SharePoint Framework (SPFx)

Agenda

SharePoint Framework (SPFx) Overview

Node.js

Webpack

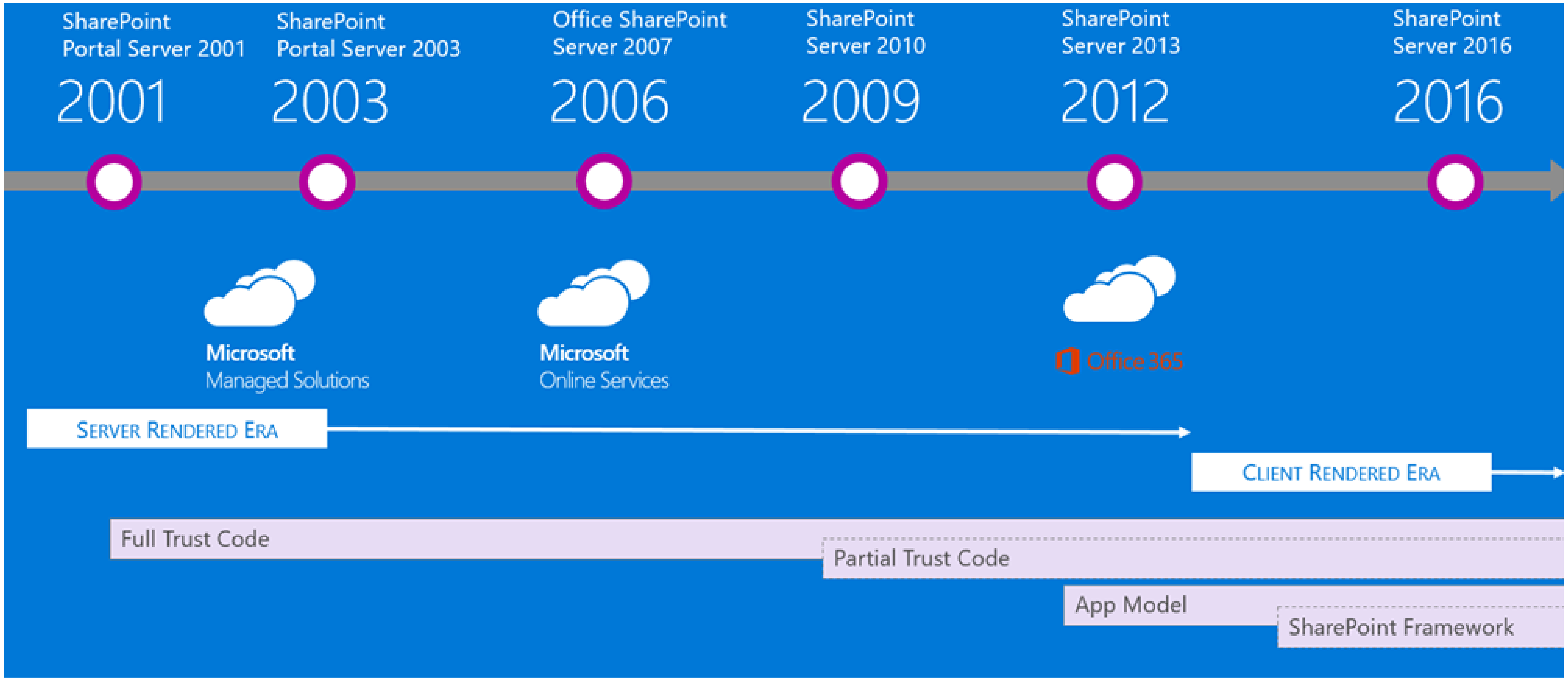
SPFx Basics

SPFx CRUD

SPFx & JavaScript Libs

SharePoint Framework Overview

Evolution of SharePoint Customizations



What is SharePoint Framework (SPFx)

Version 1.0 globally available since Q1 / 2017

Modern client-side development – Responsive by default -Empowers classic web dev skills

Based open source tools and JavaScript web frameworks

Documentation @ https://sharepoint.github.io/

Samples @ https://github.com/SharePoint/sp-dev-docs/wiki

API Reference @ https://docs.microsoft.com/en-us/javascript/api/sp-application-base?view=sp-typescript-latest

SPFx Features

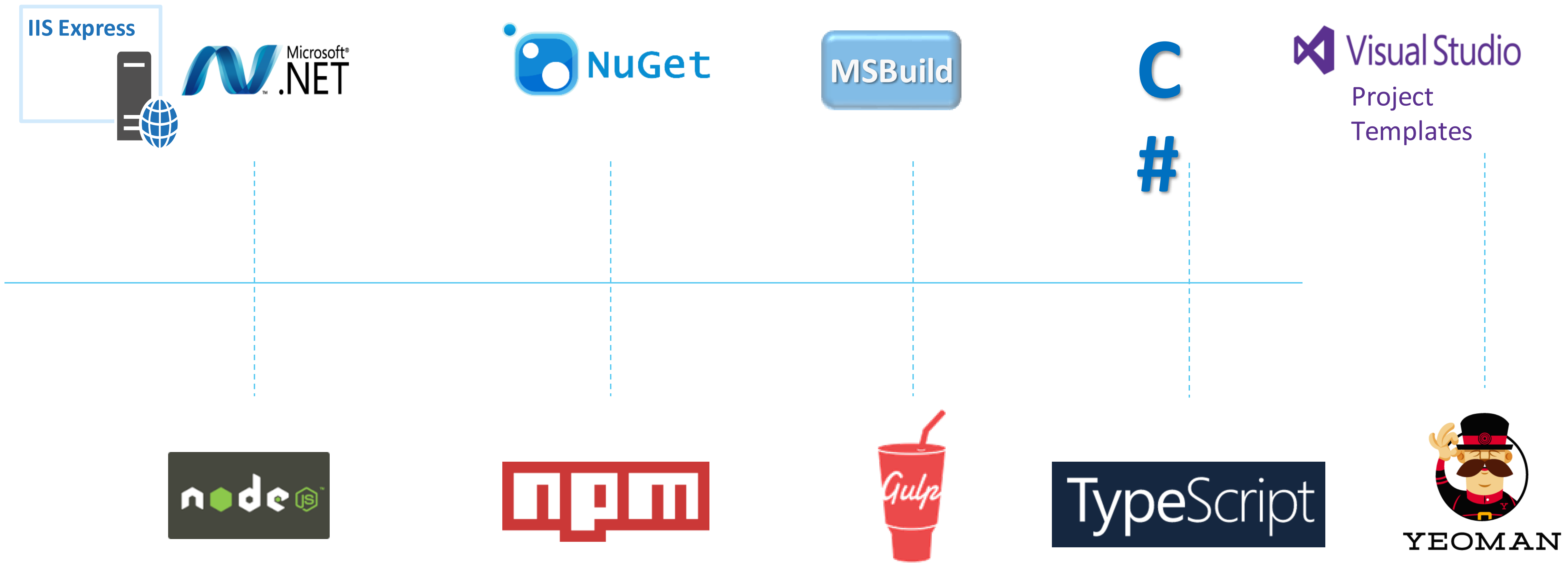
WebParts

- React -> Office UI Fabric is written in React
- Angular Hosting using Angular Elements und upcoming ng 9 Standalone Components
- Javascript, Vue.js, ...

Extensions

- App Customizer -> Global Nav or Script Injection
- Field Customizer -> Client Side Field Renderer, Replaces JS Link
- List View Command -> Replaces Custom Action

Server Side vs Client Side Tooling

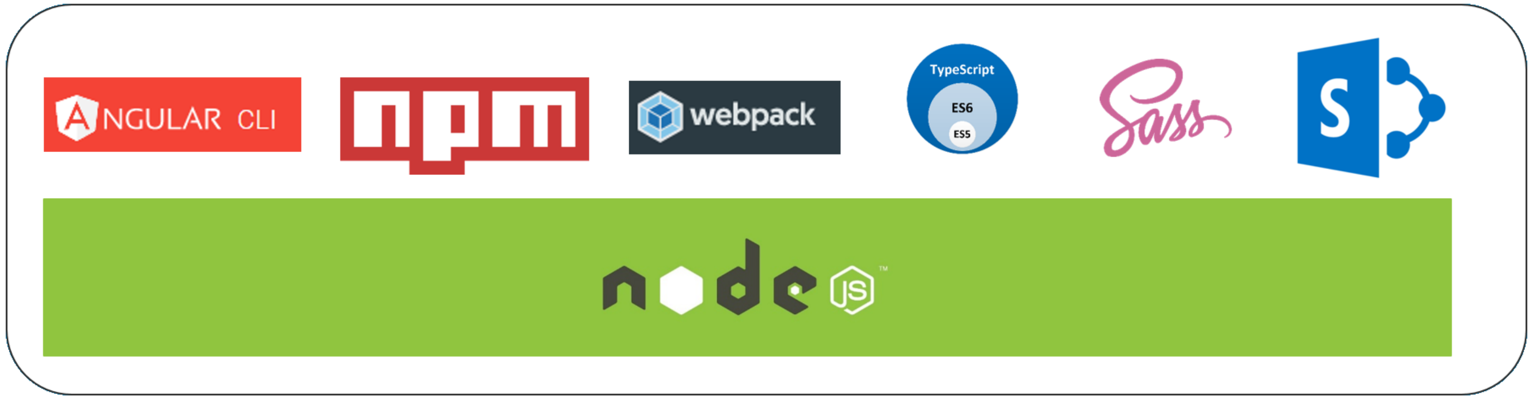


Nodes Role in modern Web Stack Dev

Node.js accts as a Runtime Host for our Dev Toolset

It provides:

- Package Management
- TS /Sass Compilation
- Bundling
- ...



Gulp

Gulp can be used to automate

- Bundling, minification or
- the cleansing of a development environment

Requires gulpfile.js

Used to „compile“ and package

- Gulp serve
- Gulp package

```
{
    "name": "ASP.NET",
    "version": "0.0.0",
    "devDependencies": {
      "gulp": "3.8.11",
      "gulp-concat": "2.5.2",
      "gulp-cssmin": "0.1.7",
      "gulp-uglify": "1.2.0",
      "rimraf": "2.2.8"
    }
  }
```

Yeoman

Yeoman helps you to kickstart new projects, prescribing best practices and tools

Based on Node.js, Yeoman uses Generators, to set up different technologies

You cann add your own generators

Using Yeoman & ASP.NET Core: https://docs.asp.net/en/latest/client-side/yeoman.html

Documentation at http://yeoman.io/

Steps to use Yeoman

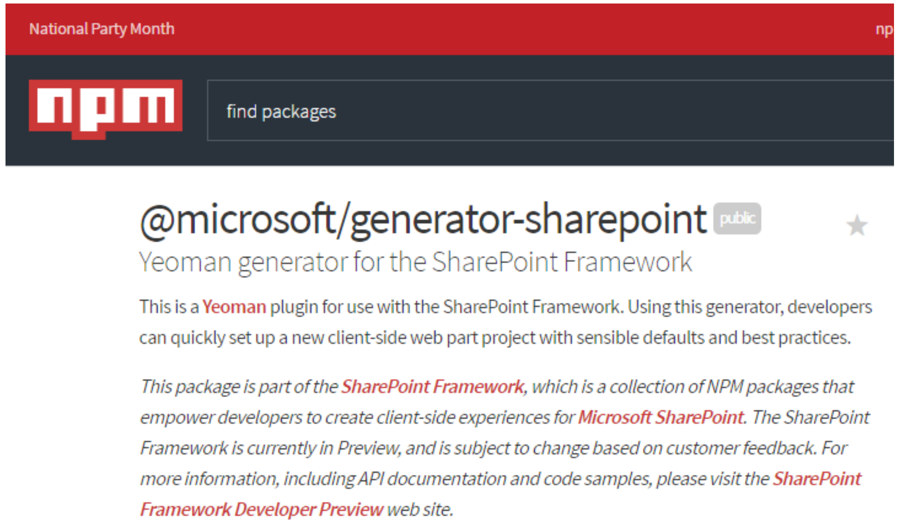
Download & Install: https://nodejs.org/en/

npm install -g yo

npm install -g @microsoft/generator-sharepoint

d:\ -> mkdir myTestDir-> cd myTestDir

yo @microsoft/sharepoint



@microsoft/generator-sharepoint public

Yeoman generator for the SharePoint Framework

This is a Yeoman plugin for use with the SharePoint Framework. Using this generator, developers can quickly set up a new client-side web part project with sensible defaults and best practices.

This package is part of the SharePoint Framework, which is a collection of NPM packages that empower developers to create client-side experiences for Microsoft SharePoint. The SharePoint Framework is currently in Preview, and is subject to change based on customer feedback. For more information, including API documentation and code samples, please visit the SharePoint Framework Developer Preview web site.

SPFx Basics

Prerequesites

Install Node.js (incl. Npm)

Install Yeoman & Gulp:

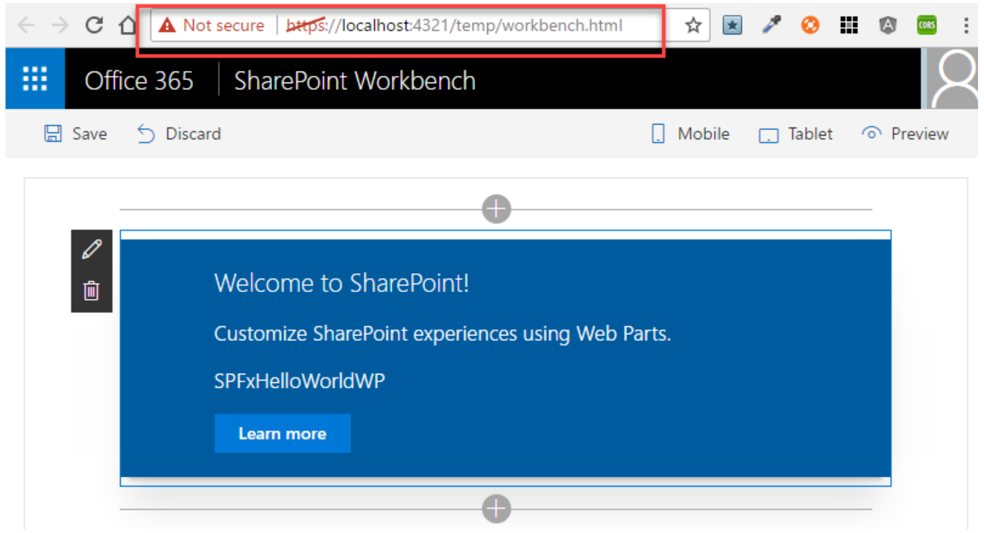
- npm install -g yo gulp

Install Yeoman SharePoint Generator:

- npm install -g @microsoft/generator-sharepoint

Trust Dev Certificate

- gulp trust-dev-cert
  - Must be executed from within an existing SPFx project



Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

SPFxHelloWorldWP

npm packages that build spfx

@microsoft/generator-sharepoint - A Yeoman plug

@microsoft/sp-client-base - Defines the core libraries

@microsoft/sp-client-preview - libraries that are still under development.

@microsoft/sp-webpart-workbench - SharePoint Workbench for testing and debugging

@microsoft/sp-module-loader - module loader, versioning

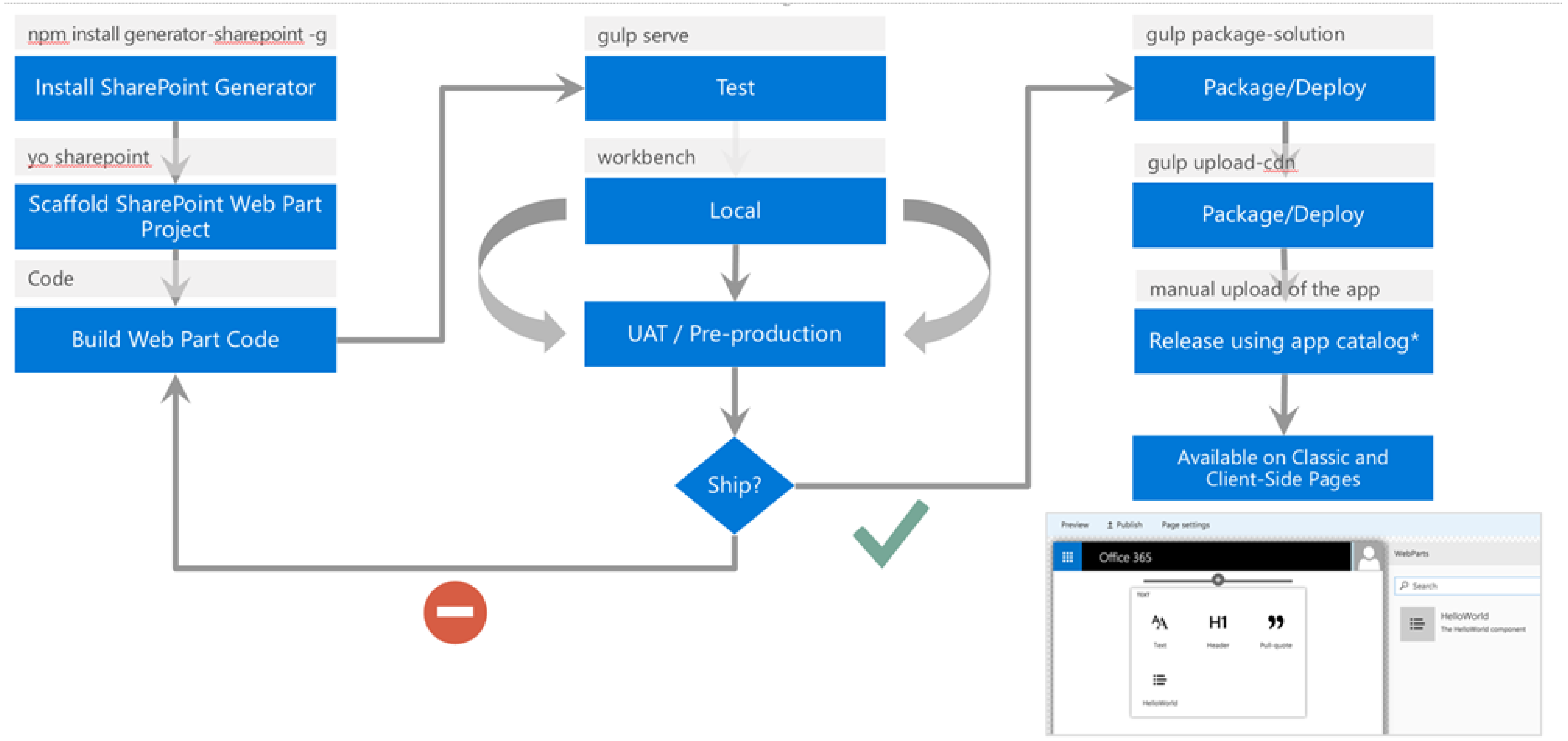
@microsoft/sp-module-interfaces - common module interfaces

@microsoft/sp-lodash-subset - custom bundle of loadsh

@microsoft/sp-tslint-rules - tslint rules

@microsoft/office-ui-react-bundle - Custom bundle of office-ui-fabric-react

Client-side Web Part Build Flow



Client-Side Web Part

Pure JavaScript implementation

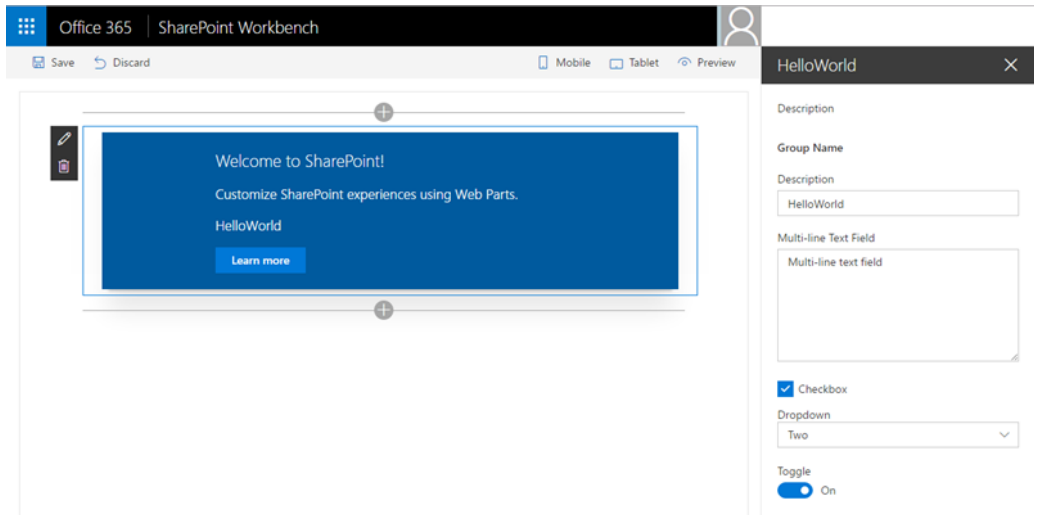
Web Part Value

- End users add functionality to SharePoint experiences
- Configurable, reusable, purpose built components built by developers
- Framework for connecting related components
- Context aware

Client-side Web Parts remain Web Parts

- Built for the modern, JavaScript-driven web
- Can run directly inside a SharePoint page, new and existing
- Web framework agnostic
- Support both on-premises and Office 365

- Samples & Docu @ https://github.com/SharePoint/sp-dev-fx-webparts



Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

HelloWorld

HelloWorld

Description

Group Name

Description: HelloWorld

Multi-line Text Field: Multi-line text field

Checkbox ✓

Dropdown: Two

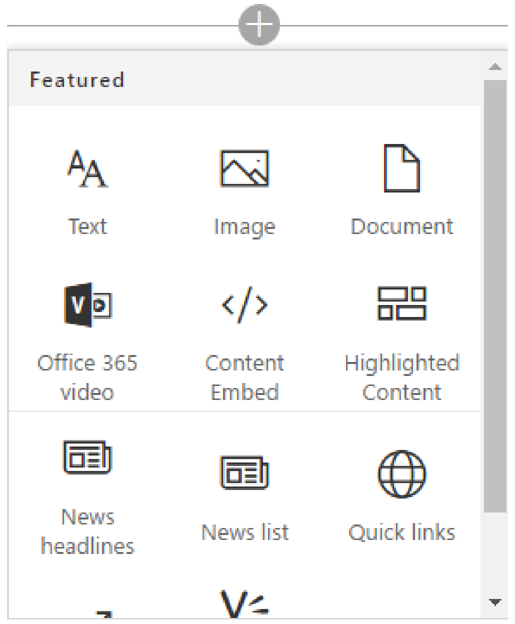
Toggle: On

Web Part Picker

Classic WebParts do not work in modern pages (html)

SPFx Web Parts work in modern pages

Add using the „new Web Part Picker“

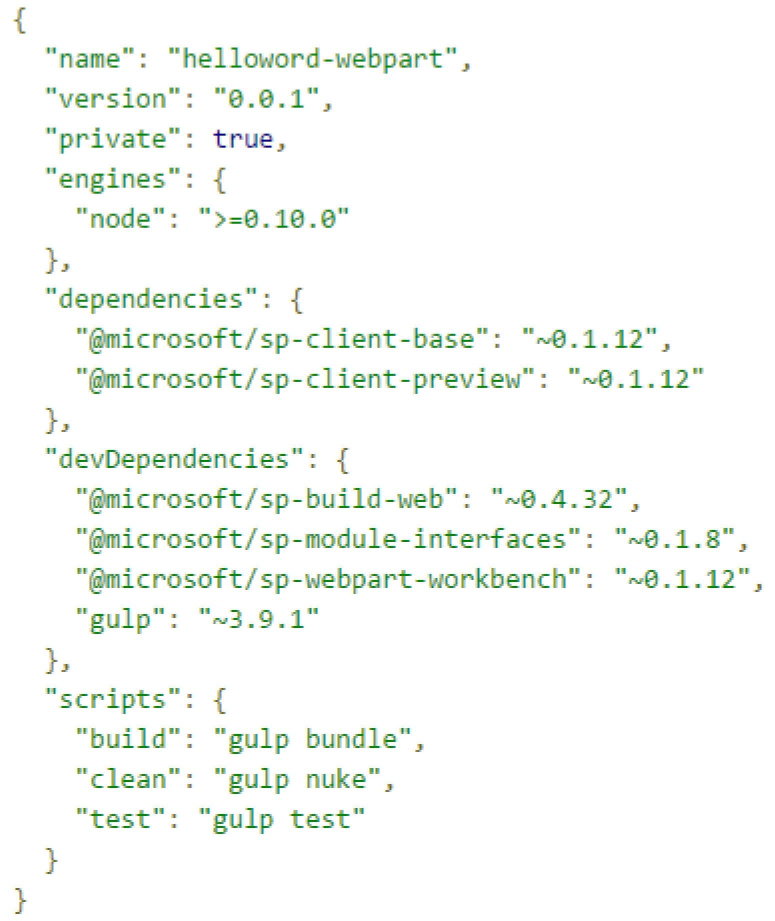


package.json

package.json is the configuration file for node.js

Packages are downloaded automatically,

when file is saved



```json
{
  "name": "helloword-webpart",
  "version": "0.0.1",
  "private": true,
  "engines": {
    "node": ">=0.10.0"
  },
  "dependencies": {
    "@microsoft/sp-client-base": "~0.1.12",
    "@microsoft/sp-client-preview": "~0.1.12"
  },
  "devDependencies": {
    "@microsoft/sp-build-web": "~0.4.32",
    "@microsoft/sp-module-interfaces": "~0.1.8",
    "@microsoft/sp-webpart-workbench": "~0.1.12",
    "gulp": "~3.9.1"
  },
  "scripts": {
    "build": "gulp bundle",
    "clean": "gulp nuke",
    "test": "gulp test"
  }
}
```

Workbench

gulp trust-dev-cert installs the developer certificate for https://*

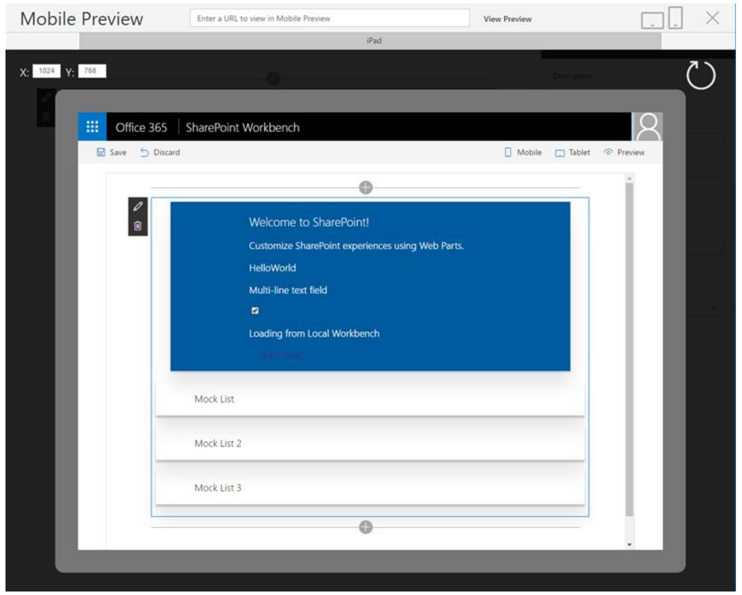
Developer design surface that enables you to quickly preview and test web parts without deploying them to SharePoint

Available for:

- Local Development
- Office 365

O365 Workbench URL:

- http://xxx/_layouts/workbench.aspx



Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

HelloWorld

Multi-line text field

Loading from Local Workbench

Files & Methods

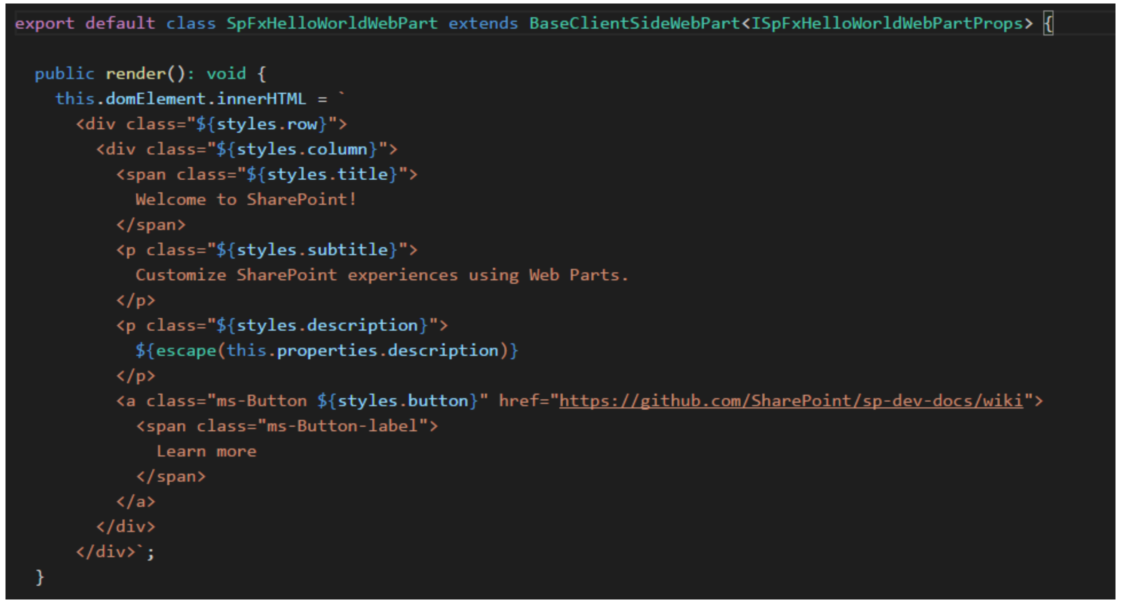
package-solution.json => Manifest

Important Files:

- xxWebPart.ts => Main implementation File
- xxWebPartProps.ts -> Properties of the WebPart
- xx.module.scss.scss => Sass Style Sheet

Main Method in xxWebPart.ts:

- render(): void
- Use ${ xx } syntax to include variables



```
export default class SpFxHelloWorldWebPart extends BaseClientSideWebPart<ISpFxHelloWorldWebPartProps> {

  public render(): void {
    this.domElement.innerHTML = `
      <div class="${styles.row}">
        <div class="${styles.column}">
          <span class="${styles.title}">
            Welcome to SharePoint!
          </span>
          <p class="${styles.subtitle}">
            Customize SharePoint experiences using Web Parts.
          </p>
          <p class="${styles.description}">
            ${escape(this.properties.description)}
          </p>
          <a class="ms-Button ${styles.button}" href="https://github.com/SharePoint/sp-dev-docs/wiki">
            <span class="ms-Button-label">
              Learn more
            </span>
          </a>
        </div>
      </div>`;
  }
```

Property Pane

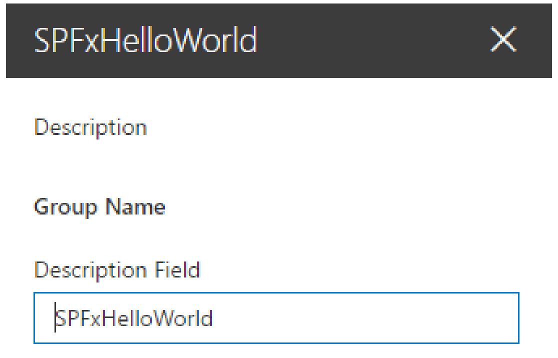
Used to configure Properties of a WebPart

Need to be implemented as interface

„Controls“ defined in sp-client-platform and need to be imported

The following field types are supported:

- Label, Textbox, Multi-line Textbox
- Checkbox, Dropdown
- Link, Slider, Toggle, …


SPFxHelloWorld

Description

Group Name

Description Field

SPFxHelloWorld

```
export interface IHelloWorldWebPartProps { targetProperty: string }
```

```
protected get disableReactivePropertyChanges():boolean{
  return true;
}
```

```
import { PropertyPaneTextField,
            PropertyPaneCheckbox,
            PropertyPaneLabel,
            PropertyPaneLink,
            PropertyPaneSlider,
            PropertyPaneToggle,
            PropertyPaneDropdown }
from '@microsoft/sp-client-preview';
```

package-solution.json

Creates an *.sppkg that can be uploaded to the App Catalog

Defines the package metadata

Generate package using:

- gulp package-solution

```json
{
    "solution": {
        "name": "helloworld-webpart-client-side-solution",
        "id": "ed83e452-2286-4ea0-8f98-c79d257acea5",
        "version": "1.0.0.0"
    },
    "paths": {
        "zippedPackage": "solution/helloworld-webpart.sppkg"
    }
}
```

SPFx CRUD

pageContext

Only available in O365 / SP2019 - Not in local workbench

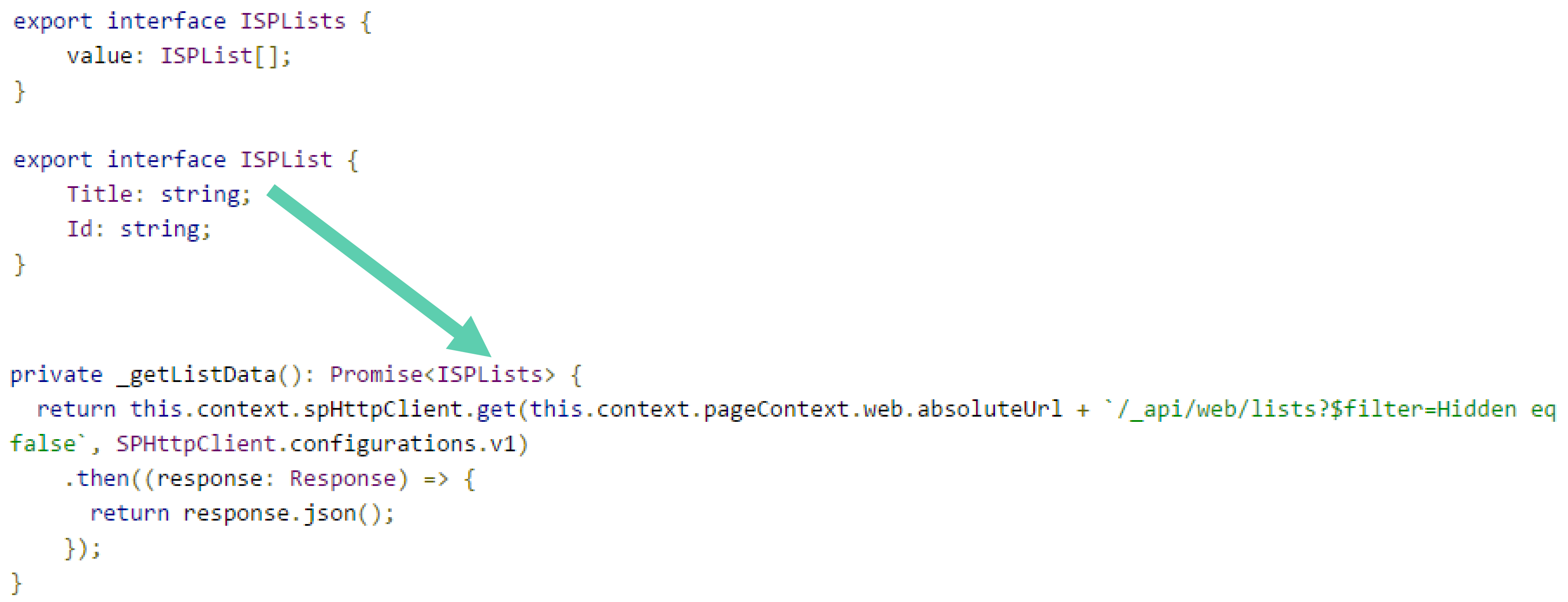
Access to information like:

- Web title
- Web absolute URL
- Web server-relative URL
- User login name

List model & Data access

List model acts as a (data) model for a certain list

SPHttpClient used to access SP data (like $.ajax or $http in jQuery or Angular)



```
export interface ISPLists {
    value: ISPList[];
}

export interface ISPList {
    Title: string;
    Id: string;
}


private _getListData(): Promise<ISPLists> {
  return this.context.spHttpClient.get(this.context.pageContext.web.absoluteUrl + `/_api/web/lists?$filter=Hidden eq 
false`, SPHttpClient.configurations.v1)
    .then((response: Response) => {
      return response.json();
    });
}
```

Environment & *EnvironmentType

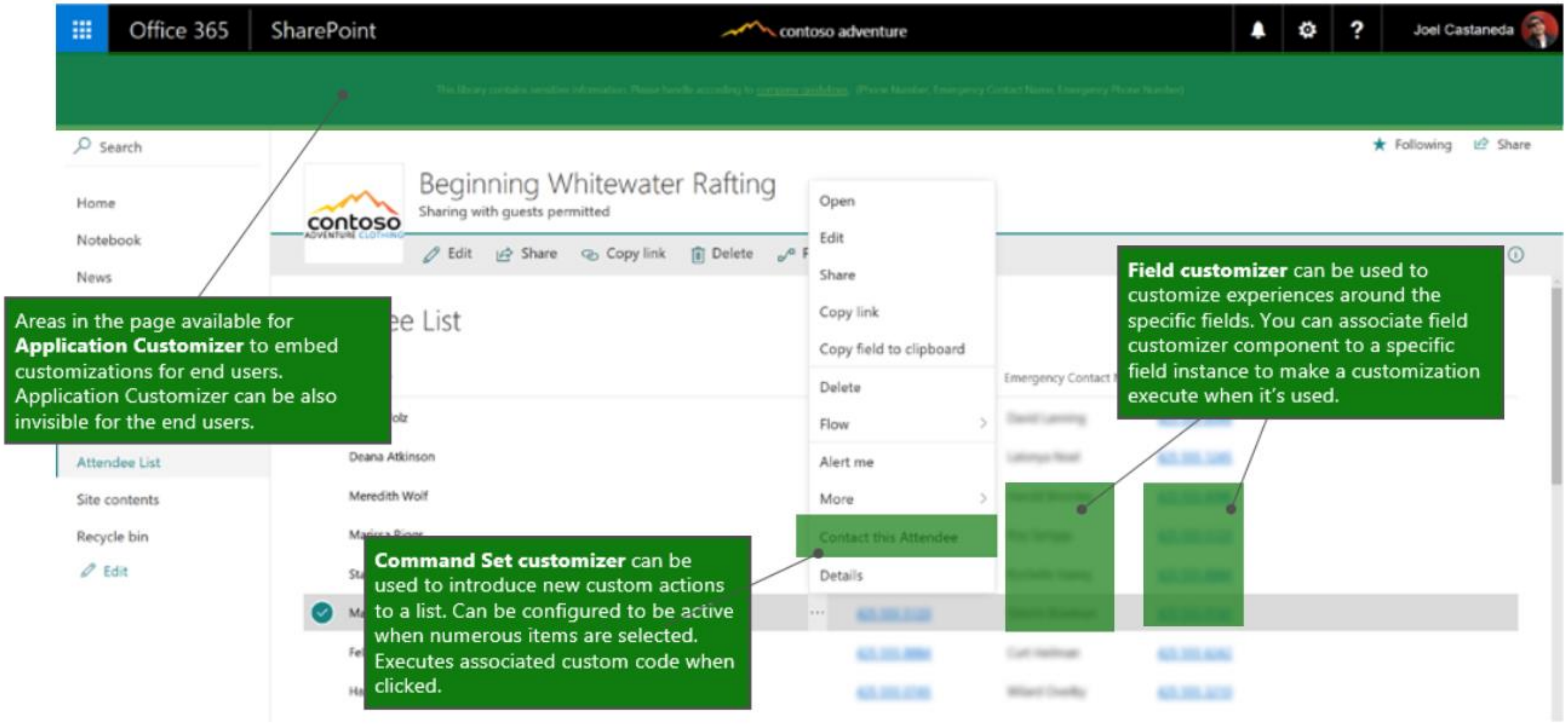
Let‘s you distinguish whether your are local or connected

```
import {
  Environment,
  EnvironmentType
} from '@microsoft/sp-core-library';
```

```
private _renderListAsync(): void {
  // Local environment
  if (Environment.type === EnvironmentType.Local) {
    this._getMockListData().then((response) => {
      this._renderList(response.value);
    });
  }
  else if (Environment.type == EnvironmentType.SharePoint ||
            Environment.type == EnvironmentType.ClassicSharePoint) {
    this._getListData()
      .then((response) => {
        this._renderList(response.value);
      });
  }
}
```

Extensions

Extension Types



Areas in the page available for **Application Customizer** to embed customizations for end users. Application Customizer can be also invisible for the end users.

Field customizer can be used to customize experiences around the specific fields. You can associate field customizer component to a specific field instance to make a customization execute when it's used.

Command Set customizer can be used to introduce new custom actions to a list. Can be configured to be active when numerous items are selected. Executes associated custom code when clicked.

Application Customizer

Adding JavaScript to pages in

- Sites / Webs / Lists

Injecting some content into every page on special placeholders

- e.g. a mega-menu/global navigation or message bar

Popping up dialog boxes in an integrated way

Adding items into certain toolbars/menus in SharePoint

Command Set

Similar to custom actions

Add buttons to lists & libraries

- Toolbar
- Item context menu

Specify conditions if command available when no / one / multiple items selected

Field Customizer

Similar to JSLink

Change rendering of a cell within a list

Only works in modern lists & libraries

For classic lists & libraries, use JSLink