Overview of SharePoint Online and how to integrate modern JavaScript Libraries using SPFx.

Maine.JS – Meeting 3

Speaker: Jamie Davis

Date: February 15th, 2023

Introduction

- Today I am going to discuss how to integrate modern JavaScript libraries with SharePoint using SharePoint Framework (SPFx).
- First off what is SharePoint? SharePoint is a collaboration tool used for:
  - Creating team-based websites to share data, news and resources.
  - Any SharePoint data may be accessed via an API to create intelligent dashboards, applications that can be used in the organization.
  - Provide a place to store team documents (like AWS S3).
  - I define SharePoint as a technology that allow you to create very sophisticated intranet sites.

Benefits of SharePoint

- One benefit to using SharePoint for a team site is you do not have to worry about security.
- This is built into the software and provides the ability to allow/deny access to the site by user. This is nice because as a developer you can focus on creating an application w/o worrying about security.
- Support for jQuery / JavaScript at least 10 years through their Script editor. This feature allowed intelligent, secure apps to be created.
- One use case is creating a dashboard using jQuery DataTables.
- Another one is creating a form-based apps that updates list / database data.

What SPFx / SharePoint online is replacing

- SharePoint has traditionally been managed in an on-prem environment. This means the organization would have a SharePoint server in their datacenter.
- Microsoft has deprecated SharePoint on-prem this year and wants everyone to move to their cloud.
- The Microsoft cloud has been rebranded from o365 / office 365 to Microsoft 365. They seem to rebrand their products so much it is hard to keep track. Hopefully they just keep Microsoft 365.

What is SharePoint Online / SPFx

- SharePoint online is the cloud version of SharePoint on-prem.
- What is it? Per Microsoft “SharePoint Framework (SPFx) is a page and web part model that provides full support for client-side SharePoint development, easy integration with SharePoint data, and extending Microsoft Teams”.
- “With the SharePoint Framework, you can use modern web technologies and tools in your preferred development environment to build productive experiences and apps that are responsive and mobile-ready.”

What is SharePoint Framework continued

- Due to tight integration between SharePoint Online, Microsoft Teams, and Microsoft Viva Connections, developers can also use SPFx to customize and extend all of these products
- It runs in the context of the current user and connection in the browser. iFrames are not required!
- The controls are rendered in the normal page DOM and are responsive and accessible by nature.
- It's framework-agnostic. You can use any JavaScript framework that you like including, but not limited to, React, Svelte, Jquery, anything.

Create scaffolding

- The developer toolchain is based on popular open-source client development tools such as NPM, TypeScript, Yeoman, webpack, and gulp.
- Microsoft has created a SharePoint Online SPFx Yeoman Generator which helps you quickly create a SharePoint client-side solution
- When running the yeoman generator, a series of questions will be asked, the answers which will generate the scaffolding you are looking for
- To create a new project, just run the command *yo @microsoft/sharepoint,* answer questions, afterwards all dependencies, scaffolding will be automatically set up.

Three types of default scaffolding

- The generator supports three types of scaffolding to be setup:
  - No framework : The default webpart file has the minimum code needed to get started. The render(), onInit() methods does not have any boilerplate.
  - Minimal: The render(), onInit() methods contain a lot of boilerplate, additional methods like _getEnvironmentMessage, onThemeChanged added.
  - React – Creates basic React project. Default webpart identical to except includes React code in the render() method.
- I used “no framework” for the jQuery/DataTables and Svelte examples.

High level explanation of scaffolding

- After the yeoman generator has finished, these directories will be present.
  - Config (place to store config data). Notable files:
    - serve.json:

      This is the place where you add a link to your tenant for previewing any build. This is done by running gulp serve.
      - Where any links to external CDN's are added.
    - package-solution.json: Config used when building the solution file that is uploaded to the app catalog. Before every build update the version numbers in this file, otherwise SharePoint will complain.
  - Teams:
    - Folder to add teams specific code, icons. SPFx as mentioned before can be integrated with Microsoft Teams as well.

High level explanation of scaffolding (cont)

- src:
  - webparts: You will spend most of your time adding code to this folder.
    - The yeoman generator creates a default WebPart and manifest file:
    - The webpart file is by far the most important because it is where the application is rendered. It is in this file where you will link the main App component.
    - This app component then may have supporting child components.
  - Components: If react project default components are created.

Local development / publishing to App Catalog

- SPFx allows you to test your changes out locally before uploading the solution file to your tenant's App Catalog.
- This is done by making changes to the app and running gulp serve.
- gulp serve launches a workbench in a browser tab, at this point you will select your local webpart.
- Once you are happy with the changes, increment the version number in the config/package-solution.json and run gulp bundle –ship and gulp package-solution –ship. This command creates a solution file.
- Then go to app catalog and upload.

Integrate with your favorite JavaScript library

- I will show you three integrations
  - DataTables / jQuery: Search for movies
  - Svelte – Proof of Concept, display rendering Svelte components
  - React: Provide example of Form Updating a SharePoint list
- DataTables:
  - Need to add links to external CDN's in the config/config.json file.
- Svelte:
  - The main challenge was linking the Svelte app to the WebPart's domElement method.
- React:
  - Supported out of the box.
  - Microsoft has a library named FluentUI which is like Bootstrap.

References

- https://learn.microsoft.com/en-us/sharepoint/dev/spfx/sharepoint-framework-overview
- https://www.microsoft.com/en-us/microsoft-365/sharepoint/collaboration
- https://support.microsoft.com/en-us/office/what-is-sharepoint-97b915e6-651b-43b2-827d-fb25777f446f
- https://developer.microsoft.com/en-us/fluentui#/controls/web FluentUI controls (MS version of Bootstrap)
- The default Webpart generated by the yeoman generator has a lot of helpful links as well.