SPFX

The 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. ... The latest version, and all previous versions, of the SharePoint Framework are hosted and available in SharePoint Online.

TYPESCRIPT: tutorialspoint.com/typescript/index.htm

Fat Arrow notation => used for functions. Lambda function? Can be used to disregard the function keyword

=== The strict **equality** operator will compare both the value and type

SharePoint Clientside webparts – building blocks of pages. Controls.

Other frameworks you can build webparts with JS:

React (Office UI Fabric React – collection of UX Framework), Angular, Vue.js

Envi Setup

Powershell – built on top of .NET(Dev Framework created by microsoft). Task automation and management configuration framework. Performs administrative functions?

Node.JS – opensrc cross-platform, back-end JS runtime environment.

Interpreter – does not rely on compiler.

Scripting vs Programming – diff in their execution on compilation. Script uses interpreter while prog uses compiler.

Yeoman – helps you kick start a project. Sets up your dev envi for your proj/ sol. Creates project scaffoldings and installs required dependencies.

Yeoman sharepoint – specifically for sharepoint

NPM – package manager for JS. Handles installations.

Node.js ver – 10.x lts ver latest

Sharepoint online (cloud-based service) vs Sharepoint on-premise (server maintenance, hardware, more people)

Gulp.js – an opensrc javascript toolkit. Taskrunner

Package.json = contains the dependencies of the proj. Used by npm install

**SharePoint Online** helps organisations share and collaborate with colleagues, partners and customers, it allows groups to set up a centralised, secure space for document sharing, editing and downloading.

An **API** (application programming interface) is an information gateway that allows the back ends of software and services to communicate with one another. uses HTTP requests to access and use data*. Information that can be freely accessed by other users permissioned by web administrators cuz there are cases where we want to share data but secure some so we show it in the form of APIs.*

Code editor used – VS code

Gulp – task runner

Typescript – developed by MS. Superset of JS. Compiled while JS is interpreted.

Note : Any client-side web part should extend the BaseClientSideWebPart class to be defined as a valid web part.

**interface** defines the syntax that any entity must adhere to. **Interfaces** define properties, methods, and events, which are the members of the **interface**.

Render() Renders the look of your solution.

Propertypane – contained also inside the webpart main class. Handles the properties of the webpart

Json – contains the default values of your sol? For e.g. the value of ur properties which is found in the properties object

The **HelloWorldWebPart.manifest.json** file defines the web part metadata such as version, ID, display name, icon, and description. Every web part must contain this manifest.

this.context.pageContext – to access contextual information in ur webpart

Notice how ${ } is used to output the variable's value in the HTML block.

https://your-sharepoint-site-url/\_layouts/workbench.aspx.

* Web title
* Web absolute URL
* Web server-relative URL
* User sign-in name
* Can also retrieve lists using REST APIs

To grab APIs from ur site to ur webpart

import {

SPHttpClient,

SPHttpClientResponse

} from '@microsoft/sp-http';

Then need to identify the environment? See if local or if connected with ur SP site

import {

Environment,

EnvironmentType

} from '@microsoft/sp-core-library';

GULPS:

The --**ship** indicates the build tool to build for distribution.

 You used **--ship** to generate minified assets for distribution.

**trust-dev-cert**The task is used to install the developer certificates for testing the webparts. This is required before running the Gulp serve task. Applied only to one Workstation

**serve**  
The serve task is used to test the developed web part locally using workbench files.

**bundle**  
The bundle task will generate the scripts required to run the webpart on the site. The following snapshot shows the files generated.

**clean**  
The task will clear the temporary folders and files created in the solution. Some of the folders cleaned up during the process are **temp**, **dist**.

**build**  
The build task precompiles the code available and it has series of subtasks running similar to serve. This compiles the code and creates the necessary temporary files and folders.

a **build** is a version of a program. As a rule, a **build** is a pre-release version and as such is identified by a **build** number, rather than by a release number. ... As a verb, to **build** can mean either to write code or to put individual coded components of a program together.

**package-solution .sppkg**This task will create the solution package under sharepoint\solution folder required for deploying the webpart to the site. The **package-solution.json** file is present under config folder, which has basic information about the webpart.

Upload .sppkg to Tenant on the app catalog. Add the webpart on the site via app catalog.

**React** is an open-source, front end, JavaScript library for building user interfaces or UI components.

.tsx = main react component that yoeman added to the proj

Office UI Fabric and Office UI Fabric React - default frontend used by React

Provisioning assets from SP requires XML files (elements -structure and schema - attributes xmls) – these needs to be included in the JSON manifest as well

Schema (Layman’s) – pattern of thought or behavior

Basic Type script

Var varnam:datatype = “value”

Var key word can be function or others. Can also be disregarded or not typed literally

Prompt rendering is among the best features of **React** that gives a significant edge over Angular. ... The core difference between **reactjs** and **angularjs** is that **React** is **JS**-centric, while ng2 remains HTML-centric. JavaScript is far more robust, **than** HTML, that makes **React** far more simple, focused and consistent.

Framework vs library = you are incharge of the flow of the library while framework is incharge of the flow of library. Framework calls library

s

**PROPERTIES**

propertiesMetadata

getPropertyPaneConfiguration

3 key metadata of properties: Pages (contains header, groups), Header, Groups

Fields example = Button, checkbox, dropdown

Needs to import from **@microsoft/sp-property-pane**

OfficeFabricIconName = change name to change icon. Found in manifest json

iconImageUrl = use this instead if external image will be used

base 64 encoded image = diff format for iconimage url.

single part app page? When creating a single part app page

full width column? – spans the full width of the page. Additional section layout. Adjusted in manifest.json via Boolean called supportsfullbleed

hide webpart via manifest.json make hidden from toolbox true

PreconfiguredEntries – found in manifest json. Provides info about webpart toolbox. Default settings of webpart. **Can also be used to make multiple webparts with same logic**

There are seven pre-defined modern groups of webpart. This is used in the groupId found in manifest json to classify what part of group the webpart is.

OngetErrorMessage – validate the value of the webpart property

How to: 1. create a method in default class 2. Use ongeterrormessage on getpropertypane and reference the method created in def class.

How to use dropdowns? Import dropdown styles from ui fabric react. Then create asynchronous logic?

DEPLOYMENT

Need to make Domain is isolated to false

Yo-rc.json

Package-solution.json

NEW old ver to work

<https://stackoverflow.com/questions/67252829/yo-microsoft-sharepoint-fails-on-typeerror-lookups-flatmap-is-not-a-function>

npm install -g gulp@3.9.1

npm install -g yo@3.1.0