SPFx: An ISV Insight to Microsoft's latest customization model

By: Shai Petel

Microsoft Partner
Gold Application Development
Silver Collaboration and Content

www.KWizCom.com

KWizCom
KNOWLEDGE WORKER COMPONENTS

Live chat call 1-855-594-9266

KWizCom Forms

True SharePoint-Native Forms & Mobile solution.

Enhance SharePoint, don't replace it...

Read more

"We have chosen KWizCom's web parts to make the surf experience of our end-users simpler and more natural."

Guy Vermeulen,
Project Leader Web and Software,
Vlerick Leuven Gent Management School, Belgium

SharePoint 2013 Apps

Org. chart web part

Multi-Row Forms in SharePoint!? **Repeating Rows Column!**

Form custom layouts

Shai Petel

Director of Research and Development at KWizCom

MCT, MCPD
6 times MVP: SharePoint (2011-2016)

shai@kwizcom.com | @shaibs | http://kwizcom.blogspot.com

Focus:

- Publishing versions, upgrades
- Pushing updates through CDN
- Shared code and external libraries
- Custom PropertyPane properties
- Code, code, code.

Overview of extensibility opportunities in SharePoint’s history

SHAREPOINT DEVELOPMENT MODELS

SharePoint Development Models

Microsoft SharePoint Portal Server 2001

Microsoft Office SharePoint Server 2007

SharePoint 2013

SharePoint Online

2001	2003	2007	2010	2013	SPO	2016
STS	SPS	MOSS	SharePoint Foundation	SharePoint Foundation	SharePoint Online	SharePoint Server
SPS	WSS	WSS	SharePoint Server	SharePoint Server	Group sites *	

Microsoft Office SharePoint Portal Server 2003

Microsoft® SharePoint® 2010

SharePoint 2016

SharePoint Development Models

- 2001
  - ASP (no, I didn't forget the .Net)
  - Page parts, tokens (_wpq_, _wpr_)
  - No clear packaging and deployment
- 2003
  - Web parts (ASP.Net custom controls)
  - Packaged in CAB
- 2007
  - Features
  - WSP package
- 2010
  - Timer jobs
  - Sandbox
- 2013
  - SharePoint hosted ~~apps~~ add-ins
  - Provider hosted ~~apps~~ add-ins
  - JSLink / CSR
- SPO
  - SharePoint hosted ~~apps~~ add-ins
  - Provider hosted ~~apps~~ add-ins
  - ~~JSLink / CSR No code sandbox solutions~~
  - SharePoint Framework (SPFx)

Brief explanation of the SharePoint Framework

WHAT IS SPFX?

What is SPFx?

“a Page and Part model enables fully supported client-side development for building SharePoint experiences, easy integration with the SharePoint data and support for open source tooling development.”

What is SPFx? - advantages

- ☺ Runs in the context of the current user. No iFrames
- ☺ There is a life cycle that the developer is involved in
- ☺ It is framework agnostic *
- ☺ Open source tool chain
- ☺ Performance
- ☺ Solutions approved by the tenant admin (or their delegates)
- ☺ Classic & modern pages
- ☺ Can be used in noscript sites

What is SPFx? - disadvantages

- ☹ SPO: up-to-date - on prem: only 2016, older build
- ☹ Extremely big learning curve
- ☹ Limited artifact types supported
- ☹ Publishing updates ~
- ☹ No support for the app store

SPFx compared to previous generations

DOES IT DELIVER?

How would you render the UI of your web part?

CHOICE OF UI FRAMEWORK

Choice of UI framework

- SPFx is framework agnostic by design. Really?
- This is the most important decision when building a new project

Choice of UI framework

React JS

ReactJS - the clear leader & first class citizen

- This is the engine MSFT use themselves
- The PropertyPane is built on react, and the engine wrapping your WebPart is too.
- The only Office UI Fabric components actively supported by Microsoft

Choice of UI framework

KnockoutJS

- Good when you need dynamic templates (i.e. user supply / customize HTML output)

- Lack of fabric components. Use FabricJS

- Building in --ship mode removes KO comments

Choice of UI framework

Handlebars, angular, etc…

Experiment, choose the one that fits your skills and needs.

AngularJS has a good community supported office UI fabric components library.

BUILDING YOUR SOLUTION

Building your solution

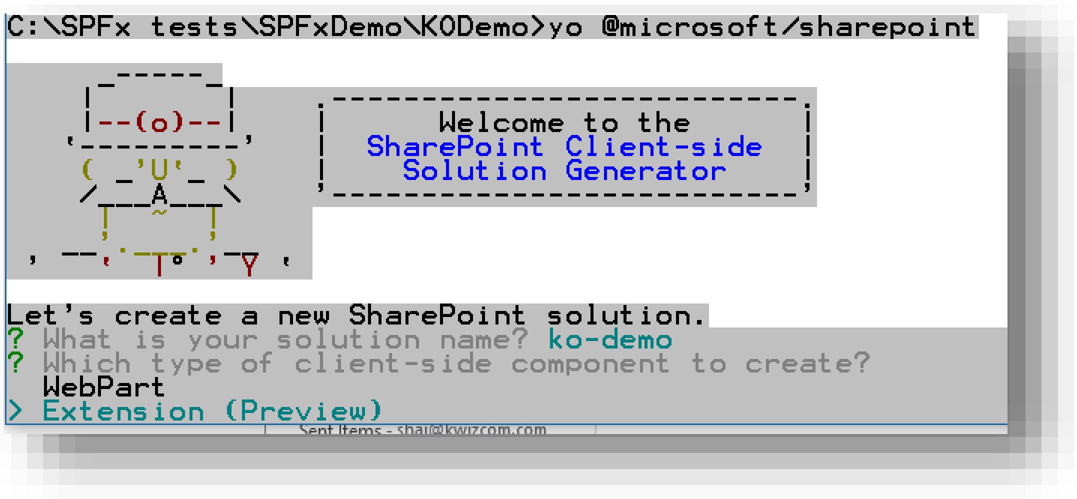
Building a new project

- Create a folder
- Run “yo @microsoft/sharepoint”
- Set name, description

Building your solution

Building a new project

- Select “WebPart”

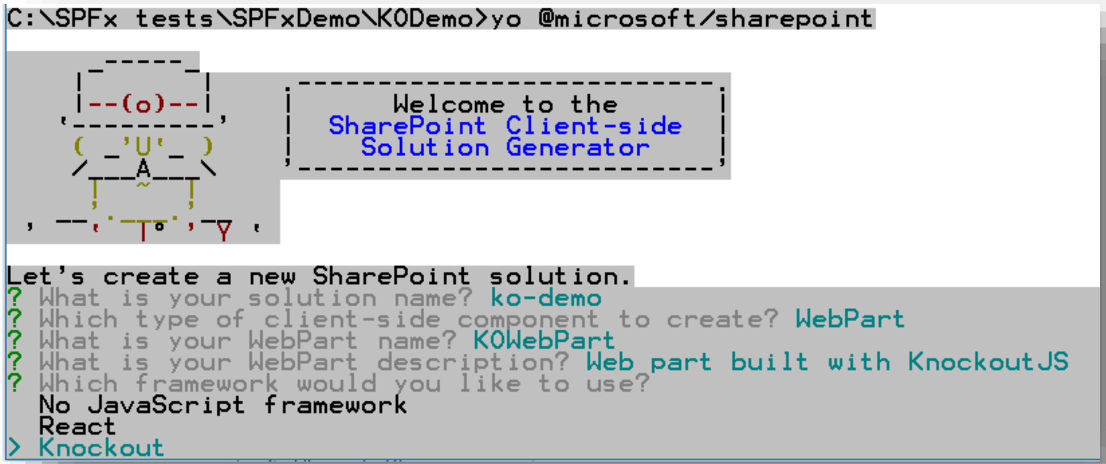


```
C:\SPFx tests\SPFxDemo\KODemo>yo @microsoft/sharepoint

                      .--------------------------.
                      |       Welcome to the      |
                      |  SharePoint Client-side   |
                      |    Solution Generator     |
                      '--------------------------'

Let's create a new SharePoint solution.
? What is your solution name? ko-demo
? Which type of client-side component to create?
  WebPart
> Extension (Preview)
```

Building your solution

Building a new project

- Select your JavaScript framework



```
C:\SPFx tests\SPFxDemo\KODemo>yo @microsoft/sharepoint

                              ---------------------------
                             |      Welcome to the       |
                             | SharePoint Client-side    |
                             |   Solution Generator      |
                              ---------------------------

Let's create a new SharePoint solution.
? What is your solution name? ko-demo
? Which type of client-side component to create? WebPart
? What is your WebPart name? KOWebPart
? What is your WebPart description? Web part built with KnockoutJS
? Which framework would you like to use?
  No JavaScript framework
  React
> Knockout
```

Building your solution

Add several artifacts to a single package

- Currently only web parts are supported on prem
- Application Customizer, Field Customizer and ListView Command Set – very limited & online only
- Add more artifacts by running
  “yo @microsoft/sharepoint”
  inside an existing project folder

Publishing your SPFx solution

PUBLISHING

Publishing

Content Delivery Network (CDN)

- Specify CDN in config/write-manifests.json
- Host script files, css, images, html
- Push minor fixes to clients
- Non-ISVs on SPO can use Office 365 public CDN

Publishing

Control production build file names

Running gulp --ship will produce a bundle file for production:

- Default: react-web-part.bundle_a4b2ffc1a3b03f7ce4b5bd78bdd7ac62.js
- Recommend: react-web-part.1.0.0.0.js

Publishing

Publish a package

1. Update web part version in config/package-solution.json
2. Run gulp --ship
3. Go to temp/deploy folder
4. Rename the JS file (I use {project}.{version}.js)
5. Edit the json file: replace the bundle JS file to the new name *
6. Run gulp package-solution --ship
7. Take your new packages from the SharePoint/solution folder
8. Drop your new JS to your CDN **

[optional steps]

Publishing

Publishing updates: Do I need to release a new package?

- You might be able to push updates to your customers
- Check if your new version is backward compatible
- Track version via a static variable

Publishing

Publishing updates: Update a package

- When a new package is needed – follow “publish a package” steps
- When a new package is not needed
  1. Update BuildVersion in your code *
  2. Run gulp --ship
  3. Go to temp/deploy folder
  4. Copy the new JS file content into your existing file on your CDN
  5. To use your own minifier, drop the --ship flag, take the file from the dist folder

Publishing

Installing / upgrading package

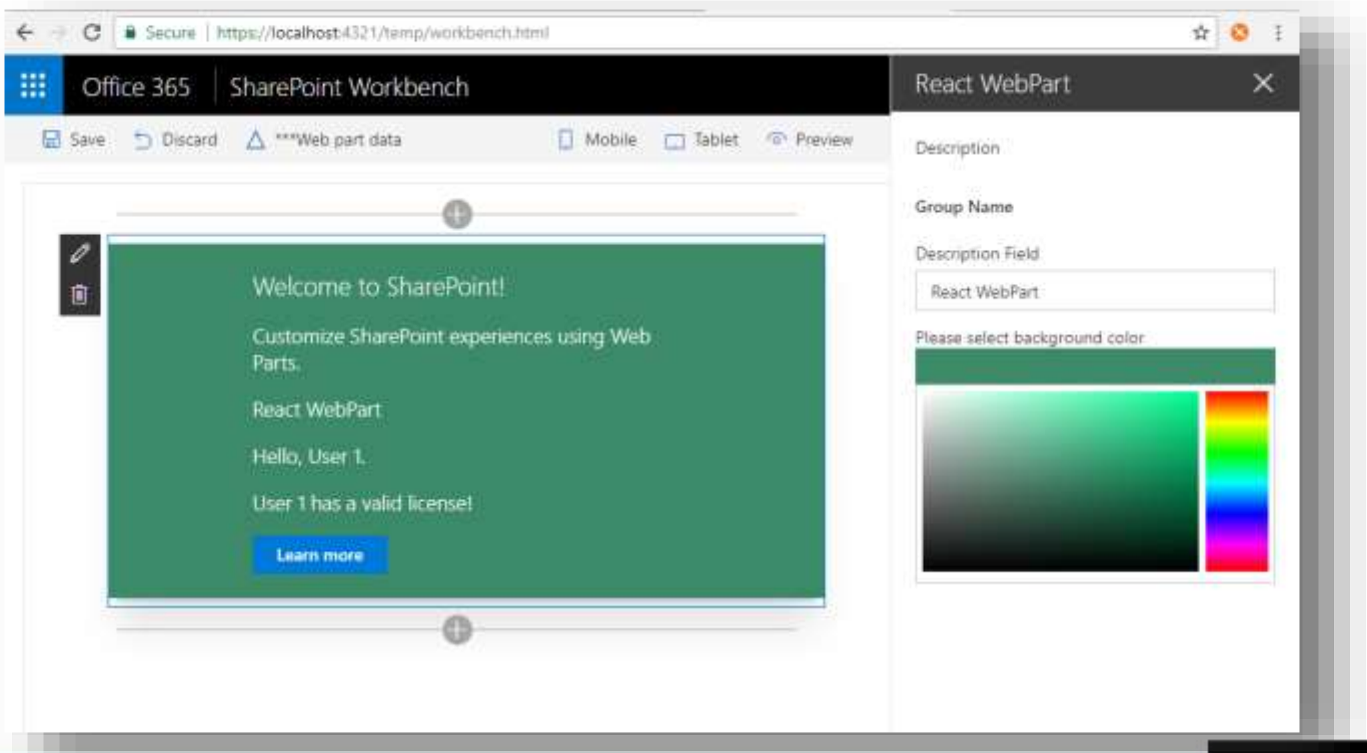
1. Upload package to the App Catalog
2. Trust the app
3. Fixed? ~~A new version: delete old package first~~

Dev time environments and options

DEVELOPMENT OPTIONS

Development options - local workbench

gulp serve
https://localhost:4321/temp/workbench.html


Secure | https://localhost:4321/temp/workbench.html

React WebPart

Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

React WebPart

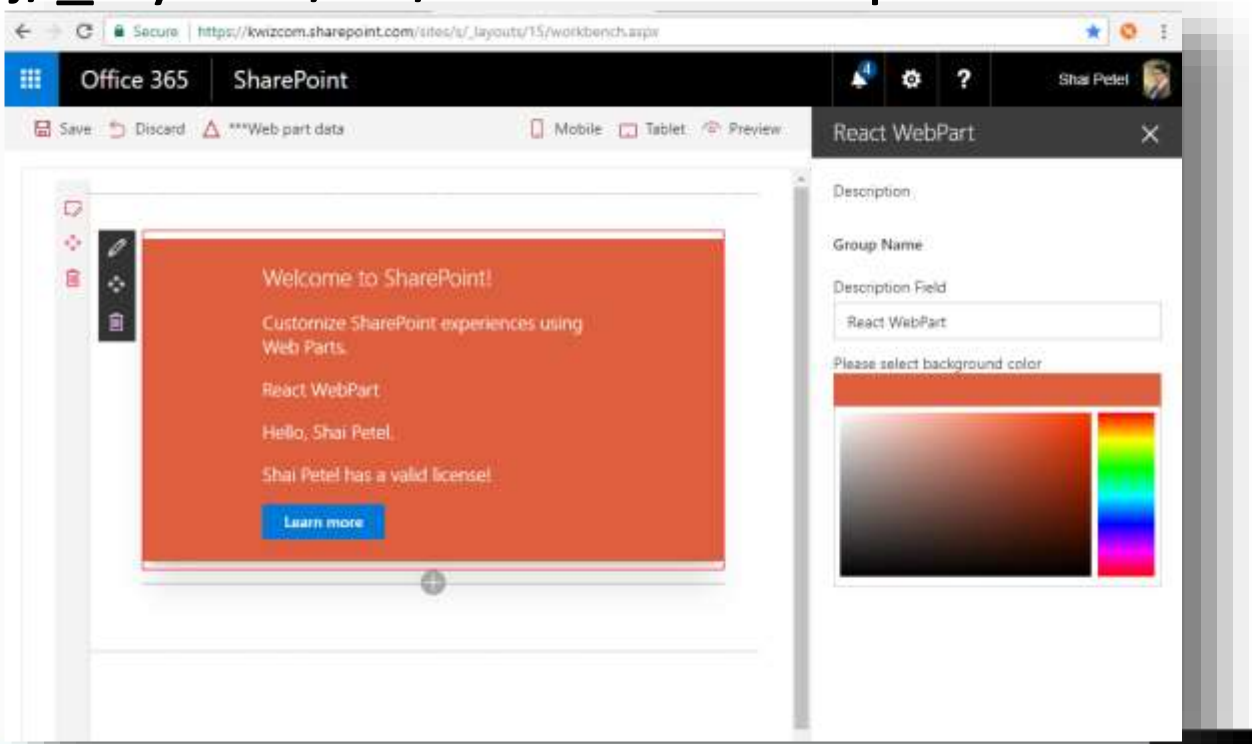
Hello, User 1.

User 1 has a valid license!

Development options – online workbench

gulp serve --nobrowser

{spo site}/_layouts/15/workbench.aspx



Welcome to SharePoint!

Customize SharePoint experiences using Web Parts.

React WebPart

Hello, Shai Petel.

Shai Petel has a valid license!

Description

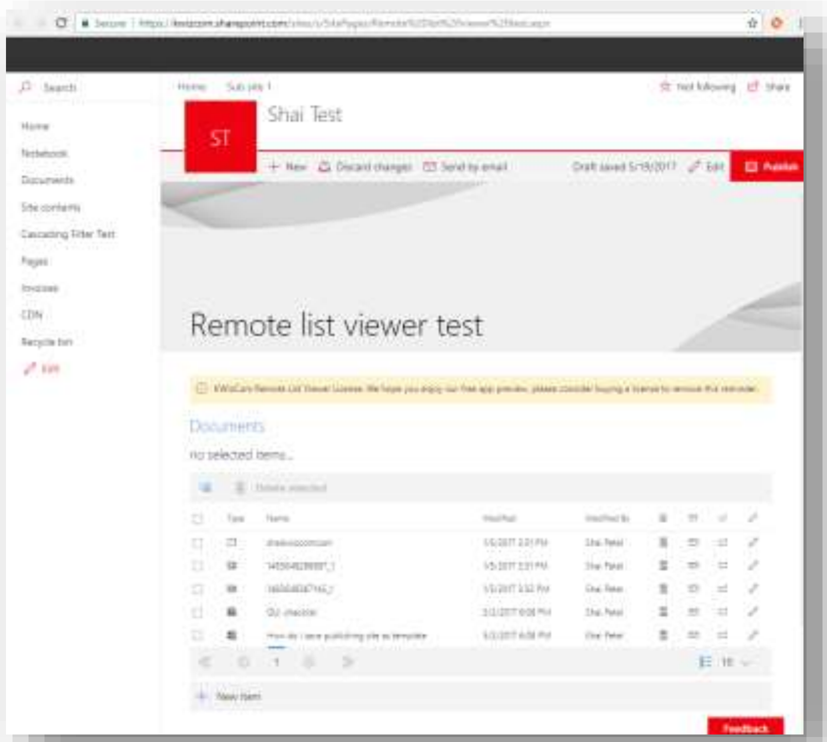
Group Name

Description Field: React WebPart

Please select background color

Development options – classic/modern page

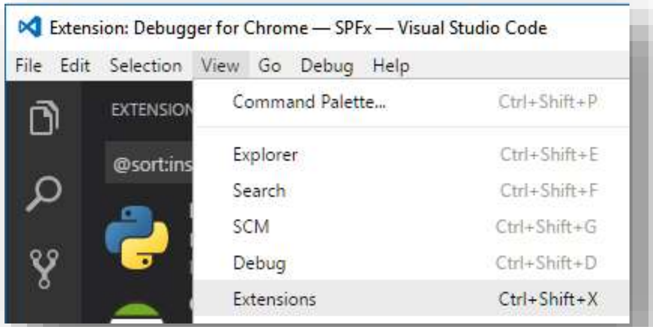
1. gulp --ship
2. gulp package-solution --ship
3. Publish to CDN
4. Deploy to catalog
5. Add app to your site*

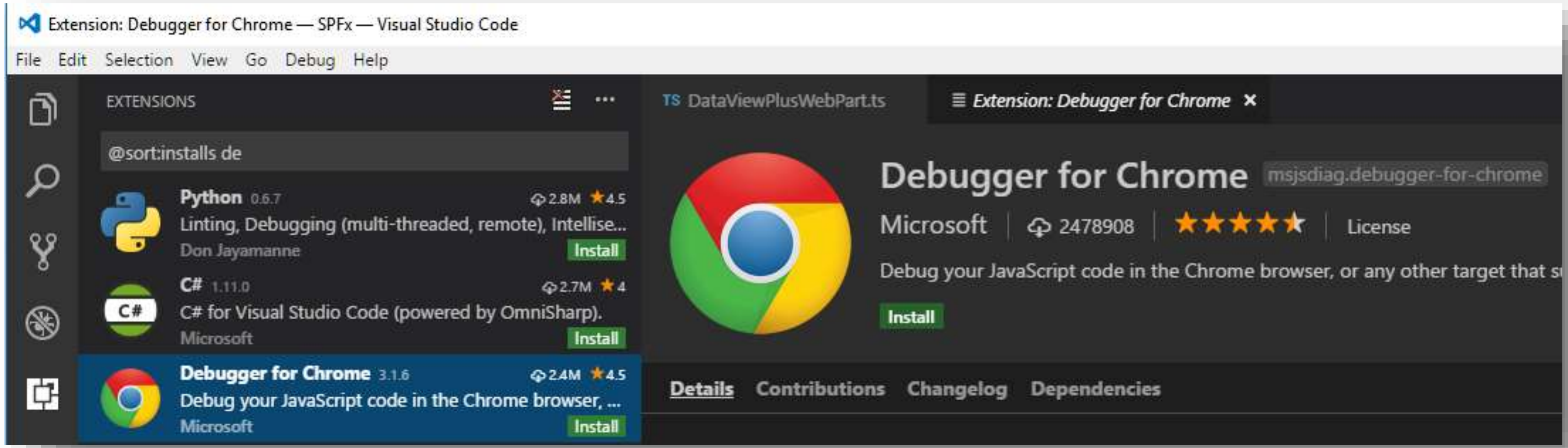


Development options

Debugging using VSCode in Chrome

- Open VSCode, View->Extensions
- Install “Debugger for chrome”

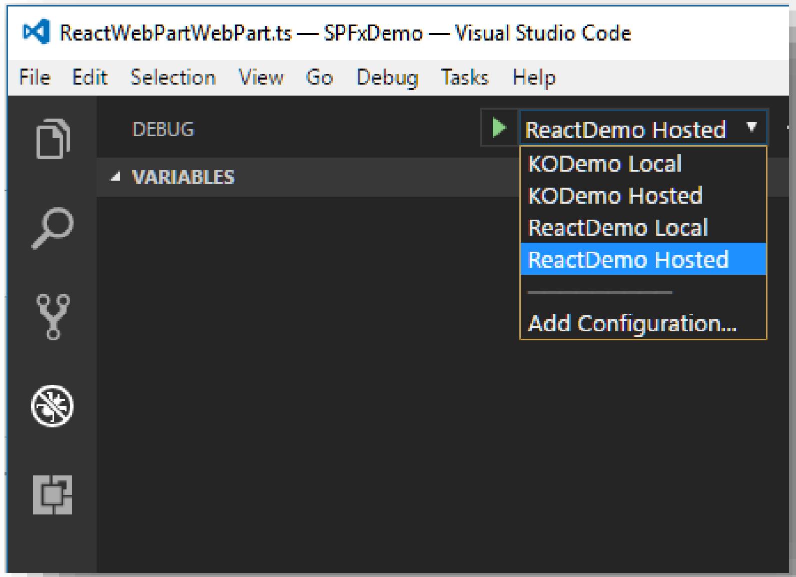

Extension: Debugger for Chrome — SPFx — Visual Studio Code
File Edit Selection View Go Debug Help
EXTENSION
@sort:ins
Command Palette... Ctrl+Shift+P
Explorer Ctrl+Shift+E
Search Ctrl+Shift+F
SCM Ctrl+Shift+G
Debug Ctrl+Shift+D
Extensions Ctrl+Shift+X


Extension: Debugger for Chrome — SPFx — Visual Studio Code
File Edit Selection View Go Debug Help
EXTENSIONS
@sort:installs de
Python 0.6.7 2.8M 4.5
Linting, Debugging (multi-threaded, remote), Intellise...
Don Jayamanne
Install
C# 1.11.0 2.7M 4
C# for Visual Studio Code (powered by OmniSharp).
Microsoft
Install
Debugger for Chrome 3.1.6 2.4M 4.5
Debug your JavaScript code in the Chrome browser, ...
Microsoft
Install
DataViewPlusWebPart.ts
Extension: Debugger for Chrome
Debugger for Chrome msjsdiag.debugger-for-chrome
Microsoft 2478908 License
Debug your JavaScript code in the Chrome browser, or any other target that s
Install
Details Contributions Changelog Dependencies

Development options

Debugging using VSCode in Chrome

- Create launch.json
- Select your configuration
- Start gulp serve --nobrowser
- Press F5



Using npm modules

External dependencies

Legacy script (global)

DEPENDENCIES

Dependencies - npm

Bundling npm modules

- When you wish to use an npm available module
- If it is a <u>small</u> file, and <u>not reused</u> across many different components

* Bundling includes this entire module inside your JS file.

Dependencies - npm

How?

- npm install {package} --save
- tsd install {package} --save (or create your own typings {package}.d.ts)

In your web part code:

- import * as {object} from ‘{package}’
- import {object} from ‘{package}’

Dependencies - npm

External npm modules

- When you wish to use an npm available module
- If it is a large file, or reused across many different components

* External modules will be loaded from a CDN

Dependencies - npm

How?

- Follow previous steps to load the npm module
- Mark this module as external, to prevent bundling:
  - Edit config/config.json
  - Add this under “externals” object:

“<package>”: “{path to js file}”

Dependencies - npm

Important!

- .gitignore excludes node_modules folder
- run “npm install” when moving computers
- Dependencies change and your project may break as a result - npm shrinkwrap!

Dependencies – global library

Loading a legacy script file using config.json

- When you want to load an external script file from a CDN
- Add it as an external (with a global declaration if needed)
- Import it in your web part
- Optionally, create typings for your global.
  module declare ‘extLib’{…}
- Alternatively, you can declare it as type any:
  declare var extLib: any;

Dependencies – global library

Loading a legacy script file programmatically

- When you don’t want to change your package signature, or when you want to load it conditionally

- When loading an external script file from a CDN

Dependencies – global library

Loading a legacy script file programmatically

How?

- Declare its global:

```
declare var kwfabric: any;
```

- Use a promise to load it, execute your code once the promise resolves:

```
SPComponentLoader.loadScript(`https://apps.kwizcom.c
om/libs/office-ui-fabric-
js/1.4.0/js/fabric.js?prefix=kw`, {
globalExportsName: 'kwfabric' }).then((p: any) => {
  //use kwfabric
});
```

Dependencies –fabric js

Using Office UI Fabric JS

- If you can, use react. It comes with amazing support for Fabric UI.
- If not, I recommend using Fabric JS.
- Optionally, use KWizCom's hosted branch of Office UI Fabric JS:

http://kwizcom.blogspot.ca/2017/03/using-office-ui-fabric-js-in-spfx.html

Reuse your code between different SPFx projects

SHARED REUSABLE CODE

Shared Reusable Code

- **Create folder for your shared code**
- If you plan to use type script
- Use your code relatively to your project folder

Shared Reusable Code

- Create folder for your shared code
- **If you plan to use type script**
  - Install dependencies manually:
    npm install @microsoft/sp-core-library@~1.4.0
    npm install @microsoft/sp-webpart-base@~1.4.0
  - add tsconfig.json, run tsc to compile
- Use your code relatively to your project folder

Shared Reusable Code

- Create folder for your shared code
- If you plan to use type script
- **Use your code relatively to your project folder**

```
import Utilities from '../../../../SharedCode/Utilities';
```

Shared Reusable Code

Consider creating an npm package

☺ Benefits: great versioning and manageability

☹ Disadvantages: overhead in managing the package and publishing updates

Shared Reusable Code - npm

Creating a new package

- Run “npm init”
- Edit package.json, add dependencies
- run “npm install”
- If you plan to use type script:
  - Add tsconfig.json
  - Run tsc to build JS files from TS
  - Specify "typings" in your package.json

Shared Reusable Code - npm

Using your new package

- Edit SPFx package dependencies, add your package:
  "kwizcom-license": "file:../kwizcom-license“
- Bundled by default, should you mark as external?
  - Plan on using this package from several different projects?
  - Its JS file is large?
- How?
  - Edit config/config.json add to externals:
    "kwizcom-license":
    https://kwizcom.sharepoint.com/sites/s/cdn/License.js
  - Now, you can also push minor updates/fixes without having to re-publish all your projects (when you keep backward compatibility).

Shared Reusable Code - npm

Publishing updates?

- Update version number in package.json
- Run “npm update” everywhere you used it to update the package

CONSUMING DATA

Consuming data

Connecting to data

- SharePoint
- Microsoft Graph

* Local workbench not supported

Consuming data

Requesting SharePoint content

- Use SPHttpClient (this.context.spHttpClient) post/get to make rest requests
- Load JSOM into your SPFx
- DO NOT try to use the JSOM global objects. No one guarantees they will be there in modern pages/other pages.

Consuming data

What is Microsoft Graph API?

- Graph API is a rest end point for all of your users O365 services.
- This includes access to his OneDrive, Mail, Calendar, Contacts, SharePoint, and the list goes on.

Consuming data

Requesting Graph content

- Today – do it yourself. Register an application for graph, request the trust/permissions you need, and make rest requests.
- Soon – Microsoft will share one token/key we can all use for basic read access
  - Use GraphHttpClient/MSGraphClient (preview)
  - Will automatically use a shared available key/token, with limited set of permissions
  - This token will be limited to accepting requests ONLY from SharePoint online hosted sites, as a security measure.

What can you do with the PropertyPane?

Building custom controls

ADVANCED PROPERTYPANE

Advanced PropertyPane

Basic concepts

- Rendered based off a dynamic JSON definition object.
- The tool part will re-render when changing a dropdown value, or when clicking a button.
- When a property changes, it will update your web part by re-calling the "render" event.
- Basic structure:
  - Pages -> Groups -> Properties -> Property

Advanced PropertyPane

Advanced concepts - Validations

- onGetErrorMessage return error message string, or function returning promise.
- During validation, you can control and change other properties.
- In most cases, it is best to fire on change event to notify SharePoint of the change.

Advanced PropertyPane

Advanced concepts - Custom controls

- You can create your own controls for the property pane
- Use the render method to render your own controls
- Get configuration info from the web part, such as context, label and where to store your values.
- Notify the web part on value changes (You will need to create that logic – see next slide)

Advanced PropertyPane

Advanced concepts - Trigger on change

- First, add the two helper function:

```
import { get, update } from "@microsoft/sp-lodash-subset";
```

- Create a handler for your property "on change"

```
private onPropertyChange(propertyPath: string, newValue: any):
void {
  const oldValue: any = get(this.properties, propertyPath);
  update(this.properties, propertyPath, (): any => { return
newValue; });
  this.onPropertyPaneFieldChanged(propertyPath, oldValue,
newValue);
  if (!this.disableReactivePropertyChanges)
    this.render();//update the webpart
}
```

KWizCom
KNOWLEDGE WORKER COMPONENTS

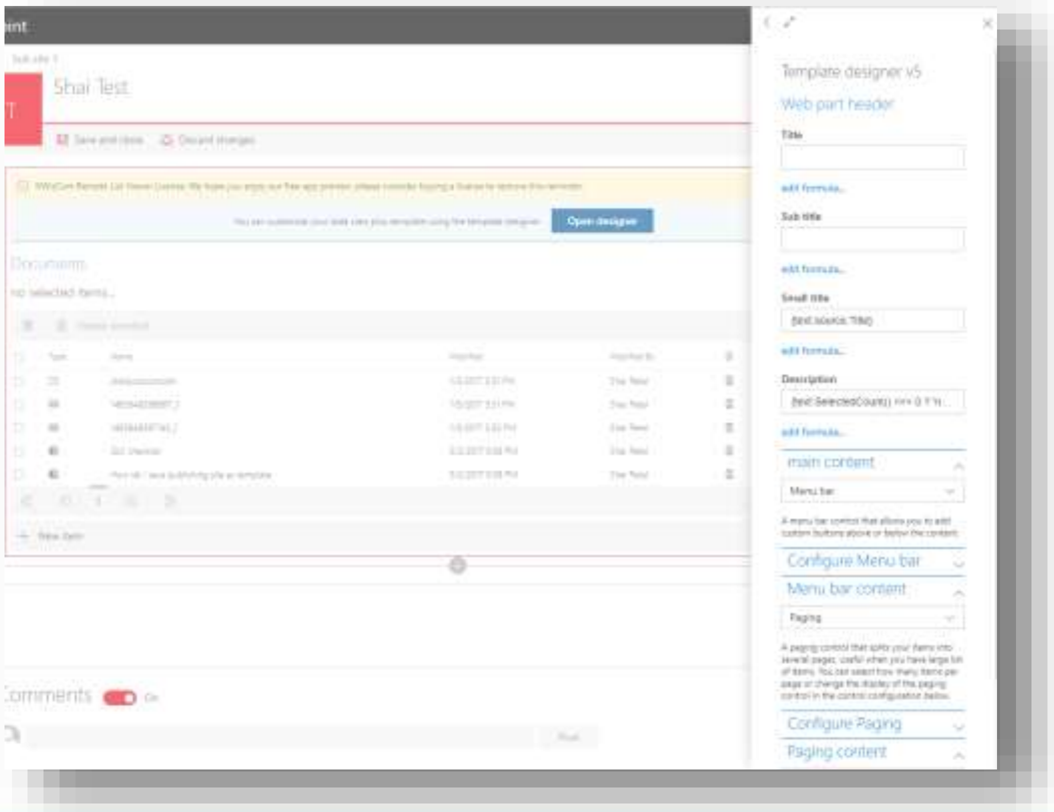
Advanced PropertyPane

- “this.properties” will get serialized and saved when your web part is saved. No matter how or what changed its value.

- Write your own designer logic: popup, panels or inline. Show it when your web part is in edit mode: if (this.displayMode === core.DisplayMode.Edit)

Advanced PropertyPane

Example: custom designer



How to get updates to SPFx?

UPDATING SPFX FRAMEWORK

Updating SPFx framework

Upgrading SPFx version is a challenge

- Pre-GA, it meant building a new project and moving your code
- Promised this will stop after GA *

Updating SPFx framework

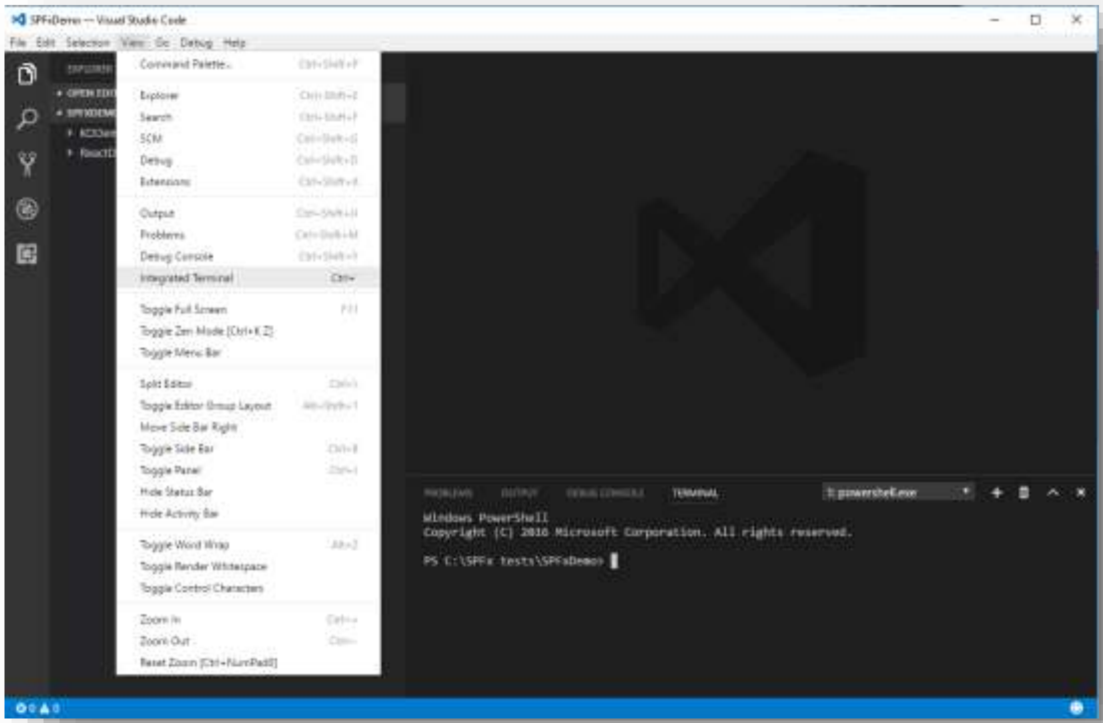
Dependencies

- In theory, as simple as updating package.json and running npm install/update
- In practice, involves a lot of praying, occasionally deleting node_modules, and dealing with unexpected errors in dependencies
- Some dependencies are still global, but that will pass

OTHER HELPFUL TIPS

Helpful tips

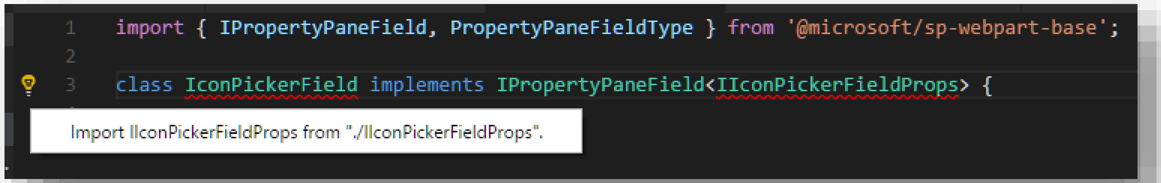
Open integrated terminal in VSCode



```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\SPFx tests\SPFxDemo>
```

Add import statements

You don't have to type the import yourself, look for the yellow bulb icon when standing on an unrecognized element:



```
import { IPropertyPaneField, PropertyPaneFieldType } from '@microsoft/sp-webpart-base';

class IconPickerField implements IPropertyPaneField<IIconPickerFieldProps> {
```

Import IIconPickerFieldProps from "./IIconPickerFieldProps".

Helpful tips

Delete/exclude node_modules from your source control / when zipping the project.

Demo project size:

646MB, 86K files.

Without node_modules:

1.75MB, 138 files.

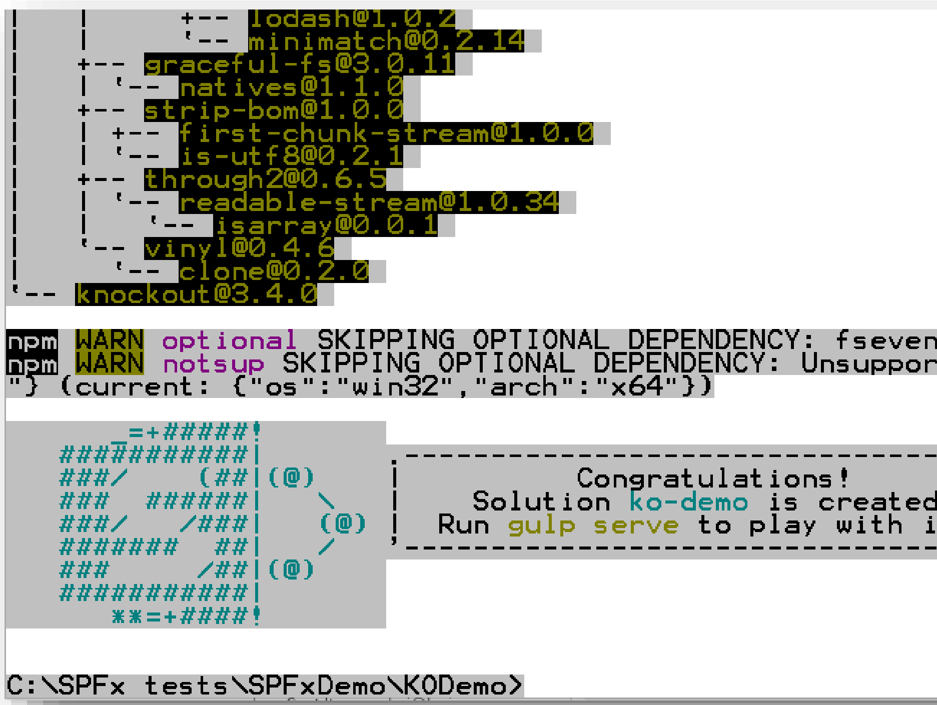
Helpful tips

Get used to memorizing commands…

- gulp = build
- gulp --ship = build in production
- gulp trust-dev-cert/untrust-dev-cert = trust the local certificate
- gulp package-solution = create package
- gulp serve = start local server
- code . = open VSCode in current folder

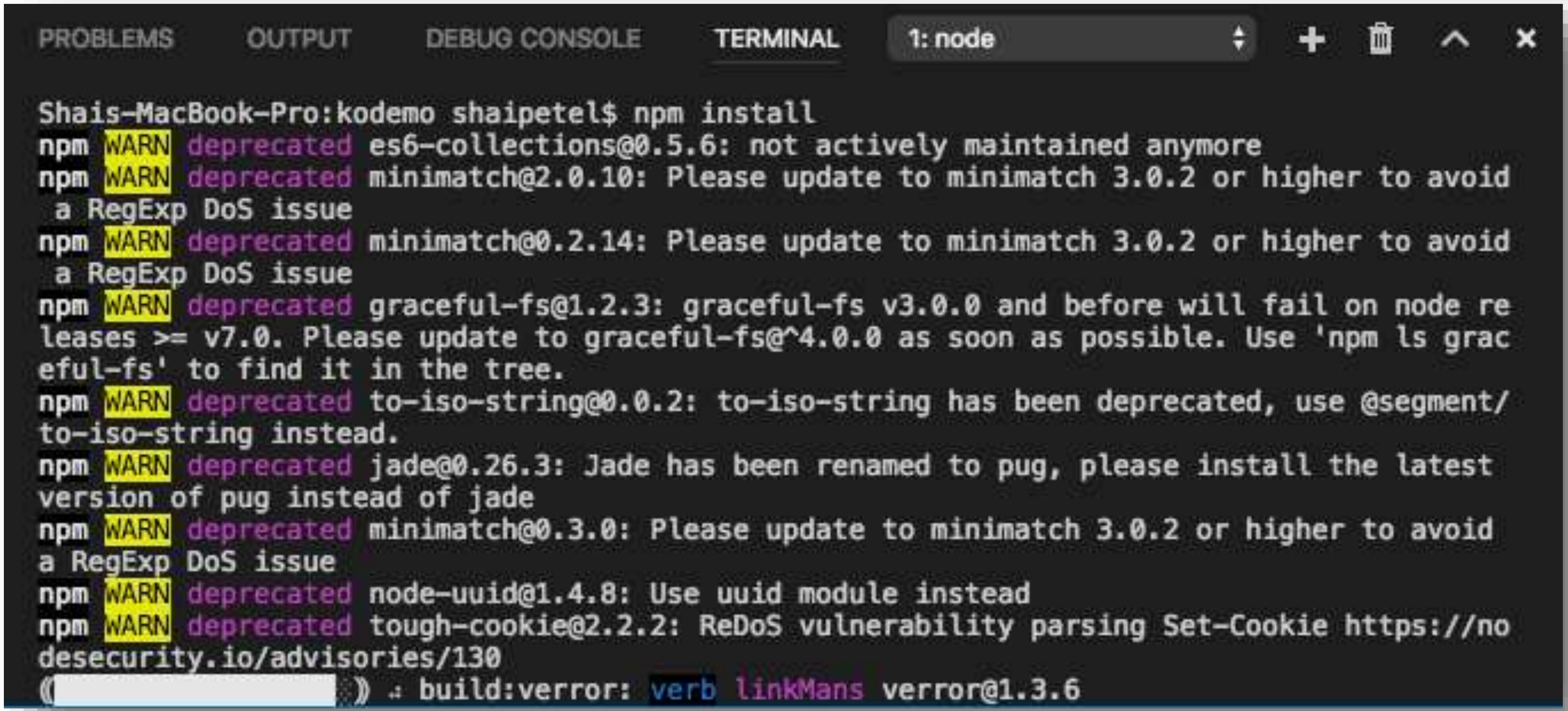
Helpful tips

- Get used to seeing warnings in the output of the different commands, especially npm-install.
- Errors – you should resolve or something will be broken.
- Warnings during build (gulp) may come from your code so you should fix these (missing semicolon, unneeded import, etc)



```
|     |       +-- lodash@1.0.2
|     |       `-- minimatch@0.2.14
|     +-- graceful-fs@3.0.11
|     | `-- natives@1.1.0
|     +-- strip-bom@1.0.0
|     | +-- first-chunk-stream@1.0.0
|     | `-- is-utf8@0.2.1
|     +-- through2@0.6.5
|     | `-- readable-stream@1.0.34
|     |   `-- isarray@0.0.1
|     `-- vinyl@0.4.6
|       `-- clone@0.2.0
`-- knockout@3.4.0

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fseven
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsuppor
"} (current: {"os":"win32","arch":"x64"})

       _=+#####!
   ###########|
   ###/    (##|(@)
   ###  ######|   \
   ###/   /###|    (@)   | ----------------------------------
   #######   ##|   /     |          Congratulations!
   ###     /##|(@)       |   Solution ko-demo is created
   ###########|          |  Run gulp serve to play with i
      **=+####!           ----------------------------------

C:\SPFx tests\SPFxDemo\KODemo>
```

Helpful tips



```
Shais-MacBook-Pro:kodemo shaipetel$ npm install
npm WARN deprecated es6-collections@0.5.6: not actively maintained anymore
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid
 a RegExp DoS issue
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid
 a RegExp DoS issue
npm WARN deprecated graceful-fs@1.2.3: graceful-fs v3.0.0 and before will fail on node re
leases >= v7.0. Please update to graceful-fs@^4.0.0 as soon as possible. Use 'npm ls grac
eful-fs' to find it in the tree.
npm WARN deprecated to-iso-string@0.0.2: to-iso-string has been deprecated, use @segment/
to-iso-string instead.
npm WARN deprecated jade@0.26.3: Jade has been renamed to pug, please install the latest
version of pug instead of jade
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid
a RegExp DoS issue
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie https://no
desecurity.io/advisories/130
(████████████████  ) ⠴ build:verror: verb linkMans verror@1.3.6
```

follow up:

shai@kwizcom.com

kwizcom.blogspot.com

QUESTIONS?