Creating Data Integrations with SharePoint Framework

CREATING AND USING COMPONENTS IN SPFX PROJECTS

JS PADOAN
ARCHITECT & MICROSOFT CERTIFIED TRAINER

@JsPadoan https://www.linkedin.com/in/jspadoan

Overview

Business Scenario

Technical prerequisites

Sharing Code between SPFx Projects

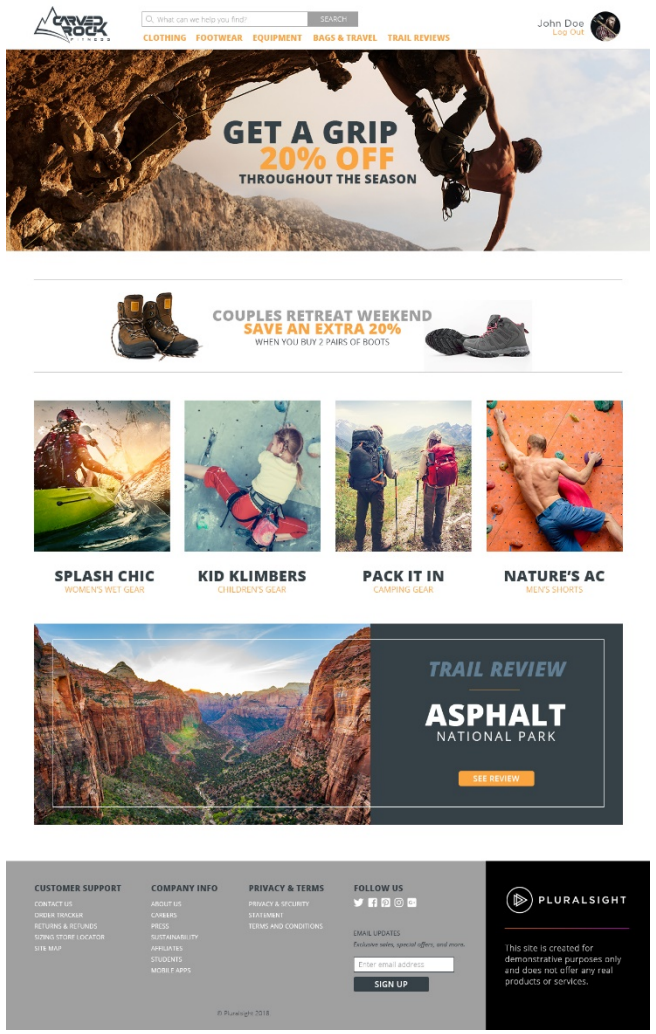
Connecting with SharePoint APIs

Business Context

You are Franck, a developer ...

You work for Carved Rock Fitness...

You will make some custom developments in SharePoint Online



Business requirements for the custom developments :

- Promote into the Home Page the last “Positive” comments made by employees on most recent article page.

- Capture the latest News in real time, determine their sentiment and present them in an attractive carousel.

- Create a WebPart to display Current User Information as well as the number of Teams of which the current user is a member.

SHOW Final Result here

Technical Prerequisites

Provision your Microsoft 365 Tenant

Microsoft 365 Tenant

https://docs.microsoft.com/en-us/office/developer-program/microsoft-365-developer-program

Provision your Microsoft 365 Tenant

Microsoft 365 Tenant

https://docs.microsoft.com/en-us/office/developer-program/microsoft-365-developer-program

Provision your Microsoft 365 Tenant

Microsoft 365 Tenant

App Catalog

https://docs.microsoft.com/en-us/sharepoint/use-app-catalog

Set up your Developer Environment

Node.JS v10.x

VS Code (or other client-side IDE)

Gulp : Task runner

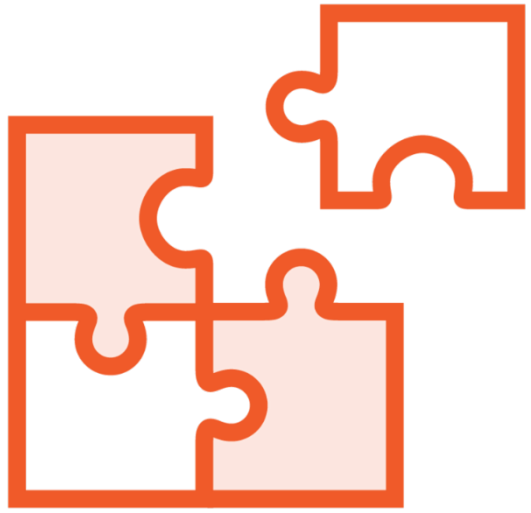
Yeoman : Project Generator

npm install gulp yo @microsoft/generator-sharepoint –global

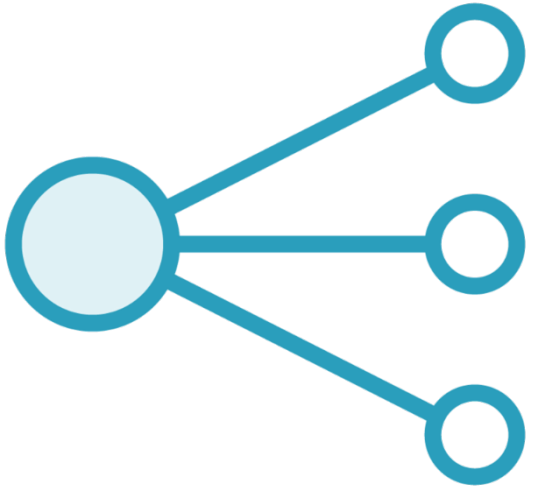
Sharing Code Between SPFx Projects

Overview of Library Component

Set of tools
(classes, methods, etc.)



Is simply instanciated and manipulated



Can be used by multiple other Components

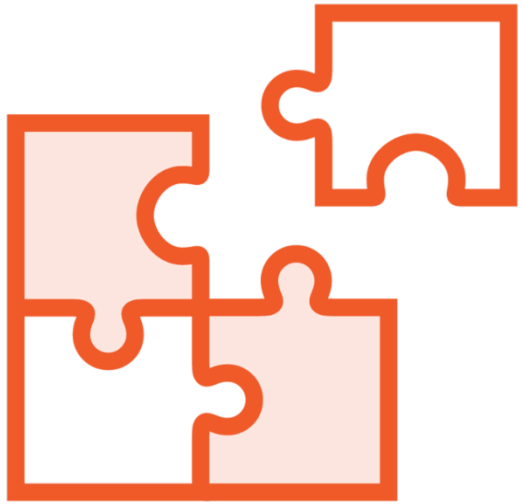
Creating a new Library component project

Yeoman “Library” Project

Create your Classes and Methods

Update **Index.ts** to list the exported members

Using a Library Component in a SPFx project (for local testing)

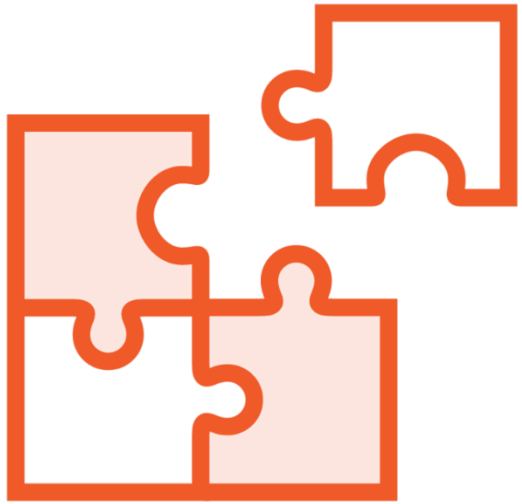


Create a Symbolic Link from the Library Project

Instantiate and Manipulate

npm link

Using a Library Component in a SPFx project (for local testing)



Create a Symbolic Link from the Library Project

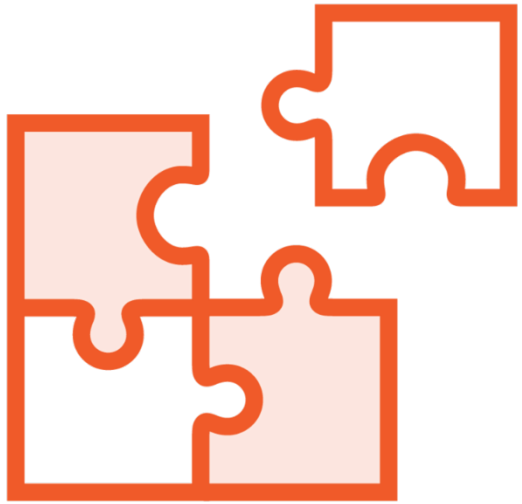
Instantiate and Manipulate

npm link

Using a Library Component in a SPFx project (for local testing)

Create a Symbolic Link from the Library Project

Reference the Symbolic Link from the "Client" Project



Instantiate and Manipulate

npm link your-library

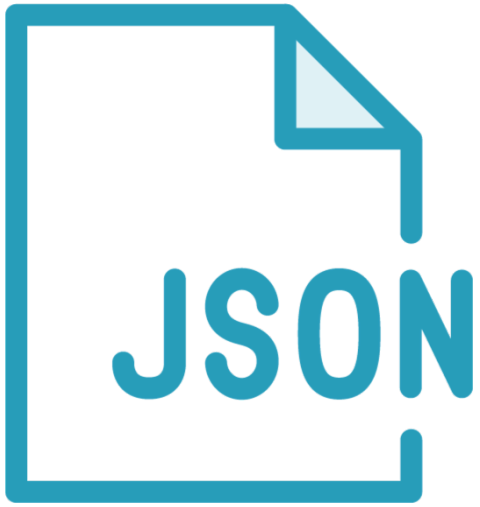
Deploying a Library Component

Bundle then Package with Gulp

Deploy Library To App Catalog

```
gulp bundle --ship
gulp package-solution --ship
```

Using a Library Component in a SPFx project (from tenant App Catalog)

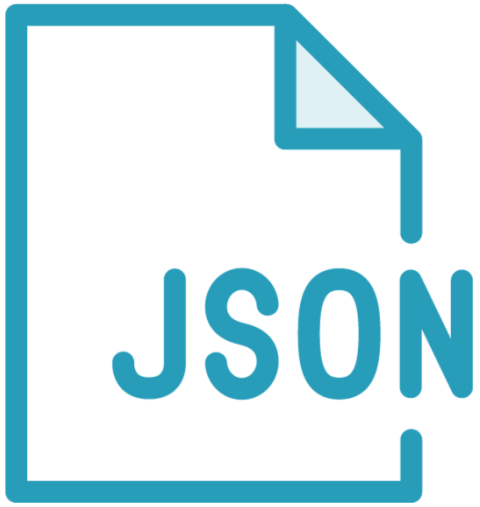


Update Package.json

Add a Dependency for your Library

```
"dependencies": {
  “CarvedRock-library": "0.0.1", // here we added the reference to the library
  "@microsoft/sp-core-library": "1.9.0",
  "@types/webpack-env": "1.13.1",
  "@types/es6-promise": "0.0.33"
},
```

Using a Library Component in a SPFx project (from tenant App Catalog)



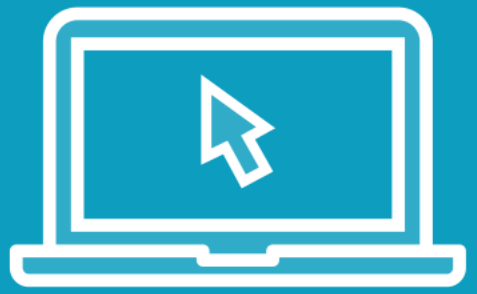
Update Package.json

Bundle then Package with Gulp

Deploy WebPart To App Catalog

```
gulp bundle --ship
gulp package-solution --ship
```

Demo



Create and use a Library component with SPFx :

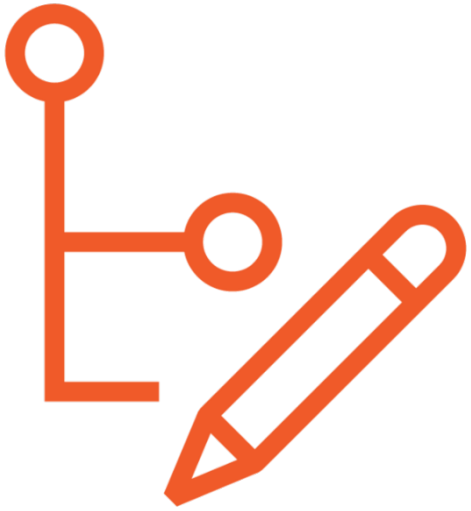
- Create a new Library Component project
- Reuse that library locally in a WebPart
- Deploy the Library
- Prepare the WebPart to use a remote Library Component
- Deploy the WebPart

Update the Library to validate the Process

Connecting with SharePoint APIs

Presentation of SharePoint APIs

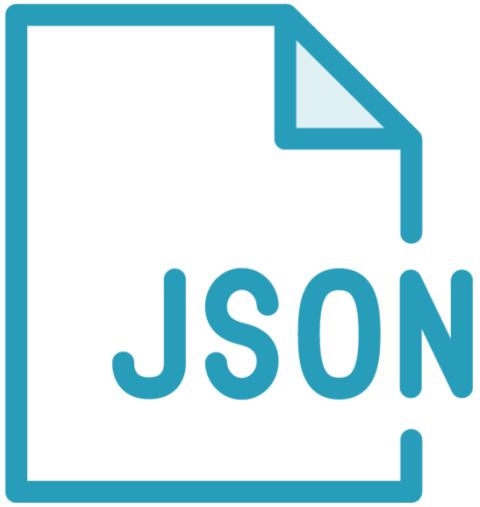
Set of Tools accessible to Developers



Allow Manipulation of SharePoint Components

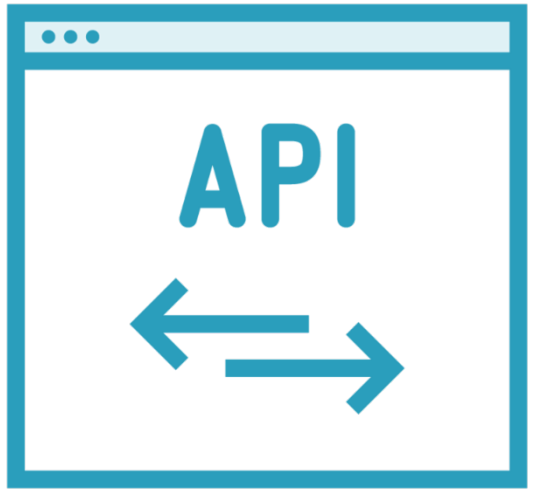
Can Manage common SharePoint Services (User Profile, Search, ...)

Connecting to SharePoint APIs



REST Paradigm

Secured by essence (Calls must use the current context)



Calls must be made through SPHttpClient

Using SharePoint Lists & Services APIs

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

1

```
…

Let c = context.spHttpClient;
```

1

```
let response = await this._spHttpClient.get(“URL?Parameters=value"
SPHttpClient.configurations.v1,
        {
        headers: {
            'Accept': 'application/json;odata=nometadata',
            'odata-version': ''
        }
});

let body = await response.json();
```

2

3

4

Demo

Manipulate SharePoint APIs

Create a WebPart (EmployeeProfile) :

- Connect to User Profile Service API
- Retrieve details for the connected user

Create a WebPart (ArticleComments) :

- Retrieve Comments on an Article Page
- Connect to SharePoint List API

Summary

Technical prerequisites

Sharing Code between SPFx Projects

Connecting with SharePoint APIs