SharePoint Framework (SPFx)

Overview of the SharePoint Framework

SharePoint Framework (SPFx) is a web part model or a page. SharePoint Framework is a smooth integration with the SharePoint Data and open source tooling. It provides full support for client-side SharePoint development.

SharePoint Framework (SPFx) works for SharePoint On-premise (SharePoint2016 Feature Pack 2 and SharePoint 2019) and also for SharePoint Online.

Key features of SharePoint Framework

There are some key features in SharePoint Framework which is in below:

There is no iFrame customization in the SharePoint Framework. That means JavaScript is embedded directly to the page.

SharePoint Framework runs in the context of the current user and connection in the browser.

SharePoint Framework controls are responsible or accessible by nature.

Performance of the SharePoint Framework is reliable.

It is Framework-agnostic. That means it can use any JavaScript Framework like “NodeJS“, “ReactJS“, “Handlebars“, “Knockout“, “AngularJS” etc.

End users can use this SharePoint Framework(SPFx) that are approved by the tenant administrator on all sites including “Team Site”, “Group/Personal sites” etc.

SharePoint Framework (SPFx) web part can be added to both the Classical Pages and as well as Modern Pages.

For a C# developer, “Typescript” is an easy implementation method into the JavaScript world. There are some developers those likes to use the IDE Visual Studio Code, ATOM, Sublime etc.

Why the SharePoint Framework?

Previously, there are two development models we use in SharePoint Online:

1. Client-side JS injection
2. SharePoint Add-ins

1. Client-side JS injection:

Basically, in the Client-side Js, We are using the Script Editor which is very popular in the SharePoint Online. In that Script Editor, Simply you just paste the JavaScript code and it will execute the code when the page renders. It runs in the same browser context with the same DOM.

While you are working with your Solution, if the end-user will drop the control on to the page, then the configuration option will not work. So for that, the end-user can edit the page and modify the script which can break the web part.

Disadvantages of Client-side Js:

There is no available of "Safe for Scripting" in the Script Editor web Part. There are present some Site Collections as like "Team Sites", "Group Sites" has a feature known as "No-Script" enabled.

2. SharePoint Add-ins:

One most useful advantage of SharePoint Add-ins is that it has no access to the current DOM or connection. Also, it is external to the system.

Because of this advantage, the End-user can easily install the "Add-ins" on the No-script sites and use it.

Disadvantages of SharePoint Add-ins:

First, it runs in the iFrame. Iframes are slower than the Script Editor Web Part.

iFrame has stronger security which can be very useful for the end-users. That means the control has no access to their connection to office 365.

To overcome these issues, Microsoft comes with SharePoint framework development model. And in SharePoint Online modern experience we can use only the SharePoint framework for SharePoint web part development.

Different SharePoint Development Models

There are various development model available in SharePoint from previous versions to till now.

SharePoint Framework:

- SharePoint Framework customizations execute within the client.
- Fully script based, not compiled and run in the context of the current page.
- No more iFrame concept
- This is the only one way to customize the SharePoint modern pages.
- They are fully responsive and mobile friendly.
- Can be uploaded to tenant's app catalog and is available in any site within the tenant.

SharePoint Farm Solutions:

- Farm solutions or full trust solutions run on .NET framework.
- Farm solutions run under the context of SharePoint farm process and they have full access to the SharePoint farm.
- Deploy using wsp files and features
- Cannot be hosted to SharePoint Online Office 365
- Farm solution scope can be anything like farm, web application, site collection or site level.
- You can create server-side web parts, timer jobs, event receivers, feature receivers, but these things are not possible in spfx.

Sandboxed Solutions:

- Same like farm solutions but are specific to SharePoint site collection, rather to the whole farm.

- Sandboxed solutions are scoped to the site collection in which they are installed and activated.
- They cannot be activated to web application or farm level.
- We still can use sandboxed solutions to provision declarative solutions such as site columns, content types, list templates and instances.

SharePoint Apps or Add-in Model

- Next, development model comes as, SharePoint Add-ins. There are two types of Add-ins:
- SharePoint Hosted Add-in: You can use jsom code and they run within a client-side context. Any custom logic, you want to add, it has to be implemented using JavaScript.
- Provider Hosted Add-in: In provider hosted add-in you can deploy the web application out side of SharePoint, may be in on-premises servers or into Microsoft Azure. You can use CSOM (C#.Net) there.
- Even if you develop a web part, it will be available within iFrame.
- For deployment, the package file can be uploaded to the SharePoint app-catalog site.
- The app scope is tenant but, functionality is scoped to the site to where it was installed.

JavaScript Injection using Script editor or content editor web part.

- You can easily customize page by injecting JavaScript using Script editor or content editor web part.
- You can use jsom, rest api etc.
- It's very easy, you can just use any text editor to write the code.
- But the customizations are scoped to a specific page and there is no packaging method available.

SPFx Development Toolchain

The SharePoint Framework toolchain is the set of build tools, framework packages, and other items that manage building and deploying your client-side projects.

A Toolchain helps:

- To build client-side components such as web parts.
- Test client-side components in your local development environment by using tools such as the SharePoint Workbench.
- Enables us to package and deploy to SharePoint.
- Provides you with a set of build commands that help you complete key build tasks such as code compilation, packaging the client-side project into a SharePoint app package.

Whatever we are developing in SharePoint, we need to deploy to our SharePoint environment. We have wsp for SharePoint solutions, we have add-in packages for SharePoint Apps or Add-ins. So what for SPFx?

In SharePoint framework it is, SharePoint packages (*.sppkg) which nothing but a .Zip file. The package contains all the files needed for your customization as well as a few generated JSON and XML files SharePoint needs.

Now, what is the tool to create the package?

SharePoint Framework Development	SharePoint Classic Development
NodeJS	.NET Framework
npm	NuGet
Visual Studio Code/Yeoman	Visual Studio
Typescript	C#
Gulp	MS Build
webpack	

Node.js:

Traditionally we use Microsoft .Net framework for building web application.

In modern days, Node.js is doing this for lots of modern web development.

Node.js is an open source, cross-platform runtime environment for hosting and serving JavaScript code.

The Node.js ecosystem is tightly coupled with npm and task runners such as gulp to provide an efficient environment for building JavaScript-based applications. Node.js is similar to IIS Express or IIS, but includes tools to simplify client-side development.

Node Package Manager (npm):

npm is like NuGet from where we can add packages to use within our application.

SharePoint client-side development tools use the npm package manager to manage dependencies and other required JavaScript helpers. npm is typically included as part of Node.js setup.

Yeoman:

Previously, we usually use Visual studio to create a new project which usually create the required folders and sub folders.

But in SharePoint framework, to get files and folders structure, we need to use a new tool known as Yeoman,

Yeoman helps you to kick-start new projects. The Yeoman SharePoint generator is available as part of the framework to kick-start new client-side web part projects.

Gulp:

Once our project is ready, traditionally we use visual studio to build or compile the project. Visual studio internally calls MSBuild to do so.

In SharePoint framework, the tool Gulp does the same thing for us. It allows you to write tasks using JavaScript and then tell Gulp to run those tasks for you.

SharePoint client-side development tools use gulp as the build process task runner to:

- Bundle and minify JavaScript and CSS files.
- Run tools to call the bundling and minification tasks before each build.
- Compile TypeScript files to JavaScript.

TypeScript:

SharePoint client-side development tools are built using TypeScript classes, modules, and interfaces.

Webpack:

Webpack is a module bundler that takes your web part files and dependencies and generates one or more JavaScript bundles so that you can load different bundles for different scenarios.

JavaScript frameworks:

- React
- AngularJS 1.x
- Angular 2 for TypeScript 2.x
- Vue.js
- Handlebars

You can go for React framework.

Apart from this, have a look at SharePoint PnP JavaScript Core library, which provides easy access on SharePoint REST APIs.

Code Editors:

SharePoint Framework is client-side driven and thus you can use your choice of HTML/JavaScript code editors, such as:

- Visual Studio Code
- Atom
- Webstorm

Visual Studio Code is a lightweight but powerful source code editor from Microsoft that runs on your desktop and is available for Windows, Mac, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js, and has a rich ecosystem of extensions for other languages (such as C++, C#, Python, PHP) and runtimes.

SharePoint REST APIs:

SharePoint framework provide supports of Rest API to interact with SharePoint to provide functionality to web parts.

Patterns and Practices (PnP):

PnP provides code samples, patterns, and other resources to help you transform your existing solution to the SharePoint Framework.

Which Tools & Technologies you should focus?

Through Microsoft provides lots of tools and technologies, but you can focus majorly on below things:

- Npm
- TypeScript
- PnP

SharePoint client-side web parts:

SharePoint client-side web parts are controls that appear inside a SharePoint page but run locally in the browser. They're the building blocks of pages that appear on a SharePoint site.

You can build client-side web parts using modern script development tools and the SharePoint workbench (a development test surface), and you can deploy your client-side web parts to modern pages and classic web part pages in Office 365 tenants.

SharePoint Framework Extensions:

You can use SharePoint Framework (SPFx) Extensions to extend the SharePoint user experience. With SPFx, we can customize areas like notification areas, toolbars, and list data views.

SharePoint Framework Extensions enable you to extend the SharePoint user experience within modern pages and document libraries, while using the familiar SharePoint Framework tools and libraries for client-side development. Specifically, the SharePoint Framework includes three new extension types:

- **Application Customizers:** Adds scripts to the page, and accesses well-known HTML element placeholders and extends them with custom renderings.
- **Field Customizers:** Provides modified views to data for fields within a list.
- **Command Sets:** Extends the SharePoint command surfaces to add new actions, and provides client-side code that you can use to implement behaviours.