BUILD YOUR FIRST SHAREPOINT FRAMEWORK (SPFx) WEBPART

ERIC OVERFIELD | @ericoverfield
SharePoint Advocate and Enthusiast PixelMill

SPTechCon San Francisco 2016
Tuesday December 6th
2:15 PM – 3:30 PM

https://github.com/eoverfield/SPFx-Workshop-Demos
INTRODUCTION  ERIC OVERFIELD

Founder and SharePoint Branding/UI Lead, PixelMill
Speaker, Teacher, Advocate, Author
SharePoint Community Organizer
Located in Davis, CA

Co-author: “Pro SharePoint 2013 Branding and Responsive Web Development”
(Apress – June 12th, 2013)
Order Your Copy
http://pxml.ly/zsqykd

Co-author: “Black Magic Solutions for White Hat SharePoint”
(August, 2013)

ericoverfield.com
@ericoverfield
WHAT YOU WILL LEARN TODAY

1. SHAREPOINT FRAMEWORK OVERVIEW
2. FRAMEWORK BUILD ENVIRONMENT
3. FRAMEWORK WEBPART COMPONENTS
4. WEBPART PROPERTIES, PACKAGING & MORE
HISTORY OF SHAREPOINT DEVELOPMENT MODELS

2003 FULL TRUST

2010 SANDBOX

2013 APP/ADD-IN MODEL

2016 CLOUD FRIENDLY SPFx

ericoverfield.com

@ericoverfield
WHAT IS THE FRAMEWORK?

A page and web part model with full support for client-side SharePoint development

- Open source tooling
  - nodeJS, npm, Yeoman, gulp, TypeScript, webpack and more
- Easy integration with SharePoint data
  - Rest API wrapper classes
- Available in the cloud and On-prem*
  - Now on SPO First Release tenants
  - On-prem availability scheduled for 2017

* ericoverfield.com  
@ericoverfield
HOW IT IS DIFFERENT

- Client-side rendering
  - No server side/compiled code / C#
  - IDE / Development platform agnostic
- New / modern tool chain
  - nodeJS / Yeoman / Gulp / Reach / etc
- Not dependent on JavaScript Injection
- No iFrame
  - Direct integration with the page model

ericoverfield.com
@ericoverfield
TOOLSET COMPARISON

SERVER SIDE DEVELOPMENT VS SPFx TOOL CHAIN
SET UP A BUILD ENVIRONMENT
PREREQUISITES

- Developer Tenant
  - Now available to First Release Tenants
- App catalog for deployment
  - http://dev.office.com/sharepoint/docs/spfx/set-up-your-developer-tenant#create-app-catalog-site
- Developer Site Collection
  - Document Library with custom site column – “ClientSideApplicationId” - text
    - May be any document library as long as site column exists
  - Upload the SharePoint Workbench (workbench.aspx) to dev site “Document” library
  - Or use PnP SharePoint Framework Developer Site Template
YOUR SPFx DEVELOPMENT ENVIRONMENT

- Use any most OS / workstation
- Install nodeJS (current Long Term Support (LTS) version)
- Yeoman and Gulp
  - c:/> npm i -g yo gulp
- SPFx Yeoman generator
  - c:/> npm i -g @microsoft/generator-sharepoint
- Use code editor
  - VS Code / Sublime Text / NotePad++, etc
CREATE YOUR FIRST **SPFx** Webpart

- `C:\> md helloworld-webpart`
- `C:\> cd helloworld-webpart`
- `C:\> yo @microsoft/sharepoint`
  - This will take a while to load all components
- `C:\> gulp serve`
- Check out your first webpart!
  - It “should” be that simple
YOUR FIRST SPFx WEBPART
WELCOME TO A NEW DEVELOPMENT PARADIGM
Get to know your Webpart folder structure

- **src**: primary webpart TypeScript source code
- **config**: json configuration files for build process
- **typings**: TypeScript typings for JS libs
- **lib**: Build files (TS compiled JS) ready for bundle
- **dist**: Final web ready code for distribution
- **sharepoint**: .spapp file for App Catalog
- **node_modules**: NodeJS modules (JS) for toolchain
Webparts normally need custom properties

- Define: /src/webparts/"webpart"/"webpart"Props.ts
  - Add in JSON
- Default values: /src/webparts/"webpart"/"webpart".manifest.json
  - Add in JSON: preconfiguredEntries.properties
- Display: /src/webparts/"webpart"/"webpart".ts
  - Method: protected get propertyPaneSettings(): IPropertyPaneSettings {}
  - Override onchange: /src/webparts/"webpart"/"webpart".ts
    - Method: public onPropertyChange(propertyPath: string, newValue: any) {}
ACCESS DYNAMIC DATA IN PROPERTY PANES

- Method `propertyPaneSettings` returns a static `IPropertyPaneSettings`
  - Method does not allow for Promises / dynamic data

- Solution: Load dynamic data within `propertyPaneSettings` then trigger pane refresh
  - `this.configureStart();`  //cause pane to refresh with new data
CONNECT TO SHAREPOINT / DYNAMIC DATA

- SPFx provides tools to quickly interact with external API data
  - TypeScript Http classes within @microsoft/sp-client-base
- BasicHttpClient
  - Basic set of features for REST operations with any external resource
- HttpClient
  - REST calls against SharePoint
- Handles context, security, etc. Could use BasicHttpClient if desired

```javascript
import { HttpClient } from '@microsoft/sp-client-base';
private _httpClient: HttpClient;

this._httpClient.get(this._webAbsoluteUrl + `/_api/Lists/?$select=Id,Title&$filter=Hidden ne true`) .then((response: Response) => {
  return response.json();
});
```
SPFx WEBPART: OVERVIEW, PROPERTIES AND SHAREPOINT DATA
PACKAGE YOUR WEBPART FOR DEPLOYMENT

- C:\> gulp package-solution
  - Creates /sharepoint/”webpart”.spapp
- Add .spapp to app catalog
- Add app to SharePoint development site
- Add webpart in development to content page
  - Webpart is still pointing to local host for JS
- Configure CDN for full webpart deployment
  - https://dev.office.com/sharepoint/docs/spfx/web-parts/get-started/deploy-web-part-to-cdn
CONNECT TO EXTERNAL LIBRARIES / FRAMEWORKS / RESOURCES

- External libraries and component require Typings
  - i.e. for jQuery:
    - C:\> npm i --save jquery
    - C:\> npm i -g tsd
    - C:\> tsd install jquery jqueryui --save
  - Within webpart
    - import * as myjQuery from 'jquery';
    - Access: let $workspace: jQuery = myjQuery('#spPageChromeAppDiv')
COMMAND REFERENCE

- `yo @microsoft/sharepoint` // create a new base project
- `gulp serve` // compile and start up local workbench
- `gulp serve --nobrowser` // same as serve, but with no browser loaded
- `gulp package-solution` // compile / create spapp file for redeployment in "Sharepoint" folder
- `gulp bundle --ship` // generate assets (js, css, etc) for deployment to CDN
- `gulp package-solution --ship` // package ready for shipment --ship is just temporary during dev preview
- `gulp deploy-azure-storage` // deploy assets to Azure storage as defined in config/deploy-azure-storage.json
BEST PRACTICES

- Learn TypeScript!

- Use HttpClient to connect to SharePoint
  - BasicHttpClient for other API’s

- Use frameworks and libraries that already has typings

- Office UI Fabric available for consistent styling
RESOURCES  THE SHAREPOINT FRAMEWORK

SharePoint Framework documentation
https://github.com/SharePoint/sp-dev-docs

SharePoint Framework API
https://sharepoint.github.io/

Set up an Office 365 Developer Tenant
https://dev.office.com/sharepoint/docs/spfx/set-up-your-developer-tenant

Build your first webpart
https://dev.office.com/sharepoint/docs/spfx/web-parts/get-started/build-a-hello-world-web-part

Webpart with React and Office UI Fabric
https://dev.office.com/sharepoint/docs/spfx/web-parts/get-started/use-fabric-react-components

Get an introduction to the SharePoint Framework

Demo Source Code
https://github.com/eoverfield/SPFx-Workshop-Demos
BUILD YOUR FIRST
SHAREPOINT FRAMEWORK WEBPART

THANK YOU
QUESTIONS?

Order Your Copy
http://pxml.ly/zsqykd

“Pro SharePoint 2013 Branding and Responsive Web Development”
(Apress – June 12th, 2013)

@ericoverfield

SPTechCon SF 2016
http://pxml.ly/EO-SPFx-Webpart