SharePoint CONFERENCE NORTH AMERICA

SharePoint CONFERENCE NORTH AMERICA

Exploring the SharePoint Framework

A SharePoint development model for now and the future

Eric Overfield

Eric Overfield

President, PixelMill
Microsoft Regional Director
Microsoft MVP – Office Servers and Services

@ericoverfield
ericoverfield.com

Top priorities

Why a new SharePoint development model

SharePoint Framework fundamentals

SPFx webparts and extensions

Packaging and deploying

Best practices and advise

A New Development Model?

SharePoint development through the ages

Full trust solutions

SharePoint 2007+
No cloud options

Sandbox solutions

SharePoint 2010+
Cloud gaps

Add-in model

SharePoint 2013+
First solid cloud offering

Keeps changing

Every three to four years, a new development model

SharePoint Framework – here to stay

It's different this time*

Microsoft has bet their future on SPFx

Same model 1st -> 3rd party

SharePoint engineering using same model

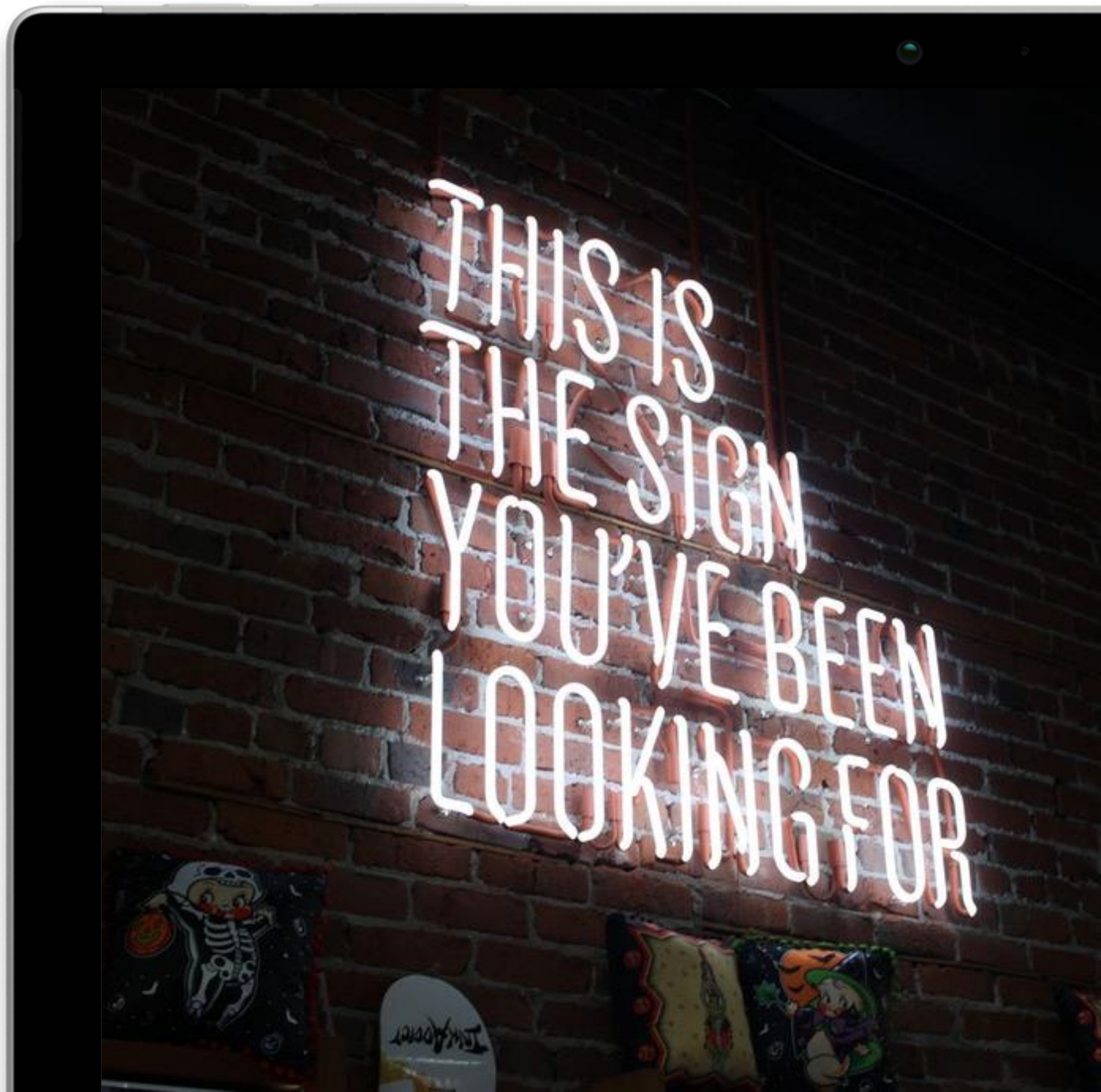
OOTB Modern experience uses SPFx

Equal footing with equal access

Capabilities

Extension points

Access to Microsoft API's



SPFx - Worth the investment

SPFx Fundamentals

SharePoint Framework basics

Native SharePoint development model based on client-side rendering

Currently includes webparts and extensions

Open source tooling / toolchain

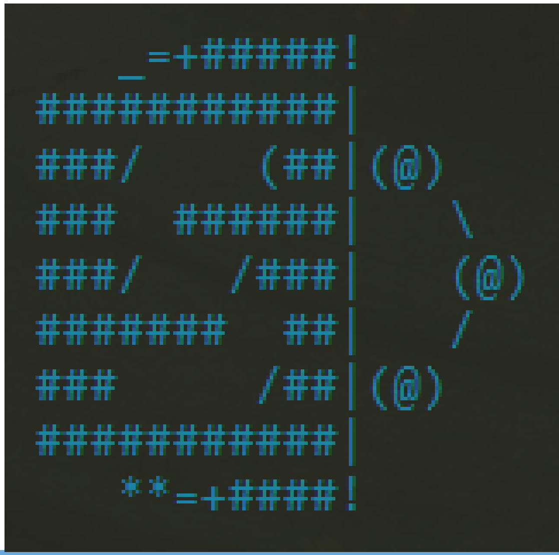
nodeJS, npm, Yeoman, gulp, TypeScript, webpack, and more

Integrates with internal and external data

Wrapper REST API classes for SharePoint, Microsoft Graph, and external data

SharePoint Online, SharePoint 2016+

Modern experiences and extensions expected in SharePoint 2019



```
   _=+#####!
###########|
###/    (##|(@)
###  ######|   \
###/   /###|   (@)
#######  ##|   /
###     /##|(@)
###########|
   **=+####!
```

Client-side rendering and SharePoint

Client-side rendering first class citizen

No server side / compiled code / C#

IDE / development platform agnostic

Enabled by open source toolchain

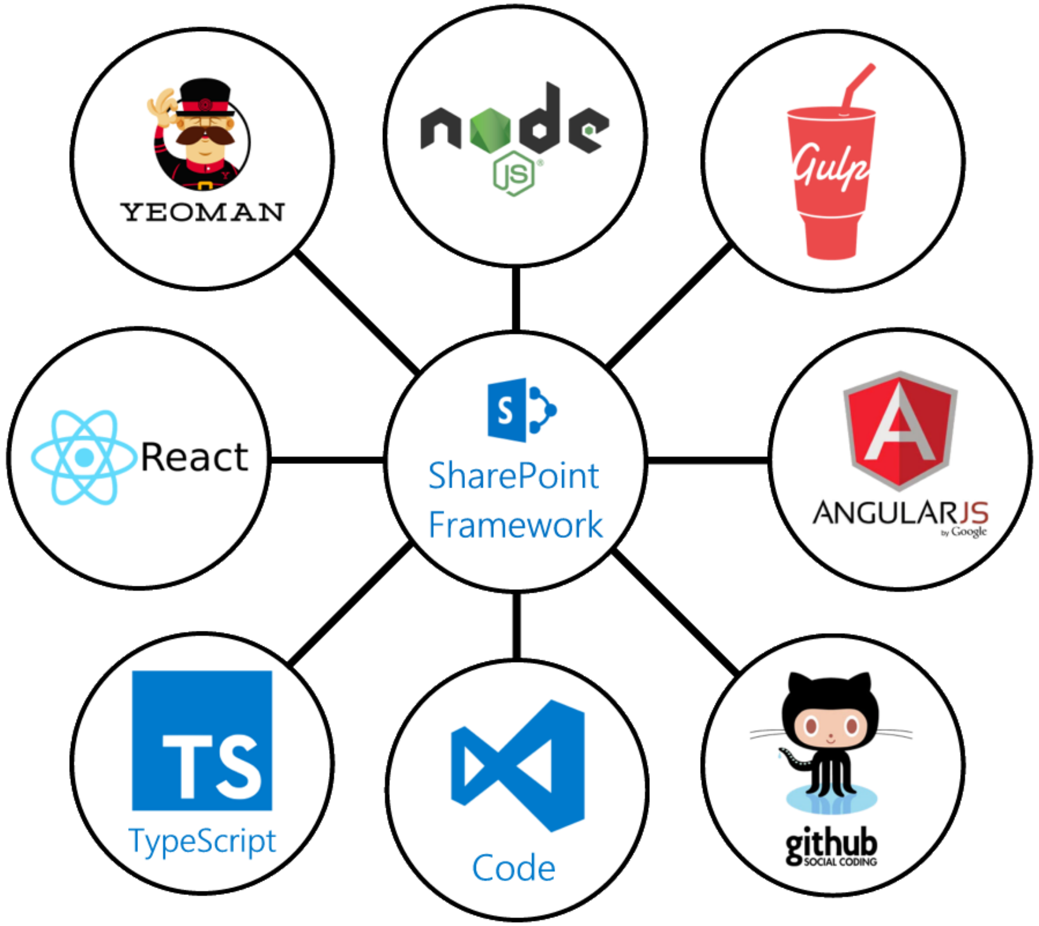
Visual Studio not required

Not dependent on JavaScript Injection

Still using JavaScript / TypeScript, now with clean integration

No iFrame requirement

Direct integration of custom code within the page model



Let’s get started

Read the documentation

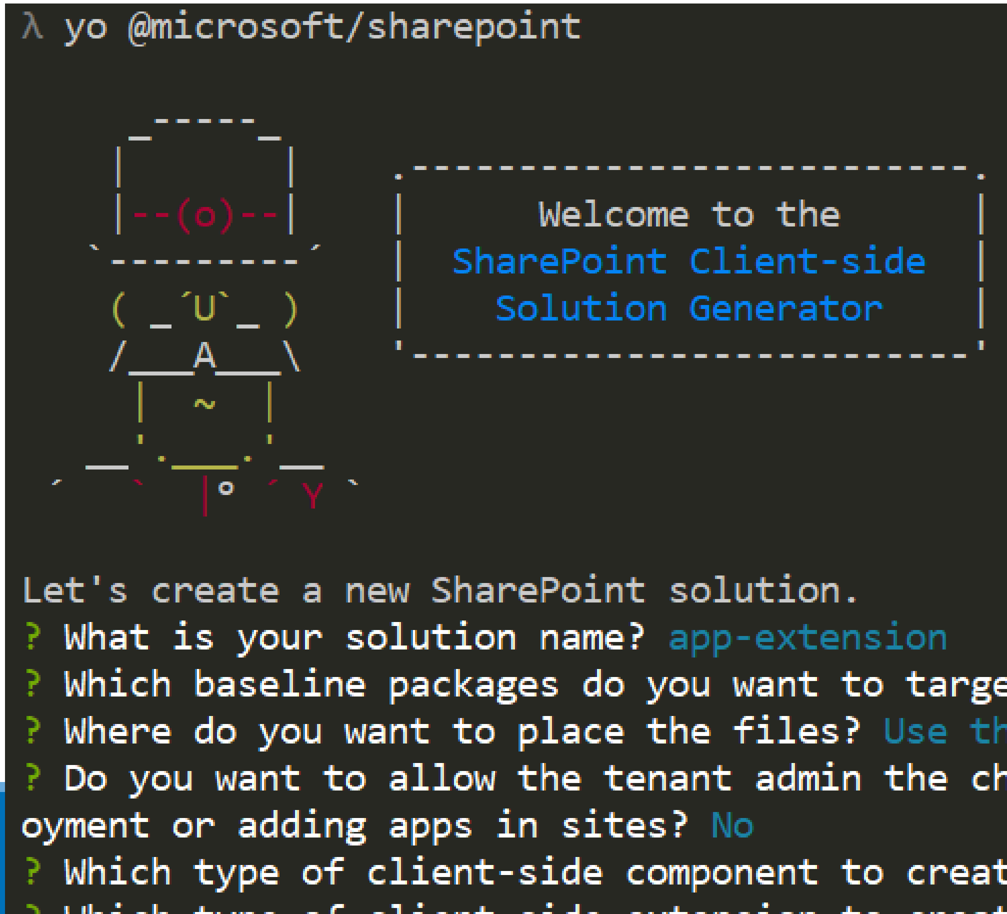
https://docs.microsoft.com/en-us/sharepoint/dev/spfx/set-up-your-developer-tenant

Install toolchain

1. Install nodeJS (current Long Term Support (LTS) version
2. Install Yeoman and gulp
```
c:\> npm i -g yo gulp
```
3. Install Yeoman generator
```
c:\> npm i -g @microsoft/generator-sharepoint
```

Create your first SPFx solution

```
c:\> yo @microsoft/sharepoint
gulp serve
```



```
λ yo @microsoft/sharepoint

       _-----_
      |       |    .--------------------------.
      |--(o)--|    |      Welcome to the      |
     `---------´   | SharePoint Client-side   |
      ( _´U`_ )    |   Solution Generator     |
      /___A___\    '--------------------------'
       |  ~  |
     __'.___.'__
   ´   `  |° ´ Y `

Let's create a new SharePoint solution.
? What is your solution name? app-extension
? Which baseline packages do you want to targe
? Where do you want to place the files? Use th
? Do you want to allow the tenant admin the ch
oyment or adding apps in sites? No
? Which type of client-side component to creat
```

Demo: Spinning up SPFx

SPFx Solution Structure

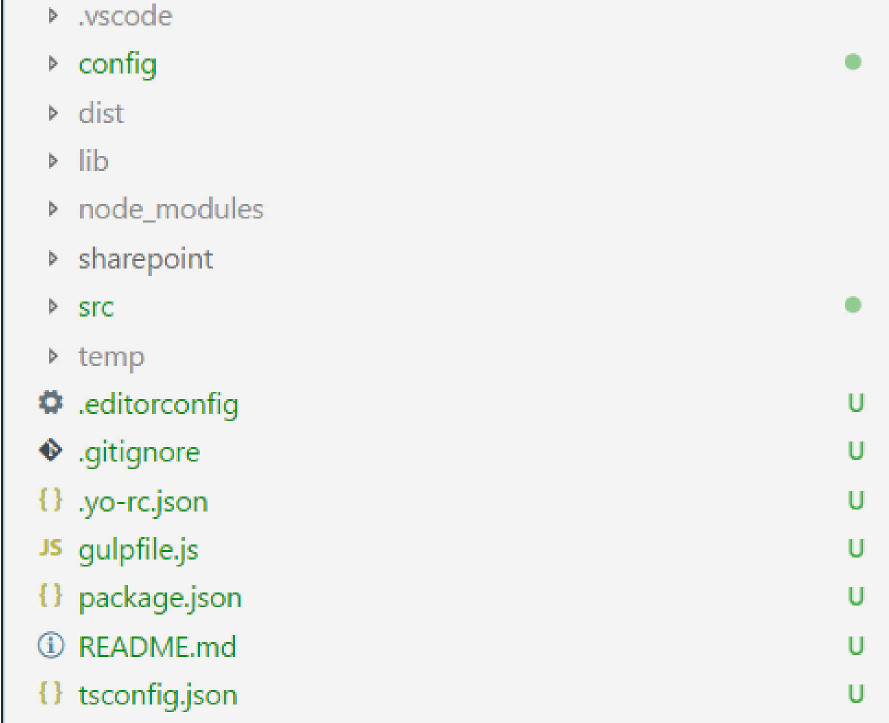
The SPFx scaffolding

Getting to know SPFx folders

./: solution description and build settings
src: primary webpart TypeScript source code
config: json configuration files for webpart and build process

sharepoint: .sppkg file for App Catalog

dist: web ready code for distribution
lib: build files (TS compiled JS) ready for bundle
temp: temp workspace for building
temp/deploy: RELEASE bundle assets
node_modules: NodeJS modules (JS) for toolchain



```
▸ .vscode
▸ config                ●
▸ dist
▸ lib
▸ node_modules
▸ sharepoint
▸ src                   ●
▸ temp
  .editorconfig         U
  .gitignore            U
  .yo-rc.json           U
  gulpfile.js           U
  package.json          U
  README.md             U
  tsconfig.json         U
```

Demo: Breaking Down SPFx

Two types of SPFx Components:

Webparts - Extensions

SPFx webparts and extensions

Webparts

Components added to a specific page

Repeatable and customizable

Added within flow of content

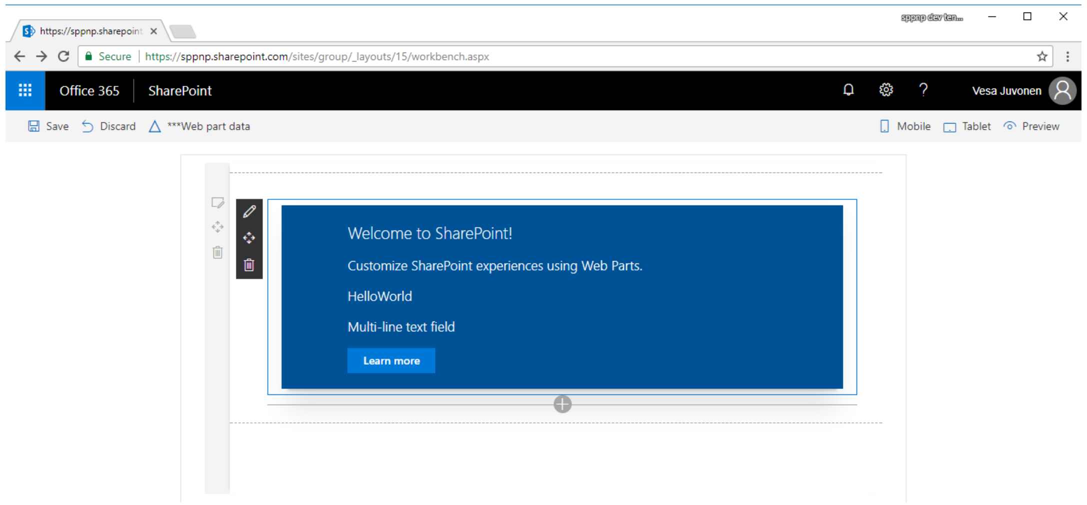
Extensions

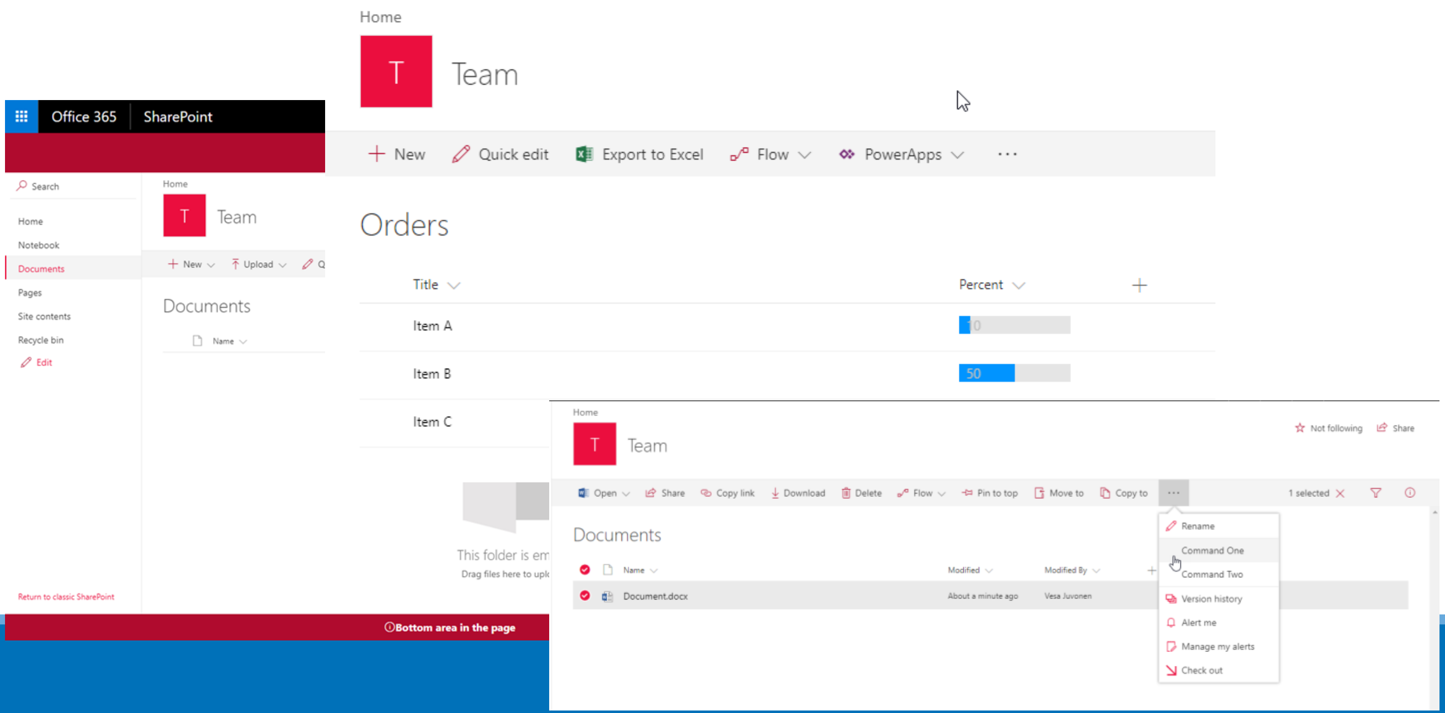
General components for pages/lists/libraries

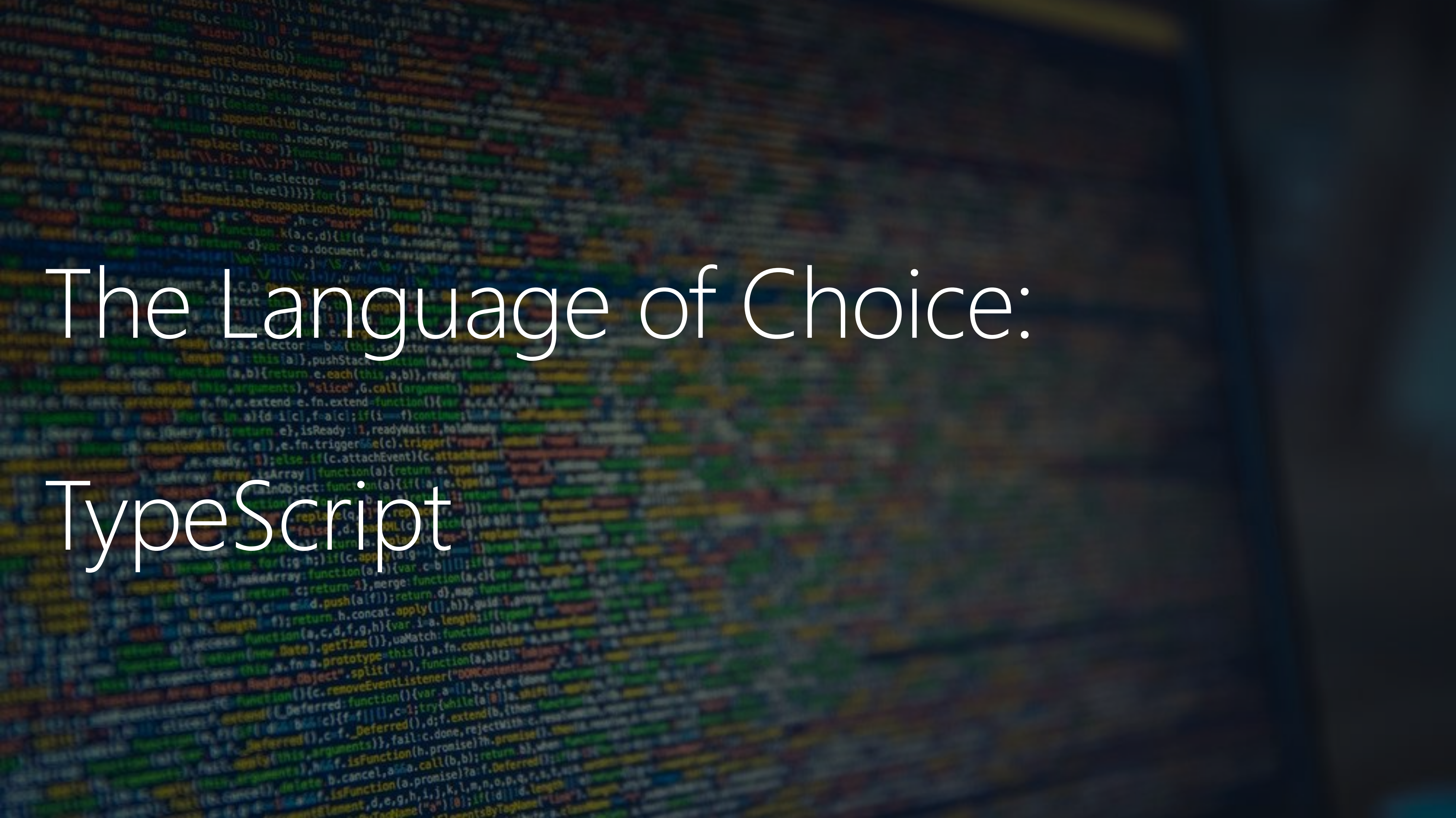
Application Customizers

Field Customizers

ListView Command Set Customizers







The Language of Choice:

TypeScript

TypeScript – Our new JavaScript

"TypeScript is a free and open source programming language developed and maintained by Microsoft. It is a strict superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language."
https://www.typescriptlang.org

TypeScript

Strict superset of JavaScript

Adds optional static typing

Class based object-oriented language

Transpiles to JavaScript

Creating a User Experience

The SPFx Way

Choose your UI framework

Native SPFx rendering options

No framework – you create the HTML
React – Microsoft recommended approach
Knockout

Other options

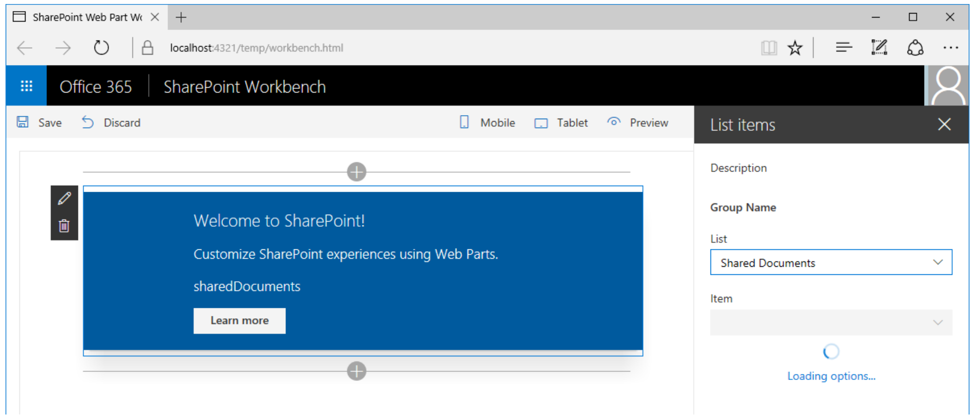
AngularJS / Angular – large external community
jQuery – a shim / helper
Handlebars
Office UI Fabric – used in conjunction with React
Any framework / library

Demo: SPFx Webparts and Extensions

SPFx Properties and Data

Allow webpart customization
Connect to SharePoint data and Microsoft Graph

Webpart properties



Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

sharedDocuments

List items

Description

Group Name

List: Shared Documents

Item

Loading options...

SPFx webparts can include custom properties

Define: /src/webparts/"webpart"/"webpart"WebPart.ts
add in: `I"webpart"WebPartProps` interface

Default values: /src/webparts/"webpart"/"webpart"WebPart.manifest.json
add in json: `preconfiguredEntries.properties`

Display: /src/webparts/"webpart"/"webpart"WebPart.ts
method: `protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {}`

Override onChange: /src/webparts/"webpart"/"webpart"WebPart.ts
method: `public onPropertyPaneFieldChanged (propertyPath: string, oldValue: any , newValue: any) {}`

Connecting to data

Connect to internal and external data with built in methods

TypeScript Http classes within `@microsoft/sp-http`

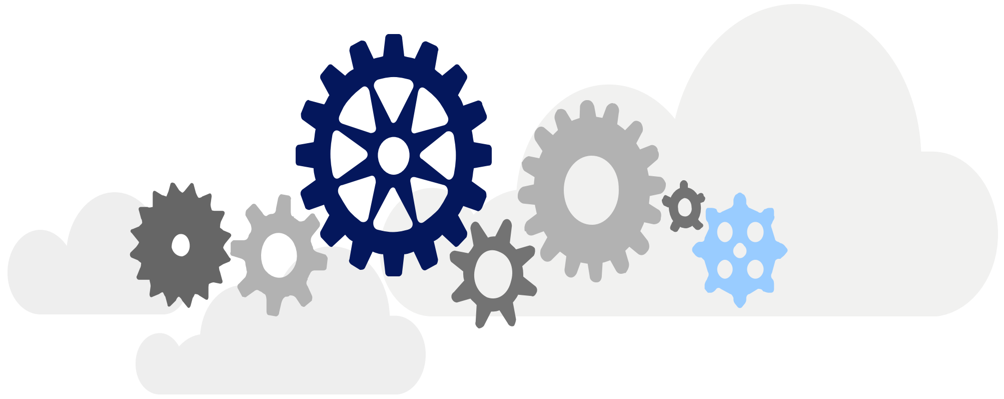
`this.context` always includes `spHttpClient`!

HttpClient

Basic set of features for REST operations with any external resource

SPHttpClient

REST calls against SharePoint, handles context, security, etc.



AadHttpClient / MSGraphClient

Make calls to Azure Active Directory Applications

MSGraphClient currently in preview - `@microsoft/sp-client-preview` – make calls to Microsoft Graph

Packaging and Deploying

Enable solution utilization

Package and deploy SPFx solutions to SharePoint

After debugging, bundle then package solution for deployment

`C:\> gulp bundle` (creates the solution bundles)

`C:\> gulp package-solution` (creates /sharepoint/solution/"webpart".sppkg)

Use --ship switch to minify bundle and reads in CDN info from config/write-manifests.json

Deploying to SharePoint

May be deployed to tenant app catalog or to specific site collections

Site collection app catalogs created via SPO Management Shell

Packaging and deployment considerations

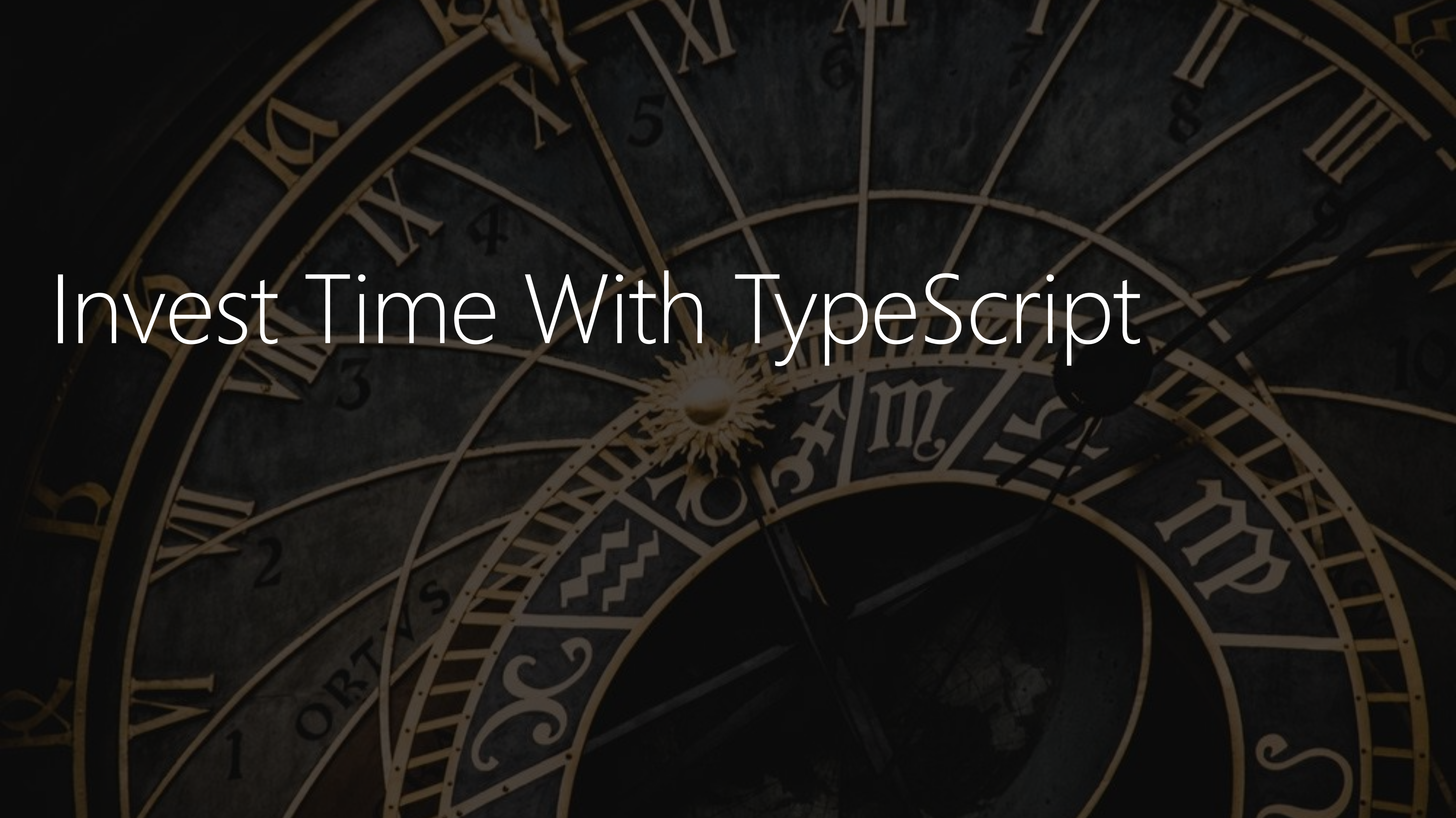
Webparts and extensions may be activated automatically across tenant

Microsoft Graph access confirmed during deployment (preview)

Get the Most From SPFx

Tips and Tricks

Learn toolchain



Invest Time With TypeScript

Utilize React

Accept Change

SPFx v1.5 already announced early May 2018
Watch your ALM

Leverage the Community!

https://docs.microsoft.com/sharepoint
https://github.com/SharePoint/sp-dev-fx-webparts
https://github.com/SharePoint/sp-dev-fx-extensions
https://github.com/SharePoint/sp-dev-solutions

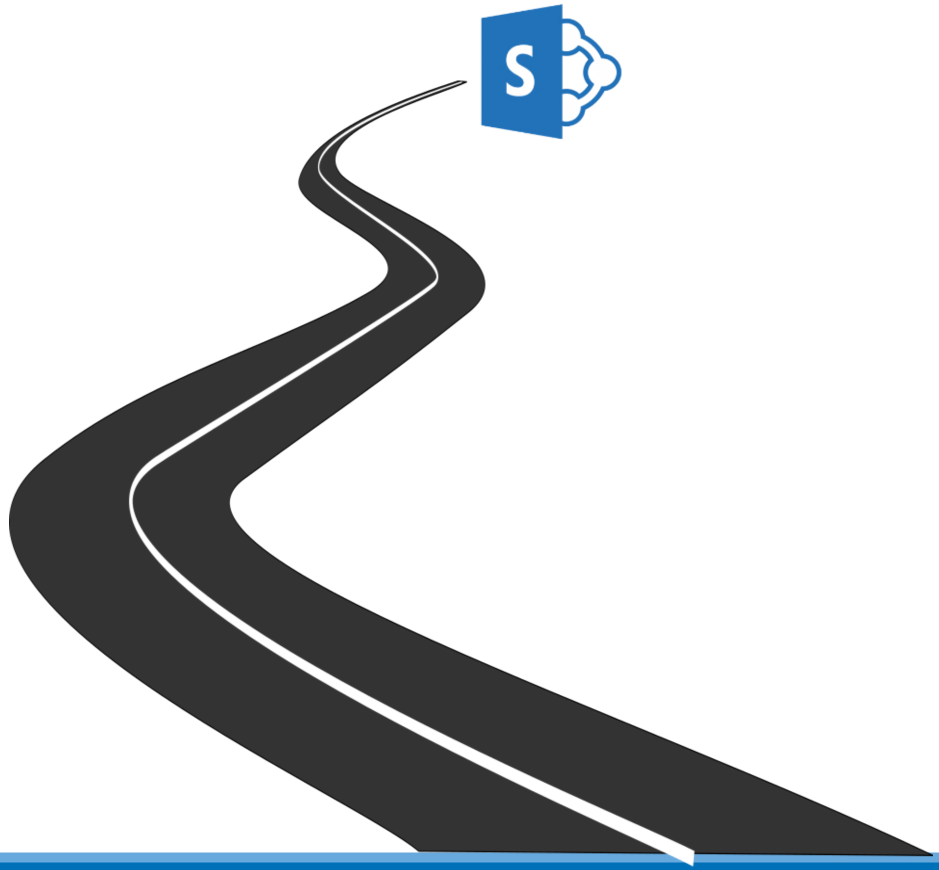
SPFx Roadmap CY 2018

Spring 2018

Asset packaging GA
Site Collection App Catalog GA
Native Graph access from SharePoint Framework preview
API and dev support for many OOTB features

Summer / Fall

Native Graph access from SPFx GA
Enterprise solutions for SPFx to Teams
Global deployment for SPFx
Dynamic data in SPFx components
SPFx in SharePoint 2019 - including modern experiences



Thank You

Eric Overfield

@ericoverfield

ericoverfield.com

Join me Wednesday 2pm

Advanced SharePoint Framework Webpart Strategies

docs.microsoft.com/sharepoint