Introduction to Single Page Applications (SPA) Development Using Angular 2 and TypeScript

Trayan Iliev

IPT – Intellectual Products & Technologies
e-mail: tiliev@iproduct.org
web: http://www.iproduct.org

Oracle®, Java™ and JavaScript™ are trademarks or registered trademarks of Oracle and/or its affiliates.
Microsoft .NET, Visual Studio and Visual Studio Code are trademarks of Microsoft Corporation.
Other names may be trademarks of their respective owners.
Agenda I

1. Creating Angular 2 Hello World Application
2. Model-View-Controller (MVC), Model-View-Presenter (MVC), Model-View-ViewModel (MVVM) – MV* patterns
3. Web components
4. Data binding
5. Differences between Angular 1 and 2. Angular 2 advantages
7. Component controllers, views & templates
Agenda II

9. Using external template and style files
10. Using Angular Command Line Interface (CLI) to setup Angular 2 project and add application components
11. Ng2 by example – Tour of Heroes official Angular 2 tutorial
Where is The Code?

Angular 2 and TypeScript Web App Development code is available @GitHub:
https://github.com/iproduct/course-angular2
Creating Angular 2 Hello World Application

- 5 min Quickstart: https://angular.io/docs/ts/latest/quickstart.html
  - install **node** and **npm**
  - create an application project folder
  - add a **package.json** that defines the packages and scripts we need (or use **npm** init to initialize the project)
  - add a **tsconfig.json** to guide the TypeScript compiler
  - add a **typings.json** that identifies missing TypeScript definitions ← no longer needed for TypeScript 2 use **@types**
  - add a **systemjs.config.js** to configure System.js loader
  - install the npm packages and typings files
Angular 2 Hello World Project Structure

- angular2-quickstart
  - app
    - app.component.ts
    - app.module.ts
    - main.ts
  - node_modules ...
  - typings ...
    ← no longer needed for TS2, use @types
  - index.html
  - package.json
  - styles.css
  - systemjs.config.js
  - tsconfig.json
  - typings.json ← no longer needed for TS2, use @types
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: '<h2>My First Angular App</h2>'
})

export class AppComponent {
}
app/app.module.ts

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule {
}
```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { AppModule } from './app.module';

const platform = platformBrowserDynamic();
platform.bootstrapModule(AppModule);
index.html: Load App using SystemJS

```html
<html>
<head>
  <title>Angular 2 Demo 01</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="app/assets/css/main.css">
  <script src="node_modules/core-js/client/shim.min.js"></script>
  <script src="node_modules/zone.js/dist/zone.js"></script>
  <script src="node_modules/systemjs/dist/system.src.js"></script>
  <script src="systemjs.config.js"></script>
  <script>
    System.import('app').catch(function(err){ console.error(err); });
  </script>
</head>
<body>
  <my-app>Loading...</my-app>
</body>
</html>
```
Web Components (1)


- *They have to exist. Sales and marketing people are talking about them.* Components are not a technology. Technology people seem to find this hard to understand. Components are about how customers want to relate to software. They want to be able to buy their software a piece at a time, and to be able to upgrade it just like they can upgrade their stereo. They want new pieces to work seamlessly with their old pieces, and to be able to upgrade on their own schedule, not the manufacturer's schedule. They want to be able to mix and match pieces from various manufacturers. This is a very reasonable requirement. It is just hard to satisfy. - RalphJohnson
Web Components (2)

- Make it possible to build widgets …which can be reused reliably …and which won’t break pages if the next version of the component changes internal implementation details.


4 emerging W3C specifications:

- Custom elements – provide a way for authors to build their own fully-featured DOM elements.
- Template – declare fragments of HTML that can be cloned and inserted in the document by script
- HTML imports – `<link rel="import" href="my-custom-cmp.html">`
Web Components (3)

```html
<template id="custom-tag-tpl">
  <style>
    h1  { color: blue; }
  </style>
  <h1>My Custom Component</h1>
</template>

var CustomCmpProto = Object.create(HTMLElement.prototype);
CustomCmpProto.createdCallback = function() {
  var template = document.querySelector('#custom-tag-tpl');
  var clone = document.importNode(template.content, true);
  this.createShadowRoot().appendChild(clone);
};
var MyCmp = document.registerElement('custom-cmp', {prototype: CustomCmpProto});
document.body.appendChild(new MyCmp());
```
Data binding I

Source: AngularJS Developer Guide: https://docs.angularjs.org/guide/databinding
License: CC BY-3.0
Data binding II

One-Way Data Binding

- View
- Template
- Model

Two-Way Data Binding

- Template
- View
- Model

Source: AngularJS Developer Guide: https://docs.angularjs.org/guide/databinding
License: CC BY-3.0
Main Differences between Angular 1 and 2

- Component-based:
  - Angular 2 is component based.
  - controllers and $scope - no longer used
  - replaced by components and directives.
  - components are directives with a template

- Directives - specification for directives is considerably simplified, @Directive declarative annotation (decorator)

- Dependency Injection – 3 parts: the injector, bindings, and actual dependencies to be injected

- Forms and Validations – NgModel, NgForm, NgFormGroup, FormControlDirective, FormGroupDirective, FormArrayName, FormBuilder, RequiredValidator, PatternValidator, etc.
Advantages of Angular 2


- **Speed** – Angular 2 is dramatically faster than Angular 1 with support for fast initial loads through server-side pre-rendering, offline compile for fast startup, and ultrafast change detection and view caching for smooth virtual scrolling and snappy view transitions.

- **Browsers** – Angular 2 supports IE 9, 10, 11, Microsoft Edge, Safari, Firefox, Chrome, Mobile Safari, and Android 4.1+.

- **Cross-platform** – By learning Angular 2, you'll gain the core knowledge you'll need to build for a full range of platforms including desktop and mobile web, hybrid and native UI mobile installed apps, and even installable desktop applications.
Angular 2 Developers Guide

Available at:
https://angular.io/docs/ts/latest/guide/

- Data architecture in Angular 2 – overview, main types of components: Module, Component, Template, Metadata, Data Binding, Service, Directive, Dependency Injection.
- Component controllers, views & templates
- Using external template and style files.
- Ng2 by example – Tour of Heroes official Angular 2 tutorial
Angular 2 Data Architecture

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Example: app.module.ts

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  imports:      [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})

export class AppModule { }
```
Angular2 Modules using @NgModule Decorator

- **declarations** - the view classes that belong to this module (components, directives, and pipes).

- **exports** - the subset of declarations that should be visible and usable in the component templates of other modules.

- **imports** - other modules whose exported classes are needed by component templates declared in this module.

- **providers** - creators of services that this module contributes to the global collection of services.

- **bootstrap** - the main application view, called the root component, that hosts all other app views (root module only).

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Components (View Models) & Templates (Views)

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Example: app.component.ts

```typescript
import { Component } from '@angular/core';
@Component({
  selector: 'my-app',
  template: `  
    <h1>{{title}}</h1>
    <h2>My favorite hero is: {{myHero}}</h2>
  `
})
export class AppComponent {
  title = 'Tour of Heroes';
  myHero = 'Windstorm';
}
```

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Angular 2 Data Binding Types

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Angular 2 Directives

- **Structural directives** – alter (add, remove replace) DOM elements – Example:

  `<li *ngFor="let hero of heroes"></li>`
  `<hero-detail *ngIf="selectedHero"></hero-detail>`

- **Attribute directives** – alter the appearance or behavior of an existing element – Example:

  `<input [(ngModel)]="hero.name">`
Services & Dependency Injection (DI)

- Dependency Injection (DI) using constructors
- Hierarchical dependency injectors – module or component provided services

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Angular 2 Features & Services

- **Animations** – animate component behavior
- **Change detection** – zones to intercept asynchronous activity
- **Events** – publishing and subscribing to events (EventEmitter)
- **Forms** – complex data entry with validation and dirty checking
- **HTTP client** – communicate with a server to get/update data
- **Lifecycle hooks** – tap into key lifetime events of a component,
- **Pipes** – transforming display data: `price | currency:'USD':true`
- **Router** – single-page application navigation
- **Testing** – running unit tests on Ng2 application components

Source: Angular 2 Developer Guide: Architecture overview
https://angular.io/docs/ts/latest/guide/architecture.html
License: CC BY 4.0.
Angular 2 Command Line Interface (CLI)  
[https://github.com/angular/angular/angular-cli]

npm install -g angular-cli

ng new PROJECT_NAME

cd PROJECT_NAME
ng serve

ng serve --port 4201 --live-reload-port 49153

Create Angular 2 components using CLI:

Component ng g component my-new-component
Directive ng g directive my-new-directive
Pipe ng g pipe my-new-pipe
Service ng g service my-new-service
Thanks for Your Attention!

Questions?