

A large, irregular red ink splatter or blotch serves as the background for the text. The splatter is centered and has a textured, painterly appearance with various shades of red and some darker spots. The text is white and positioned within the central part of the splatter.

Learn from My Mistakes – Tips for Building Better Solutions with SPFx

Presented By: Thomas Daly

Microsoft Office Servers and Services MVP

SoHo Dragon



About me

- SharePoint Team Lead @ SoHo Dragon - NYC
 - Branding & Developer for SharePoint / Office 365
- Focused on the UI side of things
- Community Involvement
 - Speaker [Branding & Front End Development]
 - [NJ SharePoint User Group](#) Organizer
 - [SharePoint Saturday NYC](#) Organizer
 - NJ & NYC Global Office 365 Dev Bootcamp Organizer 2017
 - [NJ Azure Bootcamp](#) Organizer
 - [SharePoint Saturday NJ](#) Organizer [2013-2014]
 - [My SharePoint Blog](#)
 - [Git Hub](#) [corp directory controls / o365 sticky footer / bootstrap navigation]





SharePoint / Office 365 / Azure / Data

Intranets - Migrations - Development

Recruiting - Staff Aug - Placement

**SOHO
DRAGON**



Outline

- Introduction
- Bundling
 - 3rd Part Libraries
 - Analyzing Bundles
- Externalizing References
- SP-PnP-JS
- Miscellaneous Tips

Introduction

Since the introduction of the SharePoint Framework (SPFx) Microsoft has painted it as this new and improved way of development



SPFx was said to be similar to other modern web dev practices



SPFx was TypeScript based, being easier for C# devs to learn



SPFx was to be framework agnostic - React, Knockout, Angular



MS & SPFx stood behind React as their choice for front end framework

SPFx Popularity

- A couple years later SPFx is very popular and widely adopted
- Endless amounts of information from Microsoft + Community
 - Video Tutorials + Demos
 - Bi-Weekly call with engineering teams + community members
 - Large collection of examples in GitHub
 - Large community support / blogs / forums



What Really Happened

- The people currently building SharePoint solutions were trying to learn and build SPFx solutions for modern with little to no experience with the toolset, framework, TypeScript.
- Steep Learning Curve
 - React
 - Syntax
 - State Management
 - Passing Props
 - TypeScript
 - ES6 Syntax
 - Interfaces
 - Toolchain – npm / gulp / webpack / bundling
 - Sass / CSS
 - Office UI Fabric

Problems in the Wild

Little to no
thought behind
the impact on the
tenant

Using different
versions of the
same libraries on
the page

Bloated Solutions
bundling all assets

Conflicts between
developers

Poor Architecture

No / Improper
Versioning

Unnecessarily
including 3rd party
libraries

Falling back to old
habits, doing
what's
comfortable

The background of the slide features a series of thin, curved lines in a light gray color, creating a sense of motion and depth. These lines are more prominent on the left side and fade towards the right.

Let's Get Started

- The following material has been based on working with various developers over the course of the past year
-Tips
- ...Tricks
- ...Optimizations
- ...Lessons Learned

Scenario

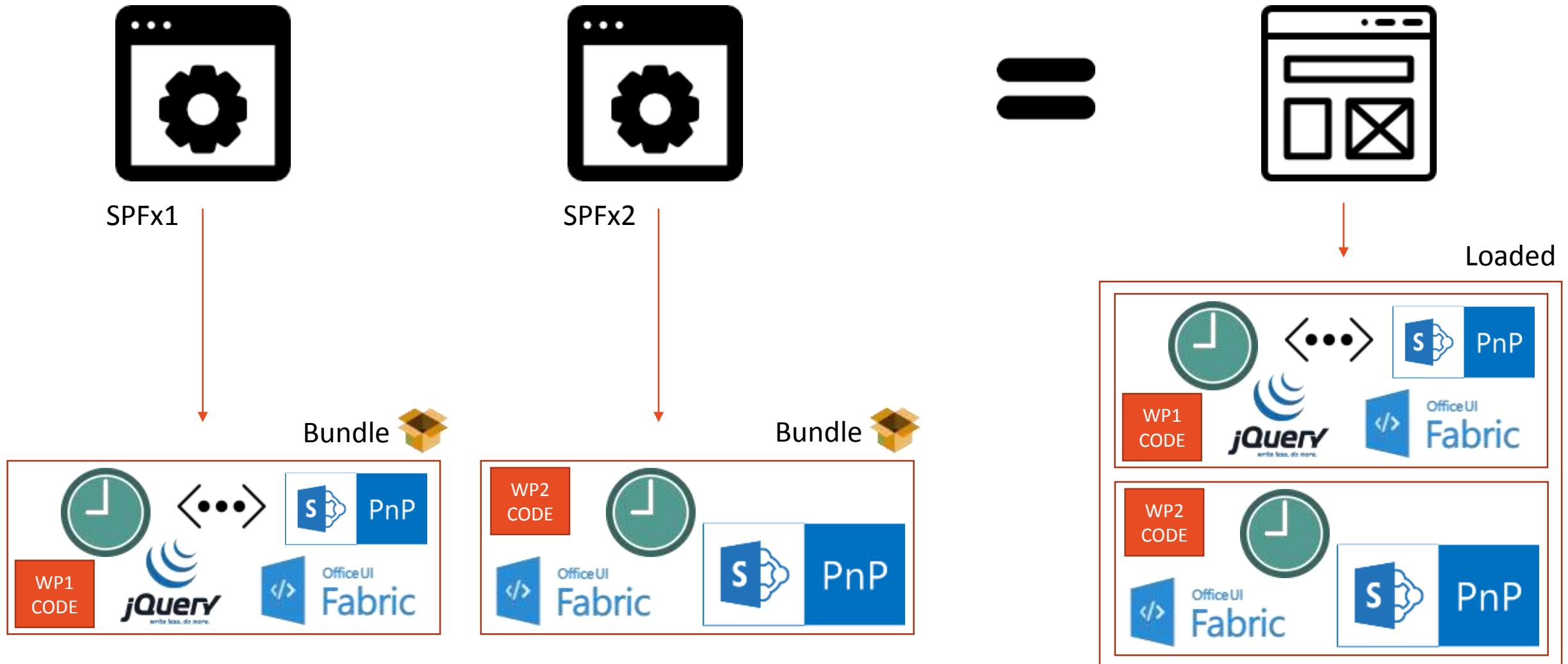
- We need to build a “thing” in SPFx
- Need libraries like:
 - Moment
 - Stock
 - Slider
 - Office UI Modal [for location selection]



Scenario (Continued)

- Adding those libraries increases the bundle size which contributes to load times causing the page to slow down
- If these were web parts we have multiple bundles bloating the page
- If other webparts were using the same packages we get 2 possible scenario's
 - Same libraries being loaded once or more on page load
 - Multiple versions of libraries being loaded on page load

As Solutions Grow



What Can We Do?

- Trim as much as you possibly can
- Identify the largest portions of code
- Identify where we can reuse packages
- Identify where we can remove packages
- Optimize Wherever Possible

Comments on 3rd Party Libraries

- Existing libraries allows you to be more productive
- Lets you focus on adding value for your organization
 - There are tons of helpful packages within the npm repository
 - Quicker to use than to build
 - You can get a sense of maturity by the statistics users, contributors, issues
- By default, 3rd libraries are bundled into your project
- Users end up downloading the same library multiple times with each component.
- The total page size grows significantly, taking longer to load and leading to a poor user experience, particularly on slower networks.

Why Bundle

- Makes deployment simpler
- Optimal to load 1 larger file vs many little files
- Loads modules in the order they are required
- Faster for end users

When to Bundle vs Externalize

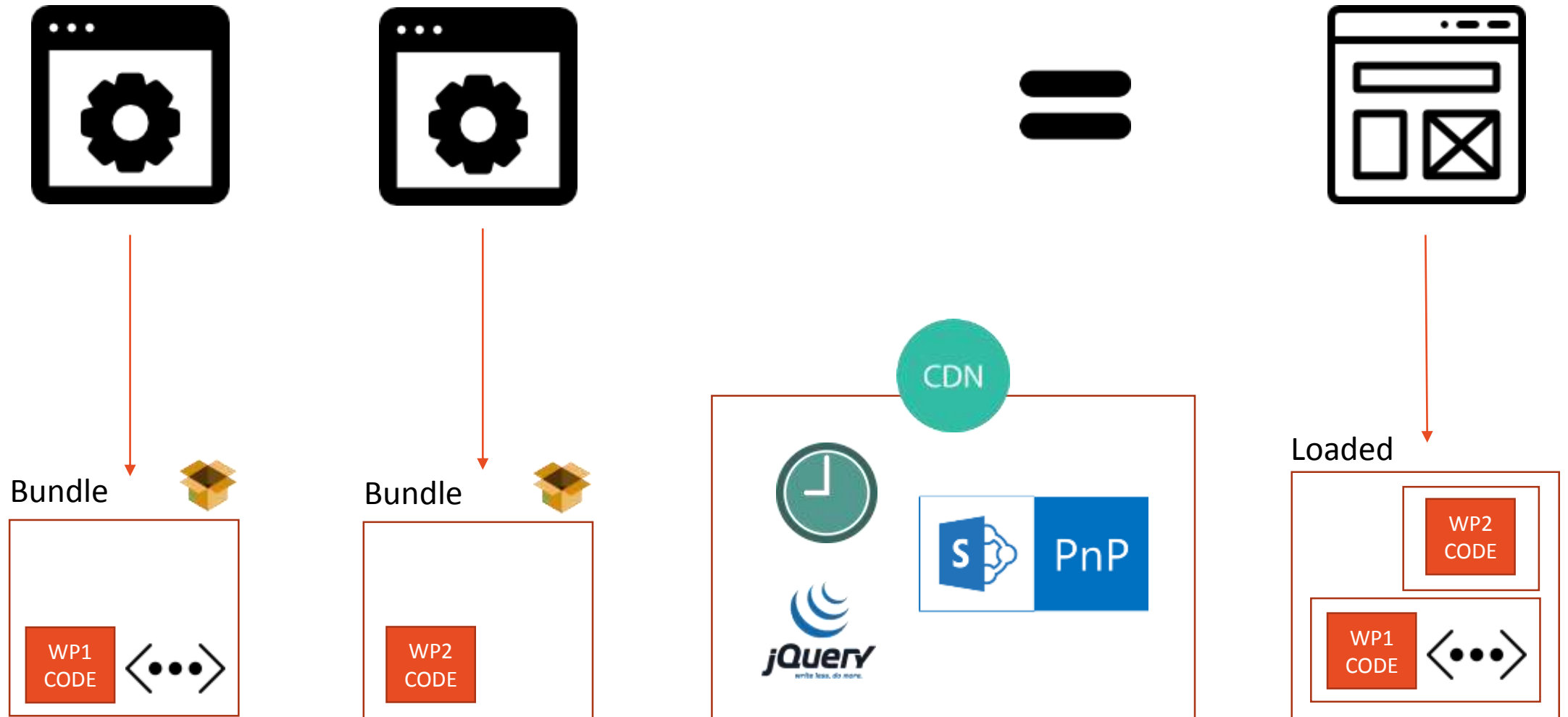
Bundle

- Lack of external internet access
- Blocking to particular websites
- Insulate yourself from external changes to CDN / code
- Maintain full control over assets & versions

Externalize

- Shared Libraries across multiple webparts
- Benefits of one time payload hit
- Generally speaking, Microsoft suggests this method

Externalize 3rd Party Libraries



Summary

- You can argue that you're web part bundle loads once and then you get the benefits of caching
- The benefit is loading the 3rd Party items once instead of 2, 3, 4x depending on web parts
- Recommended by SPFx to eternalize 3rd Party Libraries
- Recommend using CDN
 - Delivers static assets faster
 - 3rd Party libraries could already be cached



Externalizing Dependencies

Externalize Scripts

- Beware of types of JavaScript files
 - Module [AMD, UMD, CommonJS] vs Non-Module

Angular v1.x is a non-module script. You register it as an external resource in a SharePoint Framework project by specifying its URL and the name of the global variable it should register with:

```
"angular": {  
  "path": "https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.5.8/angular.min.js",  
  "globalName": "angular"  
}
```

It's important that the name specified in the `globalName` property corresponds to the name used by the script. That way it can correctly expose itself to other scripts that might depend on it.

- Don't know whether it's module or non-module?

[SharePoint Framework Script Check by Rencore](#)

Externalize Scripts w/ Dependencies

jQuery is an AMD script. To register it, you could simply use:

```
"jquery": "https://code.jquery.com/jquery-2.2.4.js"
```

Imagine now that you wanted to use jQuery with a jQuery plug-in that itself is distributed as a non-module script.

As mentioned previously, SharePoint Framework allows you to specify dependencies for non-module plug-ins. These dependencies are specified using the **globalDependencies** property:

```
"jquery": "https://code.jquery.com/jquery-2.2.4.js",  
"simpleWeather": {  
  "path": "https://cdnjs.cloudflare.com/ajax/libs/jquery.simpleWeather/3.1.0/jquery.simpleWeather.min.js",  
  "globalName": "jQuery",  
  "globalDependencies": [ "jquery" ]  
}
```

If you try to build the project now, you would get another error, this time stating that you can't specify a dependency to a non-module script.

Externalize Scripts w/ Dependencies (cont'd)

To solve this problem, all you need to do is to register jQuery as a non-module script:

```
"jquery": {  
  "path": "https://code.jquery.com/jquery-2.1.1.min.js",  
  "globalName": "jQuery"  
},  
"simpleWeather": {  
  "path": "https://cdnjs.cloudflare.com/ajax/libs/jquery.simpleWeather/3.1.0/jquery.simpleWeather.min.js",  
  "globalName": "jQuery",  
  "globalDependencies": [ "jquery" ]  
}
```

Handle Different Tenants

- Issue: external reference contain hard coded references to URL's and hosting location of the bundled files
- Requires manually modifying configuration files between builds

```
"sp-runtime": {  
  "path": "https://contoso.sharepoint.com/_layouts/15/SP.Runtime.js",  
  "globalName": "SP",  
  "globalDependencies": [  
    "microsoft-ajax"  
  ]  
},  
"sharepoint": {  
  "path": "https://contoso.sharepoint.com/_layouts/15/SP.js",  
  "globalName": "SP",  
  "globalDependencies": [  
    "sp-runtime"  
  ]  
}
```


Handle Different Tenants (cont'd)

- [spfx-build-url-rewrite by Wictor Wilén](#)
- Replaces https://contoso.sharepoint.com w/ target-cdn value

Rewrite using the command line

In order to re-write using the command line use the `--target-cdn` argument like this:

```
gulp build --target-cdn https://fabrikam.sharepoint.com
```

Reference CSS Stylesheets

- While the project configuration in the **config.json** file allows you to specify external resources, it applies only to scripts.

Load CSS from the URL using the SPComponentLoader

```
import { SPComponentLoader } from '@microsoft/sp-loader';
```

In the same file, override the onInit()

```
protected onInit(): Promise<void> {  
    SPComponentLoader.loadCss('https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.min.css');  
    return super.onInit();  
}
```

Use SPComponentLoader instead of require so assets do not get bundled

```
require('../../../../node_modules/jquery-ui/themes/base/core.css');  
require('../../../../node_modules/jquery-ui/themes/base/accordion.css');  
require('../../../../node_modules/jquery-ui/themes/base/theme.css');
```

Dynamic Bundling

- Conditionally loading CSS or JS
- Code can dynamically inject assets - does not increase the bundle size
- Assets load on the fly as the are needed
- David Warner Blog - [Dynamic SPFx Package Bundling](#)



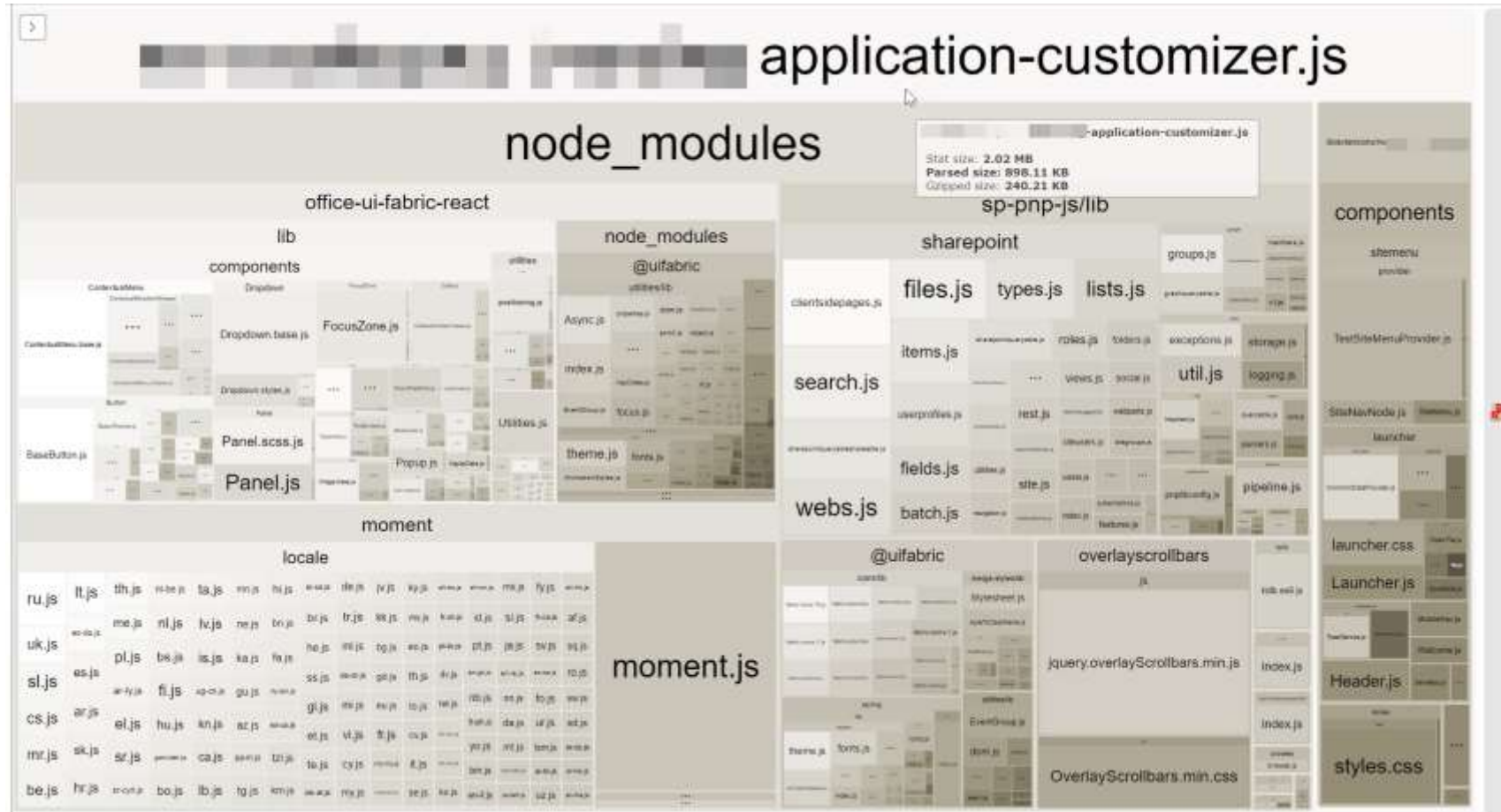
Measuring Bundles

Measuring Your Files

- The size of the .sppkg is NOT the size of the bundle
- Must make sure to build Production vs Development Builds (--ship)
- SPFx recommends webpack-bundle-analyzer
 - This measures all the combined JavaScript files in your solution (excluding external references)
- Chrome will show you full download / gzipped size or if cached

Webpack Bundle Analyzer

A visual html report each build demonstrating code size relative to each other



Install Webpack Bundle Analyzer

- Install the **webpack-bundle-analyzer** in your project directory

```
npm install webpack-bundle-analyzer --save-dev
```

- Change the gulpfile.js

```
'use strict';

const gulp = require('gulp');
const path = require('path');
const build = require('@microsoft/sp-build-web');
const bundleAnalyzer = require('webpack-bundle-analyzer');

build.configureWebpack.mergeConfig({
  additionalConfiguration: (generatedConfiguration) => {
    const lastDirName = path.basename(__dirname);
    const dropPath = path.join(__dirname, 'temp', 'stats');
    generatedConfiguration.plugins.push(new bundleAnalyzer.BundleAnalyzerPlugin({
      openAnalyzer: false,
      analyzerMode: 'static',
      reportFilename: path.join(dropPath, `${lastDirName}.stats.html`),
      generateStatsFile: true,
      statsFilename: path.join(dropPath, `${lastDirName}.stats.json`),
      logLevel: 'error'
    }));

    return generatedConfiguration;
  }
});

build.initialize(gulp);
```

Size Definitions

webpack-bundle-analyzer reports three values for sizes. `defaultSizes` can be used to control which of these is shown by default. The different reported sizes are:

stat

This is the "input" size of your files, before any transformations like minification.

It is called "stat size" because it's obtained from Webpack's `stats object`.

parsed

This is the "output" size of your files. If you're using a Webpack plugin such as Uglify, then this value will reflect the minified size of your code.

gzip

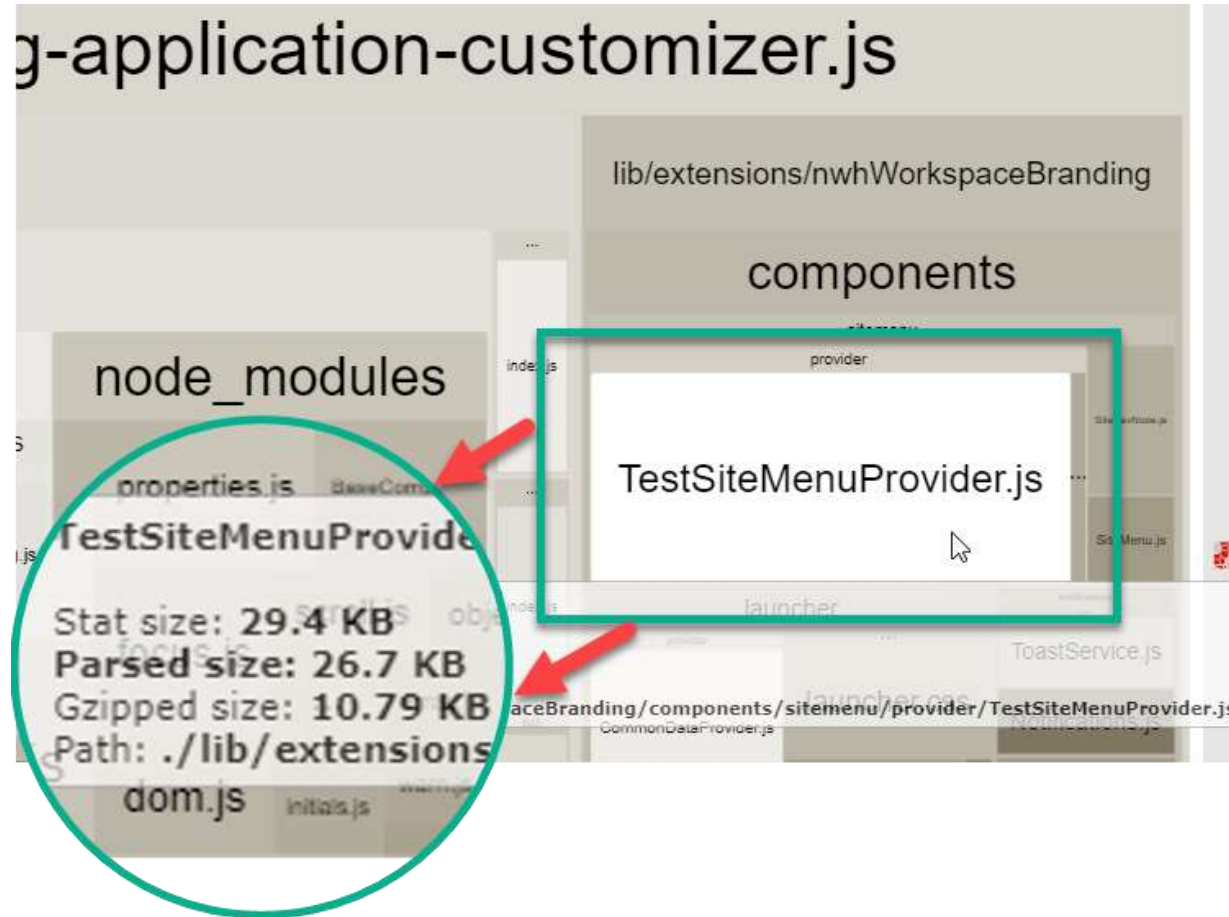
This is the size of running the parsed bundles/modules through gzip compression.



Trimming out Mock Data

- Mock data / services are used to develop and test your web part while using the workbench.
- Browser is unable to call SharePoint locally due to CORS (cross-origin resource sharing)
 - [sp-rest-proxy - SharePoint REST API Proxy for local Front-end development tool-chains](#)
- These mock services could be large depending on the amount of data
 - [Exclude your mock data and other modules from your production bundle in SPFx by Elio Struyf](#)

Mock Provider Size



How to Exclude Mock Data / Services

- Make use of if(DEBUG) to conditionally mock data

```
import ISiteMenuProvider from "../model/ISiteMenuProvider";  
  
// you will need the next two lines in order to get type safety  
import * as test from "../provider/TestSiteMenuProvider";  
let TestSiteMenuProvider: typeof test.default = null;  
if(DEBUG) {  
  TestSiteMenuProvider = require("../provider/TestSiteMenuProvider");  
}
```

•
•
•

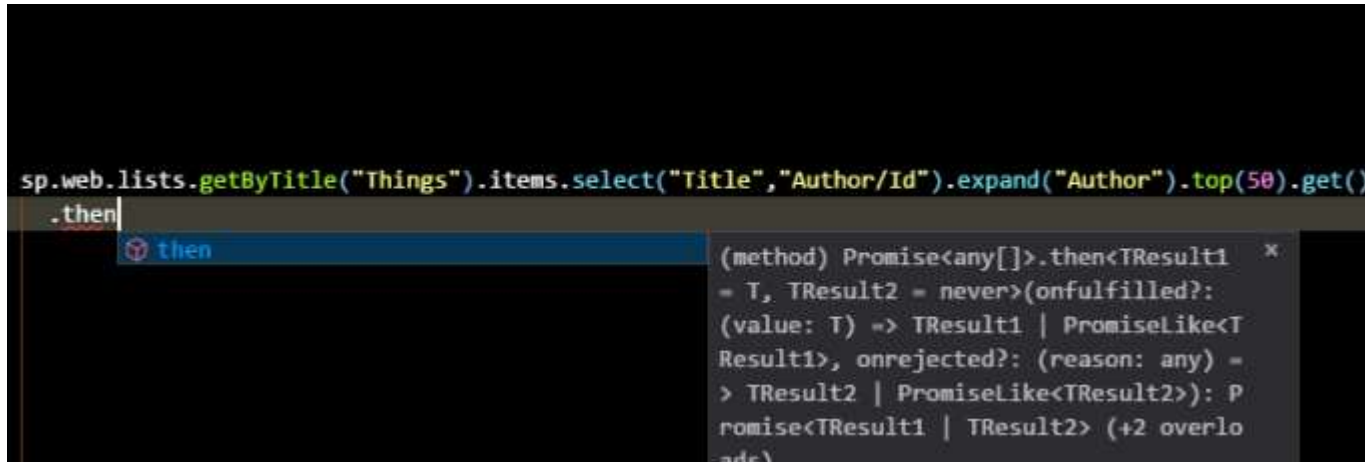
```
if (DEBUG && Environment.type === EnvironmentType.Local) {  
  this.siteNavMenuProvider = new TestSiteMenuProvider();  
} else {  
  this.siteNavMenuProvider = new SiteMenuProvider();  
}
```



Getting Data from SharePoint

SP-PnP-JS

- PnPjs is a collection of fluent libraries for consuming SharePoint, Graph, and Office 365 REST APIs in a type-safe way



```
sp.web.lists.getByTitle("Things").items.select("Title","Author/Id").expand("Author").top(50).get()  
  .then
```

The screenshot shows a code editor with a TypeScript snippet for SharePoint API usage. The code is: `sp.web.lists.getByTitle("Things").items.select("Title","Author/Id").expand("Author").top(50).get().then`. An IntelliSense dropdown is visible under the `.then` property, showing a method signature: `(method) Promise<any[]>.then<TResult1 = T, TResult2 = never>(onfulfilled?: (value: T) => TResult1 | PromiseLike<TResult1>, onrejected?: (reason: any) => TResult2 | PromiseLike<TResult2>): Promise<TResult1 | TResult2> (+2 overloads)`.

- <https://pnp.github.io/pnpjs/>

Benefits of Using SP-PnP-JS

- Intellisense & TypeChecking
- Simplify development experience
- Intuitive using fluent library
- Easier to read code intent
- All the cool kids are using it
- Asynchronous, built on Promises
- Built-in caching
- Abstracts devs from low level details

Remember SPServices?



Simple Example

JSOM

```
function getwebdetails() {  
    var ctx = SP.ClientContext.get_current();  
    oWeb = clientContext.get_web();  
    ctx.load(oWeb, 'Title', 'Description');  
    clientContext.executeQueryAsync(new function() {  
        var iHtml = "<b>Web Title:</b> " + oWeb.get_description() + " ";  
        console.log(strmsg);  
    }, onQueryFailed);  
}  
  
ExecuteOrDelayUntilScriptLoaded(getwebdetails, "sp.js");
```

REST

```
$.ajax({  
    url: _spPageContextInfo.webAbsoluteUrl + "/_api/web?$select=Title,Description",  
    method: "GET",  
    headers: {  
        "Accept": "application/json; odata=verbose"  
    },  
    success: function(data) {  
        console.log("Web: " + data.d.Title + " - Description: " + data.d.Description);  
    }  
});
```

SP-PnP-JS

```
$pnp.sp.web.get().then(function(data) {  
    console.log(data.Title + ' - ' + data.Description);  
});
```



Caching

- Use when data doesn't change quickly / often
- Caching Timeout
- Globally Disable [for testing/debugging]
- Storage Options
 - Session Storage – persists until window closed
 - Local Storage – persists until explicitly deleted

```
pnp.setup({
  defaultCachingStore: "session", // or "local"
  defaultCachingTimeoutSeconds: 30,
  globalCacheDisable: false // or true to disable caching in case of debugging/testing
});

pnp.sp.web.lists.getByTitle("Tasks").items.top(5).orderBy("Modified").usingCaching().get().then(r => {
  console.log(r)
});
```

Common Issues

- Requires Setup for Modern Pages (must establish context) - [guidance](#)
- Disabling Cache – [guidance](#)
 - Delete or clear cache when testing: turn off in code or purge in browser
 - Remember! If a user logs out and then a different users logs on the same machine / same browser cached results would be the same
 - *Beware if you are caching security trimmed results or personal properties*
- Publishing Image Fields – cannot be retrieved with an item in PnP-JS (requires 2 calls) – [guidance \[see paragraph on PublishingRollupImage\]](#)
 - Use JSOM to get everything in 1 call – [example](#)

Optimize Queries

- Regardless of REST, JSOM, or PnP-JS you should always optimize queries
- Optimizations reduce payload size of data transfers
- Use row limit or top to reduce the number of returned results
- Specify only the fields which you need back
- Use where or filters where appropriate

REST

```
var queryUrl = _spPageContextInfo.webServerRelativeUrl +
  "_api/web/lists/getbytitle('Homepage Slideshow')/items?$select=FileRef,SlideLinkUrl,Title&
  $filter=(SlideActive eq 1)&$orderby=SlideOrder";

$.ajax({
  headers: {
```

SP-PnP-JS

```
return pnp.sp.web.lists
  .getByTitle("Global Nav List")
  .items.select(
    "Title",
    "GlobalNavUrl",
    "GlobalNavLocation",
    "GlobalNavOpenInNewWindow",
    "GlobalNavParent/Title"
  )
  .filter("GlobalNavLocation eq '" + location + "'")
  .expand("GlobalNavParent")
  .orderBy("GlobalNavOrder")
  .usingCaching()
  .get()
  .then(
    (items: ISPGlobalNavItem[]): IGlobalNavItem[] => {
```

JSOM

```
let query = `<View>
  <Query>
    <Where>
      <And>
        <Eq><FieldRef Name='${Constants.ItemActiveFieldName}' /><Value Type='Tr
        <Eq><FieldRef Name='${Constants.LauncherKeyFieldName}' /><Value Type='I
      </And>
      <Eq><FieldRef Name='${Constants.ContentTypeFieldName}' /><Value Type='I
    </And>
  </Where>
  <OrderBy>
    <FieldRef Name='${Constants.ItemOrderFieldName}' />
  </OrderBy>
</Query>
<ViewFields>
  <FieldRef Name='${Constants.IdFieldName}' />
  <FieldRef Name='${Constants.TitleFieldName}' />
  <FieldRef Name='${Constants.BackgroundColorFieldName}' />
  <FieldRef Name='${Constants.IconFieldName}' />
  <FieldRef Name='${Constants.UrlFieldName}' />
  <FieldRef Name='${Constants.ContentTypeFieldName}' />
  <FieldRef Name='${Constants.OpenInNewWindowFieldName}' />
</ViewFields>
<RowLimit>${maxItems}</RowLimit>
</View>`;
```

Miscellaneous & Somewhat Random Tips

UI Frameworks

- Recommended to use Office UI Fabric
 - Core - Grids / Icons / Animations
 - Components – Loaders / Pickers / Inputs / Surfaces
- Office UI Fabric current version is greater than O365 deployed version
- Eventually the dev path of Office UI Fabric will converge w/ Office 365
- Other framework's may not work with Office 365, leaving you stuck

Office UI React Guidance

- It's recommended that you use the versions of the Office UI Fabric React package included in the project in the SharePoint Framework's Yeoman generator.
- For instance, the SharePoint Framework v1.7.0 release uses Fabric React v5.131.0
- [Using Office UI Fabric Core and Fabric React in SharePoint Framework](#)

jQuery

- Highly recommend NOT using jQuery
- React is unaware of changes made to the DOM outside of React
- There are other ways to get items from SP other than \$.ajax or \$.get
- If you absolutely must use it these are things to consider:
 - Be mindful of what you are doing to prevent React from updating
 - jQuery registers globally as \$
 - Multiple webparts loading different jQuery versions can be very bad
- Use references to react components
 - Integrating with Other Libraries

TypeScript – Typing Specifically

- You are already using TypeScript
- It's your job to type your objects, interfaces, variables
- By typing objects
 - Reduces defects
 - Easier to add new features / enhance code
 - Clearly defines inputs & outputs

TypeScript – Typing Specifically (cont'd)

- Typing inputs & outputs

```
public getFeaturedImage(key: string): Promise<IFeaturedImage> {  
    let query = `<View>
```

- Typing variables

```
let myString: string = "hello world";  
let myNumber: number = 5;  
let myObject: object = {};  
let myCustomObject: IVideo;  
let myAny: any;
```

The goal is for **everything** to have a type!

.... Try not to use **any**

Of course there are some exceptions

- It's not an object from your library
- You don't have typedefs available

[TypeScript Do's and Don'ts](#)

TypeScript – Typing Specifically (cont'd)

- Creating interfaces

```
export default interface IVideo {  
  key: string;  
  title: string;  
  videoUrl: string;  
  videoThumbnail: string;  
  body: string;  
}
```

```
let myCustomObject: IVideo = {  
  key: "x",  
  title: "y",  
  videoUrl: "z",  
  videoThumbnail: 3,  
  body: "b"  
};
```

[ts] Type 'number' is not assignable to type 'string'.
• IVideo.d.ts(5, 5): The expected type comes from property 'videoThumbnail' which is declared here on type 'IVideo'
(property) IVideo.videoThumbnail: string

different variable type

```
let myCustomObject: IVideo;  
myCustomObject = {  
  key: "x",  
  title: "y",  
  videoUrl: "z",  
  videoThumbnail: "a",  
  body: "b"  
};
```


fits the
definition

```
let myCustomObject: IVideo;  
myCustomObject = {  
  key: "x",  
  title: "y",  
  videoUrl: "z",  
  videoThumbnail: "a"  
};
```

[ts]
Type '{ key: string; title: string; videoUrl: string; videoThumbnail: string; }' is not assignable to type 'IVideo'.
Property 'body' is missing in type '{ key: string; title: string; videoUrl: string; videoThumbnail: string; }'.
let myCustomObject: IVideo
myCustomObject = {
 key: "x",
 title: "y",
 videoUrl: "z",
 videoThumbnail: "a",
};

missing parameter

Productivity & Plugins

- Use a code repository to work on the same project
- Chrome Debug Tools
 - Cache Killer Classic – always kills cache / no worries about ctrl-f5 
 - React Developer Tools – inspect React Components – State & Properties
 - SP Insider – inspect anything in O365 & Generate REST / Search Queries !
 - Scratch JS – write JS in the browser for quick testing
 - SP-Editor – PnP JS console built in – quickly write / test queries
- Visual Studio Code Plugins
 - [GitLens](#)
 - [Rencore Deploy SPFx Package](#)
 - [Prettier](#) – code formatter
 - XML – code formatter
- Stick to the same coding standards
 - [Airbnb React/JSX Style Guide](#)
- Use versioning on projects
 - [How to version new SharePoint Framework projects](#)
 - [SPFx Automatically Generating Revision Numbers + Versioning](#)

Build / Bundle / Package Script

- Commands to execute before pushing to live SharePoint env.

build-prod.cmd

```
1 cls
2
3 call gulp clean
4
5 call gulp build --ship
6
7 call gulp bundle --ship
8
9 call gulp package-solution --ship
10
11 call explorer .\sharepoint\solution\
```

[Simple Build Script for the SharePoint Framework by Thomas Daly](#)

Debugging & Testing

- Use Chrome Dev tools or FireFox, far superior than IE (not even a contest)
 - Blackbox native SP scripts
 - Disable cache
- The local workbench can increase the speed of your development
 - It's a good practice to mock your data to simulate interactive w/ SP
- Test Live using a real sites workbench
 - {siteUrl}/_layouts/15/workbench.aspx
- Test Live on your site, append this to your URL
 - Solution must be deployed and added to the site first
 - ?debugManifestsFile=https://localhost:4321/temp/manifests.js&loadSPFX=true

Choose your primary client-side library

- Consider multiple components on same page, or across different pages
- Using same library loaded from same URL, web browser will use cached results → portal loads faster
- Rationalize which libraries and versions AND where you load from
 - Not just project specific but for the whole organization

Versions in Reality



WP1 CODE

V2.22.0

PnP V3.0.9

jQuery V3.3.1

SPFx V1.5.1

Office UI Fabric V9.5.0

V2.4.0



WP2 CODE

V2.22.2

PnP V3.0.10

SPFx V1.6.0

Office UI Fabric V9.6.0



WP1 CODE WP2 CODE

jQuery V3.3.1 V2.4.0

V2.22.0 V2.22.2

PnP V3.0.9 PnP V3.0.10

Office UI Fabric V9.5.0 Office UI Fabric V9.6.0

Reference only the necessary components

- When working with external libraries, you might actually not need the whole library
- Take Lodash as example:
 - Reference entire library = 527KB (unoptimized)

```
import * as _ from 'lodash';
```

- Reference specific method = 45KB (unoptimized)

```
const at: any = require('lodash/at');
```

Check SPFx First!

- Lodash - @sp-lodash-subset
- SPHttpClient, SPHttpClientConfiguration - @microsoft/sp-http
- Logging - @microsoft/sp-core-library
 - [Logging in SharePoint Framework solutions](#)
- [@microsoft/sp-page-context](#) – [SPUser, SPWeb, SPSite]
- SPFx includes some versions of their own polyfills
- Polyfills mainly Map and Proxy. – IE 11 can be your worst enemy
- Importing ES6-Shim or Core.js/ES6 Shims for ES6 library support in SPFX [babel-polyfill]
 - [SPFX Polyfill Out of Stack Space exception](#)

Reusable Components & Examples

- [Reusable React controls for your SharePoint Framework solutions](#)
- [Reusable property pane controls for the SharePoint Framework solutions](#)
- [SharePoint Framework Client-Side Web Part Samples & Tutorial Materials](#)
- [Office Fabric UI Components](#)
- [SPFx Fantastic 40 Web Parts](#)

Packages.json

- dependencies
 - Modules required during runtime
 - Packages you would bundle
- devDependencies
 - Modules only required during development
 - Packages you won't bundle
 - Packages you plan to externalize
 - Packages only required to build
- Keep it to the packages you are using, remove anything else

```
"dependencies": {
  "@microsoft/decorators": "~1.4.1",
  "@microsoft/sp-application-base": "~1.4.1",
  "@microsoft/sp-core-library": "^1.4.1",
  "@microsoft/sp-dialog": "~1.4.1",
  "@microsoft/sp-lodash-subset": "^1.4.1",
  "@types/es6-promise": "0.0.33",
  "@types/webpack-env": "1.13.1",
  "office-ui-fabric-react": "^5.120.0",
  "overlayscrollbars": "^1.5.0",
  "react-resize-aware": "^2.7.1"
},
"devDependencies": {
  "@microsoft/sp-build-web": "~1.4.1",
  "@microsoft/sp-module-interfaces": "~1.4.1",
  "@microsoft/sp-webpart-workbench": "~1.4.1",
  "@types/chai": ">=3.4.34 <3.6.0",
  "@types/microsoft-ajax": "0.0.33",
  "@types/mocha": ">=2.2.33 <2.6.0",
  "You,
```

Packages Checker

- npm-check
- Checks packages.json for unused dependencies
- Has interactive update functionality

```
PS C:\_nw\dev\NWH-TeamsCard> npm-check

@microsoft/sp-lodash-subset  😞 NOTUSED? Still using @microsoft/sp-lodash-subset?
Depcheck did not find code similar to require('@microsoft/sp-lodash-subset') or
require('microsoft/sp-lodash-subset').
Check your code before removing as depcheck isn't able to foresee all ways depend
ed.
Use --skip-unused to skip this check.
To remove this package: npm uninstall --save @microsoft/sp-lodash-subset

@microsoft/sp-office-ui-fabric-core  😞 NOTUSED? Still using @microsoft/sp-office-ui-fabric-core?
Depcheck did not find code similar to require('@microsoft/sp-office-ui-fabric-core') or
require('microsoft/sp-office-ui-fabric-core').
Check your code before removing as depcheck isn't able to foresee all ways depend
ed.
Use --skip-unused to skip this check.
To remove this package: npm uninstall --save @microsoft/sp-office-ui-fabric-core

@types/react  😊 MAJOR UP Major update available. https://github.com/DefinitelyTyped/DefinitelyTyped
npm install --save @types/react@16.4.18 to go from 15.6.6 to 16.4.18

@types/react-dom  😊 MAJOR UP Major update available. https://github.com/DefinitelyTyped/DefinitelyTyped
npm install --save @types/react-dom@16.0.9 to go from 15.5.6 to 16.0.9

@types/webpack-env  😊 PATCH UP Patch update available. http://www.github.com/DefinitelyTyped/DefinitelyTyped
npm install --save @types/webpack-env@1.13.6 to go from 1.13.1 to 1.13.6

react  😊 MAJOR UP Major update available. https://reactjs.org/
npm install --save react@16.6.0 to go from 15.6.2 to 16.6.0
```


Warnings & Considerations

- SPFX 1.7 is now using React 16
- SPFX 1.7 still using TypeScript 2.4.2
 - Some npm libraries won't compile with that
- IE 11 is not your friend
 - [Check for browser supportability: Caniuse.com](https://caniuse.com)
- Watch out for npm packages that use polyfills that collide w/ SPFx
- Read Release Notes when new versions of SPFx are released
- Check [SPFx github when you encounter issues – Report Bugs](#)

Staying Current

- Community Calls

Community call	When	Description	Skype
Monthly community call	Second Tuesday of each month at 8:00 AM PT / 4:00 PM GMT	Monthly community call covering the latest changes in SharePoint development-related topics within the last month, including news, UserVoice updates, and community contributions	Direct Skype link to meeting
Special interest group call for SharePoint Framework	Bi-weekly on Thursdays at 7:00 AM PT / 3:00 PM GMT	SharePoint Engineering updates, SharePoint Framework, PnPJS, Office 365 CLI, and reusable SPFx controls	Direct Skype link to meeting
Special interest group call for general SharePoint development	Bi-weekly on Thursdays at 7:00 AM PT / 3:00 PM GMT	SharePoint Engineering updates, end-to-end solution designs, provisioning, PnP CSOM, and PnP PowerShell	Direct Skype link to meeting

- <https://docs.microsoft.com/en-us/sharepoint/dev/community/community>



Slides, Demos & Contact

- <https://www.slideshare.net/tommdaly>

Email: thomasd@sohodragon.com

Twitter: [@ tomdaly](https://twitter.com/tomdaly)

LinkedIn: [profile](#)