Introduction to the SharePoint Framework

Bob German
Principal Architect - BlueMetal
Bob German

Bob is a Principal Architect at BlueMetal, where he leads Office 365 and SharePoint development for enterprise customers.
Agenda

• A Geological View of SharePoint
• Developing “from scratch” for the SharePoint Framework
• Client-Side Solutions
  • Web Parts
  • Single Page Apps
  • List Apps
• Action Plan and Resources
The New SharePoint
A Geological View of SharePoint

- Add-in Model
- JSLink, Display Templates
- Custom Actions (JavaScript)
- XSLT
- SharePoint Page Models
- ASP.NET AJAX, Scripting On Demand
- ASP.NET WebForms
SharePoint Framework (SPFx)

• New client-side page and web part model
• Modern and responsive – allows “mobile first” development for SharePoint
• SPFx web parts work on both classic and client-side pages
• It’s still early! Open questions:
  • How will branding work?
  • How flexible will the page layout be?
  • What additional web parts will be available?
Microsoft moved the cheese... again!

SharePoint Framework

JavaScript and TypeScript Widgets

Add-in Model

Farm and Sandboxed Solutions

SharePoint Developer
## Comparing the Models

<table>
<thead>
<tr>
<th>Feature</th>
<th>Add-in Models</th>
<th>SP Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile-ready</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Direct access to page (e.g. Connected Web Parts)</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Ease of Data Access</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Ease of Upgrade (of your solution)</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Protection from Untrusted Code</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Storefront Distribution</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>
## Open Source Tools

<table>
<thead>
<tr>
<th></th>
<th>Trad</th>
<th>New</th>
<th>Link</th>
</tr>
</thead>
</table>

Links are case sensitive!
SP Framework Development Process

Project Folder

/src

JavaScript Bundles

/dest

/sharepoint

/spapp

Local workbench

workbench.aspx in SharePoint

Content Delivery Network

SharePoint Client-side “App”

Deployed in SharePoint
Don’t Panic!

What you are about to see is everyday development for open source web developers.

Have you noticed? Microsoft is going open source!
yo sharepoint

Part 1:
• Creating a project
• Using the SharePoint Workbench
yo sharepoint

Part 2:
• Packaging and Uploading to SharePoint
• Viewing on a Classic Page
yo sharepoint

Part 3:
• Deploying to a CDN
Now ... where is the web part?

localhost  SharePoint Online  Azure CDN
Splitting and Bundling

• Freely split your code into multiple files
• Each is translated to JavaScript and placed in a module based on the filename
  • TypeScript is compiled to JavaScript
  • Less is compiled to CSS in a JavaScript string
  • HTML becomes a JavaScript string
  • JavaScript is just placed in the module

• Use `exports` and `require` to expose and consume across modules (CommonJS conventions)

• Advantages
  • Ease of development / modularity
  • Efficient download and caching
  • Synchronous access (never wait for a script to load!)
Splitting and Bundling

Source Files

- .ts
- .html
- .js
- .less

Compilers (.ts → .js, .less → .css)

Webpack

Gulp task runner

Javascript bundle

0: javascript
1: javascript
2: javascript (html in string)
3: javascript
4: javascript (css in string)
A Brief Introduction to TypeScript
What is Typescript?

Typescript is a subset of JavaScript, and is “transpiled” into regular JavaScript.

• Static type checking catches errors earlier
• Intellisense
• Use ES6 features in ES3, ES5 (or at least get compatibility checking)
• Class structure familiar to .NET programmers

```javascript
let x = 5;
for (let x=1; x<10; x++) {
    console.log ('x is ' + x.toString());
}
console.log ('In the end, x is ' + x.toString());

var x = -1;
for (var x_1 = 1; x_1 < 10; x_1++) {
    console.log("x is "+x_1.toString());
}
console.log("In the end, x is "+x.toString()); // -1
```
## (1) Type Annotations

- **Intrinsics**
  - var myString: string = '<u>Results</u><br />';  
  - var myNum: number = 123;

- **Any and Casting**
  - var myAny: any = "Hello";
  - myString = <number>myAny;

- **Built-in objects**
  - var myDate: Date = new Date;
  - var myElement: HTMLElement = document.getElementsByTagName('body')[0];

- **Functions**
  - var myFunc: (x:number) => number = 
    function(x:number) : number { return x+1; }

- **Complex Types**
  - var myPeople: {firstName: string, lastName: string}[] =  
    [  
      { firstName: "Alice", lastName: "Grapes" },  
      { firstName: "Bob", lastName: "German" }  
    ]
var myFunc : (nx:number) => number = (n:number) => {
    return n+1;
}

let x:number = 1;
if (true) { let x:number = 100; }
if (x === 1) { alert ('It worked!') }

var target: string = 'world';
var greeting: string = `Hello, ${target}`;
interface IPerson {
    getName(): string;
}

class Person implements IPerson {
    private firstName: string;
    private lastName: string;
    constructor (firstName: string, lastName:string) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    getName() { return this.firstName + " " + this.lastName; }
}

var me: IPerson = new Person("Bob", "German");
var result = me.getName();
```typescript
var result: string = oldStuff.getMessage();

document.getElementById('output').innerHTML = result;
```

```javascript
var oldStuff = oldStuff || {};

oldStuff.getMessage = function() {
  var time = (new Date()).getHours();
  var message = "Good Day"
  if (time < 12) {
    message = "Good Morning"
  } else if (time < 18) {
    message = "Good Afternoon"
  } else {
    message = "Good Evening"
  }
  return message;
};
```

```typescript
interface IMessageGiver {
  getMessage(): string
}

declare var oldStuff: IMessageGiver;
```

```typescript
oldStuff.d.ts
(TypeScript Definition)
```

```javascript
myCode.ts
(TypeScript code wants to call legacy JavaScript)
```

```javascript
oldStuff.js
(legacy JavaScript)
```
Developing Client-Side Projects with SharePoint Framework
## Client-Side Projects

<table>
<thead>
<tr>
<th></th>
<th>Client-Side Web Parts</th>
<th>Run on classic and client-side pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Client-Side Apps (single page)</td>
<td>Your Single-Page application hosted in SharePoint</td>
</tr>
<tr>
<td>3</td>
<td>Client-Side Apps (list)</td>
<td>A SharePoint list with forms (AllItems, Display, Edit, New)</td>
</tr>
</tbody>
</table>
Weather Web Part
// Angular app is here: https://github.com/Widget-Wrangler/ww/tree/master/TypeScript.Weather

export class WeatherWebPart extends BaseClientSideWebPart<IWeatherWebPartProps> {
    // ...

    public render(mode: DisplayMode, data?: IWebPartData) {
        if (!this.renderedOnce) {
            this.domElement.innerHTML = viewHtml;
            angular.module('weatherWidget', [])
                .service('WeatherPropService', WeatherPropService)
                .service('WeatherService', WeatherService)
                .constant('appId', 'ecb1f756686518281c429bf5b7498d70')
                .controller('WeatherController', WeatherController);
            angular.bootstrap(this.domElement, ['weatherWidget']);
        }
        // Every time we render, update the web part properties
        angular.element(this.domElement).injector().get('WeatherPropService').UpdateProperties(this.properties);
    }
}
Client-Side Page Application
Client-Side List Application
So now what?

WTF! Is Microsoft serious?
When will it ship?
Where do I get it?
How do I get started?
Yes – Microsoft is serious about modernizing SharePoint!

Prediction: Over the next few years, Microsoft will change the entire SharePoint user experience to meet the needs of a modern, mobile workforce. It’s like upgrading a car, one part at a time – while you’re driving it cross-country!

Web Parts
O365 – mid 2016

New Pages
O365 – late 2016

SP2016 Feature Pack
early 2017

Cloud and Device Era

Web Era
Yes – Microsoft is serious about modernizing SharePoint!

The SharePoint Framework is additive!
• Classic SharePoint pages can run the new web parts
• Classic SharePoint sites can include the new full-page solutions

It should be possible to transition our sites gradually to the new model
### Tooling Up

<table>
<thead>
<tr>
<th>Toolkit</th>
<th>Description</th>
</tr>
</thead>
</table>

### Tutorials


Join me Wednesday at 4:00 for:  
Future-Proof Development for SharePoint and SharePoint Online
Thank you.