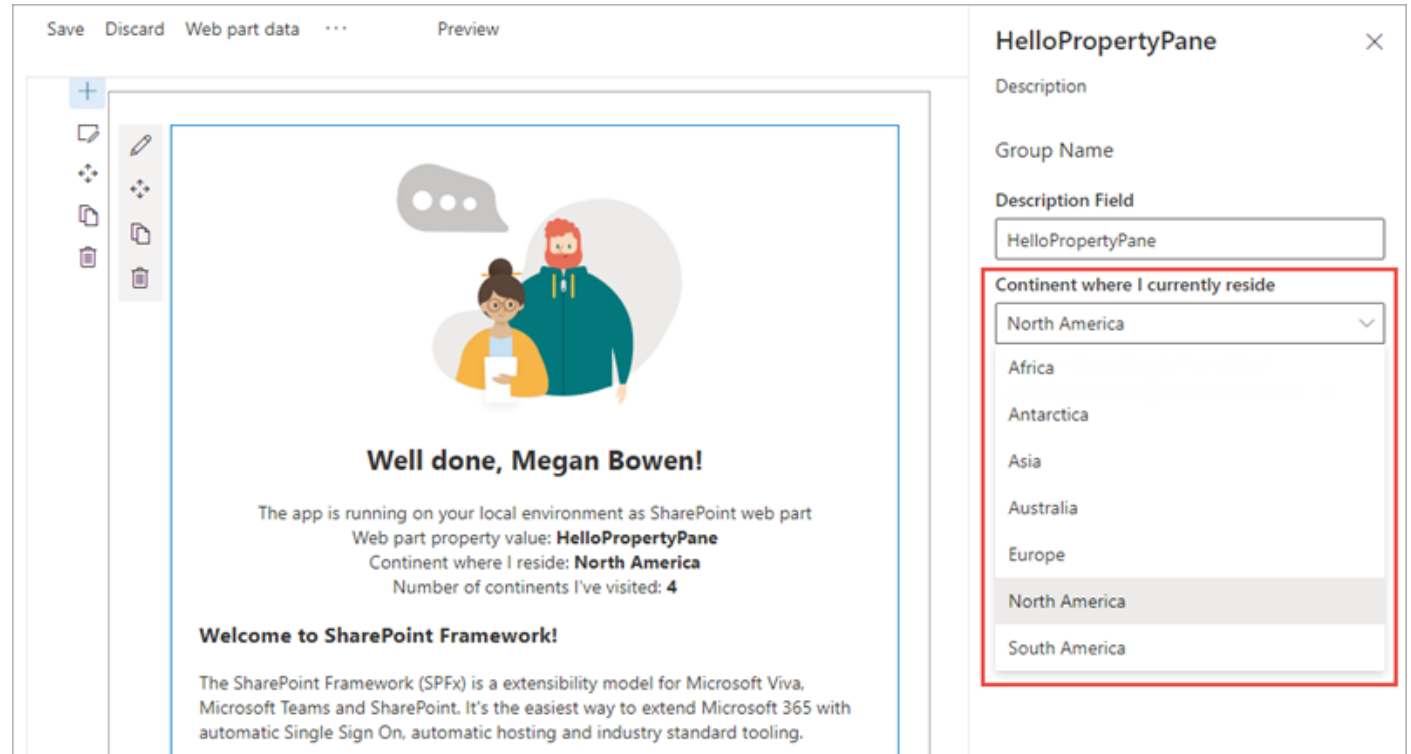
Adding Web Part Properties

- Define an interface in your web part that includes one or more properties
- Import the corresponding field types in the web part class
- Field types are available as modules in the **@microsoft/sp-property-pane** library
- Modify the default **getPropertyPaneConfiguration** method and add the properties to the **groupFields** array

```
groupFields: [  
  PropertyPaneTextField('description', {  
    label: strings.DescriptionFieldLabel  
  }),  
  PropertyPaneTextField('listName', {  
    label: "List Name"  
  })  
]
```

Native Property Pane Field Types

- Property pane supports the following field types out-of-the-box
 - Label
 - Textbox
 - Multi-line Textbox
 - Checkbox
 - Dropdown
 - Link
 - Slider
 - Toggle



SharePoint Framework and the REST API

- SPFx implements calls to SharePoint REST API via the **SPHttpClient**
- Available from the web part context
- Based on the existing **HttpClient** API
- Handles the authentication and default config settings:
 - Authorization HTTP header
 - OData v4
 - Minimal metadata returned

SharePoint Framework and the REST API

```
private async _getListItems(): Promise<ICountryListItem[]> {  
    const response = await this.context.spHttpClient.get(  
        this.context.pageContext.web.absoluteUrl +  
        `/_api/web/lists/getbytitle('Countries')/items?$select=Id,Title`,  
        SPHttpClient.configurations.v1);  
  
    if (!response.ok) {  
        const responseText = await response.text();  
        throw new Error(responseText);  
    }  
  
    const responseJson = await response.json();  
  
    return responseJson.value as ICountryListItem[];  
}
```

Web Part Styles

```
@import '~@microsoft/sp-office-ui-fabric-core/dist/sass/SPFabricCore.scss';

.helloWorld {
  overflow: hidden;
  padding: 1em;
  color: "[theme:bodyText, default: #323130]";
  color: var(--bodyText);
  &.teams {
    font-family: $ms-font-family-fallbacks;
  }
}

.welcome {
  text-align: center;
}
```

- Defines the web part styles

Deployment – SharePoint Packages

Deployment of SharePoint Framework assets is done via SharePoint packages. A SharePoint package is a ZIP file with a SPPKG extension. The steps to create a SharePoint package for production deployment are:

- Run **gulp bundle --ship** to build the solution and create a minified bundle. The generated files are placed in both the **dist** folder and the **release/assets** folder.
- Run **gulp package-solution --ship** to generate/collect the files necessary for deployment. This includes the app manifest, feature definitions, element manifests, the bundle, the component manifest(s) and string localization files. All these files are placed into the SharePoint package. The SharePoint package is placed in the **sharepoint/solution** folder. The files contained in the SharePoint package can also be found in the **sharepoint/solution/debug** folder.

After creating a SharePoint package, you need to upload it to an app catalog.

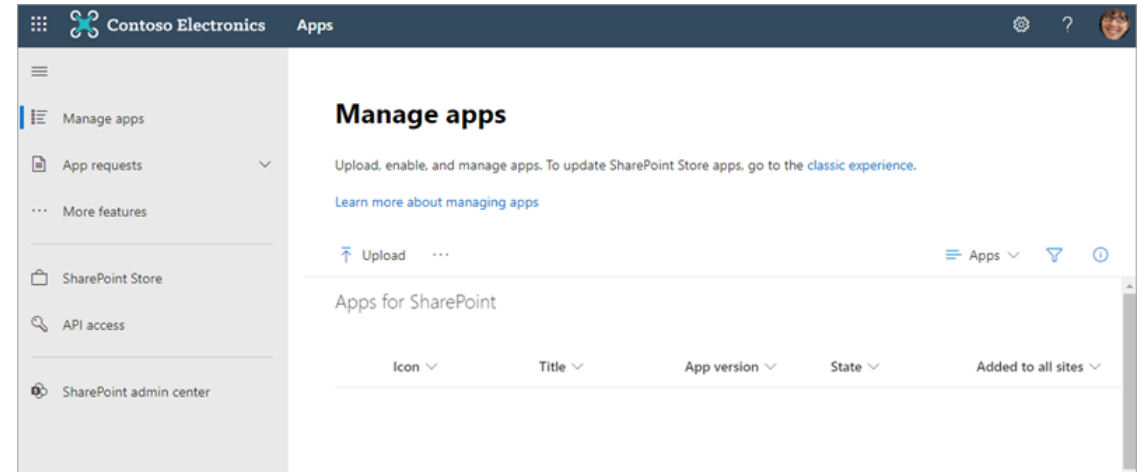
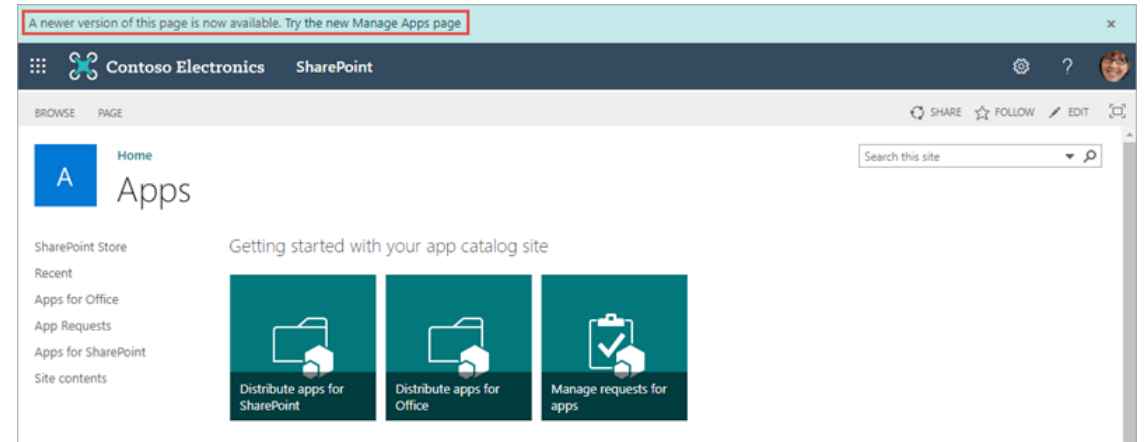
Deployment – Tenant App Catalog

Each tenant can have exactly one tenant app catalog. This is a site collection that is automatically created (if it doesn't already exist) when you go to **More features | Apps** in the SharePoint Admin Center.

Microsoft is in the process of transitioning from the classic user experience to a modern user experience for the tenant app catalog.

Navigate to **<https://{your-tenant-domain}.sharepoint.com/sites/appcatalog>** for the classic app catalog user experience.

Navigate to **https://{your-tenant-domain}.sharepoint.com/sites/appcatalog/_layouts/15/tenantAppCatalog.aspx** for the modern app catalog user experience.



Deployment – Site Collection App Catalogs

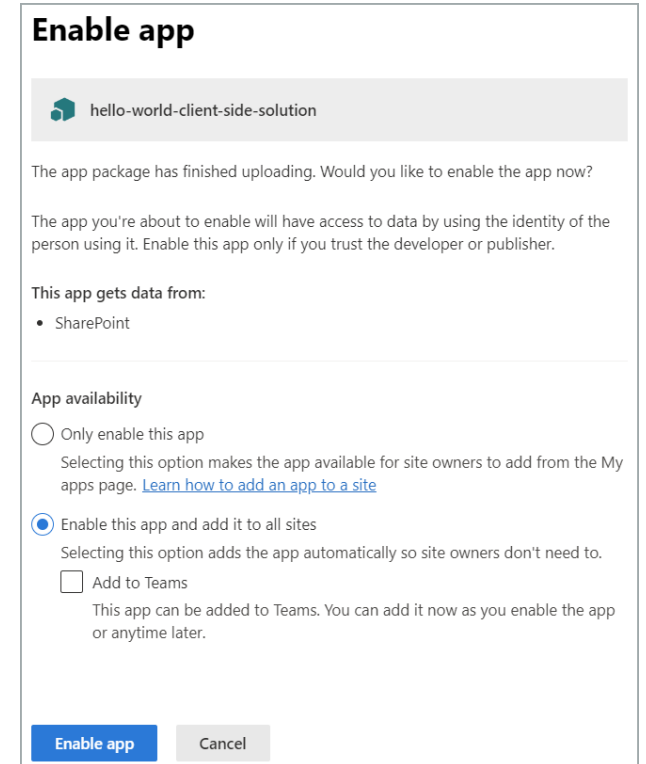
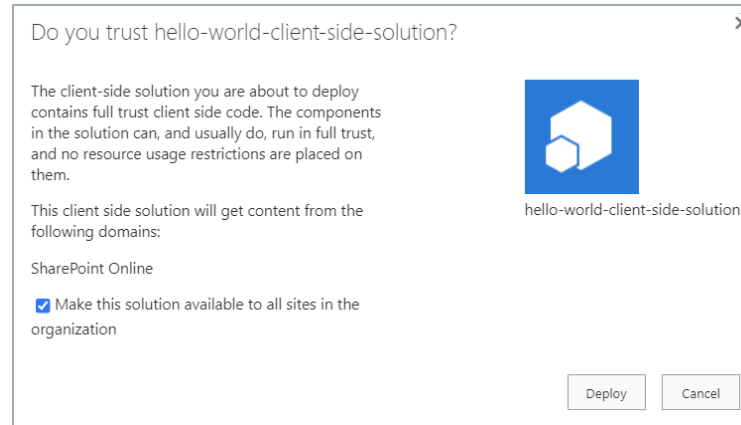
Administrators can add a site collection app catalog to any site collection.

- Use **PnP Powershell** or **CLI for Microsoft 365** to add app catalog
- Adds a list named **Apps for SharePoint** to root site
- Can limit scope of deployment to site collection rather than the entire tenant
- Site collection administrators can deploy SharePoint packages to the site collection app catalog
- Site collection app catalogs use the classic user experience

Deployment – Trust Dialog

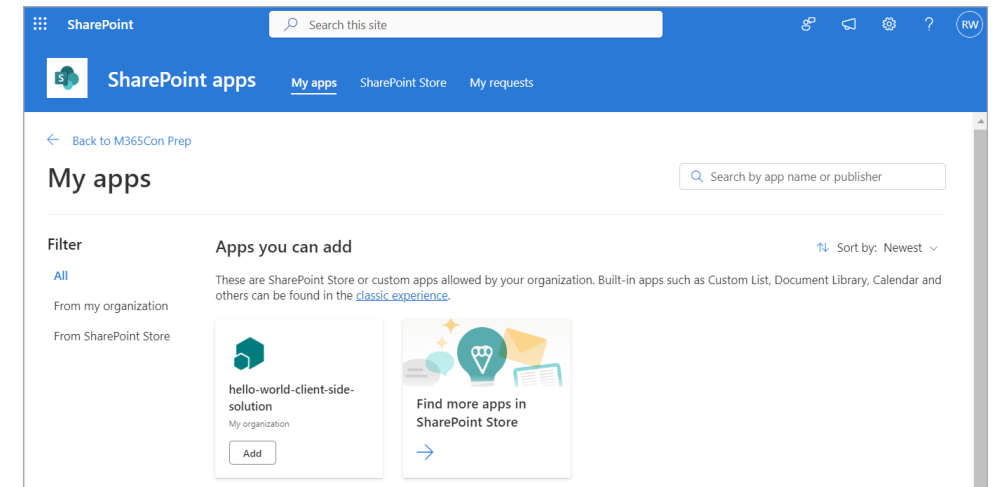
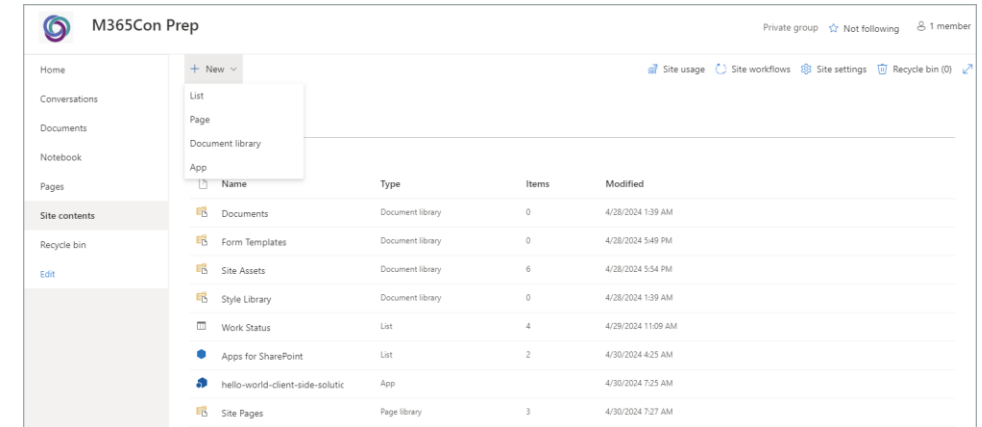
After uploading a SharePoint package to an app catalog, you will be shown a trust dialog or side panel.

- Select **Deploy** or **Enable app** to indicate you trust the package you just uploaded. This will complete the deployment process.



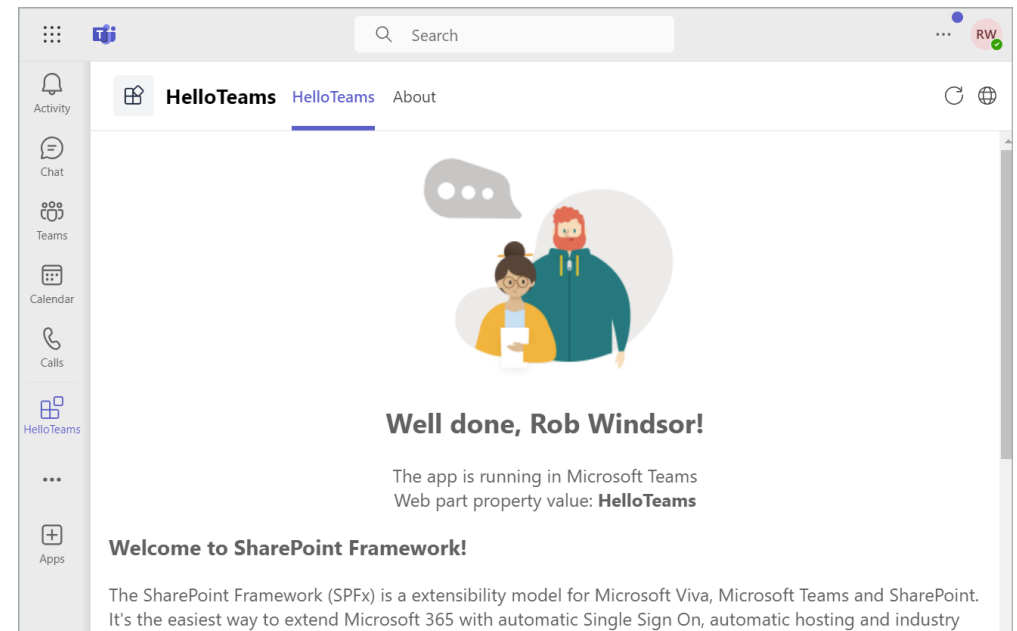
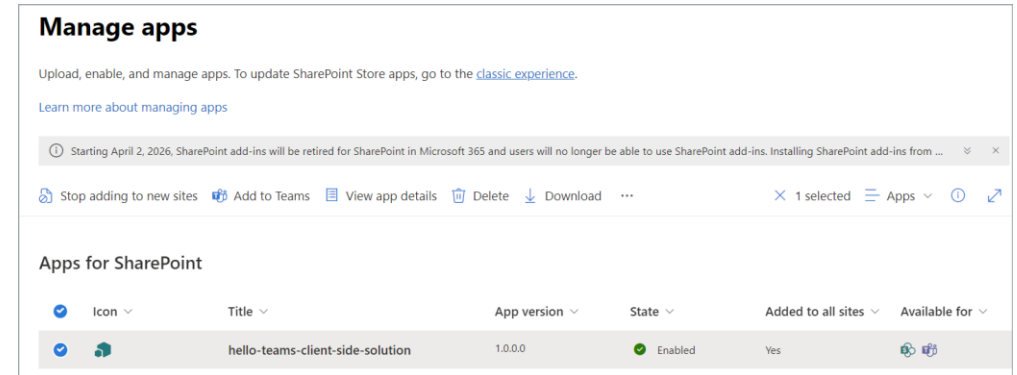
Installing SharePoint Framework Components

- Navigate to Site contents in the site where you wish to install the component(s)
- From the **New** menu, select **App**
- In the **My Apps** page, select the SharePoint package containing the component(s) you wish to install



Using Client-side Web Parts in Teams

- Ensure the **supportedHosts** property in the web part manifest includes **TeamsTab** and/or **TeamsPersonalApp**
- Deploy the SharePoint package to the tenant app catalog and configure it for tenant wide deployment
- Select the package in the **Manage Apps** page and then select the **Add to Teams** button in the command bar.



Adding configuration properties to React Client-Side web parts

Handle the property values in the React component when the component starts and whenever the property value is updated

```
public componentDidMount(): void {  
    this.doSomething(this.props.description);  
}  
  
public componentDidUpdate(prevProps: IHelloworldWebPartProps,  
                           prevState: IHelloworldWebPartState): void {  
    if (this.props.description !== prevProps.description) {  
        this.doSomething(this.props.description);  
    }  
}  
  
private doSomething(description: string): void {  
    // Do something with the property  
}
```

Resources

Overview of the SharePoint Framework

<https://learn.microsoft.com/sharepoint/dev/spfx/sharepoint-framework-overview>

Make your SharePoint Client-Side Web Part Configurable

<https://learn.microsoft.com/sharepoint/dev/spfx/web-parts/basics/integrate-with-property-pane>

Build Custom Controls for the Property Pane

<https://learn.microsoft.com/sharepoint/dev/spfx/web-parts/guidance/build-custom-property-pane-controls>

Reusable Property Panel Controls for SharePoint Framework Solutions

<https://sharepoint.github.io/sp-dev-fx-property-controls/>