Angular 2 - Introducción

IIC3585 - JS Frameworks

Andrighetti, Butorovic, Cerda, Lucchini

Índice

Introducción:

1. Motivación

2. AngularJS vs Angular 2

3. Ventajas

4. Desventajas

Introducción

- Framework de JS
- Mantenido por Google
- 100% client-side
- 100% Javascript (o Typescript)
- Sistema de templates con Data Binding en dos direcciones.
- Angular 2 es **orientado a componentes**

Introducción:

Motivación para la creación de Angular2

1. Performance: Angular JS sufrió cambios para ajustarse a diversas situaciones, pero sufrió el desempeño.
2. La Web ha cambiado: cross-browser, ES6, Web Components (Custom Elements, HTML Imports, Template Element, Shadow DOM)
3. Mobile Devices: inclusión de soporte de dispositivos móviles

Introducción:

1. Motivación

2. AngularJS vs Angular 2

3. Ventajas

4. Desventajas

AngularJS vs Angular 2

Features

1. Angular 2 es más rápido y sencillo que Angular 1.
2. Se pueden actualizar sets de datos grandes con mínimo overhead.
3. Agiliza la carga inicial a través de server rendering.
4. Soporta las últimas versiones de los browsers y al mismo tiempo, soporta browsers antiguos (IE9+, Android 4.1+).
5. Es un framework multiplataforma.
6. Angular 2 está enfocado principalmente en apps móviles.
7. La estructura del código es más simple que la versión anterior.

Introducción:

Ventajas

- Mantiene responsivas aplicaciones con carga pesada.
- Usa rendering en el servidor para agilizar la visualización de vistas en móviles.
- Funciona bien con ES6 y otros lenguajes que compilan a JS.
- Usa inyección de dependencias para mantener una aplicación sin escribir demasiado código.
- Todo sigue el modelo de orientación a componentes.

Introducción:

1. Motivación

2. AngularJS vs Angular 2

3. Ventajas

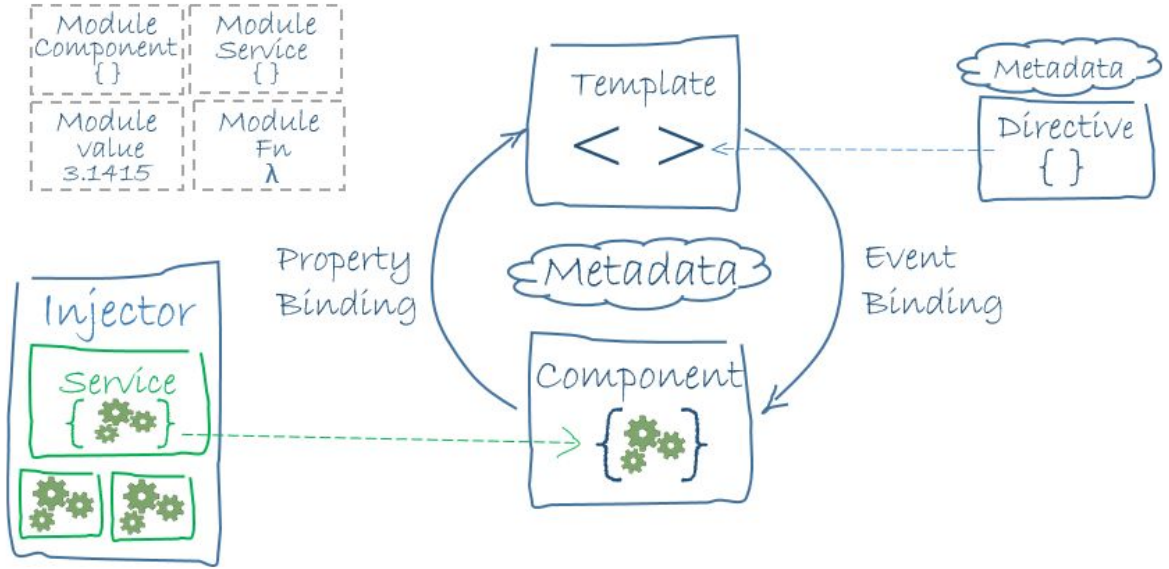
4. Desventajas

Desventajas

- Gran tiempo de aprendizaje
- No hay “type checking” en los templates
- Gran cantidad de comandos específicos para Angular 2.

Arquitectura

Bloques principales en la arquitectura de una app en Angular



Módulos

- Un módulo define un conjunto de archivos dedicados a un fin común.
- Clase adornada con la función decoradora @NgModule
- Fomenta el bajo acoplamiento y alta cohesión. Además, organizan app de gran complejidad, aumentando la escalabilidad.
- Mantiene una lista de los componentes pertinentes a su bloque de funcionalidad.

Componentes

- Código que maneja y está asociado a una vista.
- Se implementa como una clase adornada con la función decorador @Component.
- Interactúa con la vista a través de una API

app/app.component.ts

```
1. import { Component } from '@angular/core';
2.
3. @Component({
4.   selector: 'my-app',
5.   template: '<h1>My First Angular App</h1>'
6. })
7. export class AppComponent { }
```

Templates

- Acompaña a un componente para la definición de una vista
- Tiene código HTML principalmente, pero soporta sintáxis de Angular para vistas.

app/app.component.ts

```
1.  import { Component } from '@angular/core';
2.
3.  @Component({
4.    selector: 'my-app',
5.    template: '<h1>My First Angular App</h1>'
6.  })
7.  export class AppComponent { }
```

Metadata

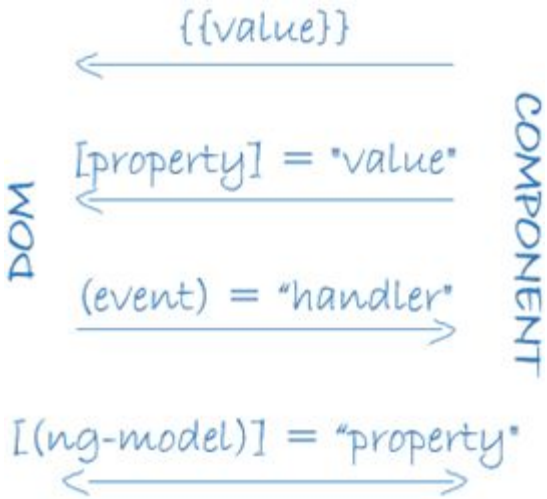
- El metadata le dice a angular como procesar una clase
- Ejemplos: @Component, @Injectable
- El decorador @Component asocia con al componente
- El decorador @Injectable permite inyectar dependencias
- Configuración: selector, template, moduleID, templateURL, style...

app/app.component.ts

```
1. import { Component } from '@angular/core';
2.
3. @Component({
4.   selector: 'my-app',
5.   template: '<h1>My First Angular App</h1>'
6. })
7. export class AppComponent { }
```

Data Binding

- Mecanismo de coordinación entre un template y su componente
- Tipos
  - Interpolación
  - Property Binding
  - Event Binding
  - Two-way binding, usa directive ngModel



Directivas

- Instrucciones especiales que permiten modificar dinámicamente el DOM.
- Suelen usarse en dentro del tag de un elemento HTML, como un attributo.
- Examples:

  *ngFor tells Angular to stamp out one <li> per element in the list.

  *ngIf includes the wrapping component only if the sentence is true.

```
3
4    <li *ngFor="let hero of heroes"></li>
5    <hero-detail *ngIf="selectedHero"></hero-detail>
6
```

Servicios

- Cualquier cosa necesitada por la aplicación; es un categoría amplia
- Ejemplos:
  i. Logging service
  ii. Data service
  iii. Message bus
  iv. Application configuration

Dependency Injection

- Forma que tiene Angular 2 para suplir a una nueva instancia de una clase de las dependencias que necesita.
- Por lo tanto, ocurre al momento de crearse una clase. La clase le pregunta a un inyector por instancias de las dependencias que necesita.
- En caso de no estar, las crea en el momento.
- La mayoría son servicios

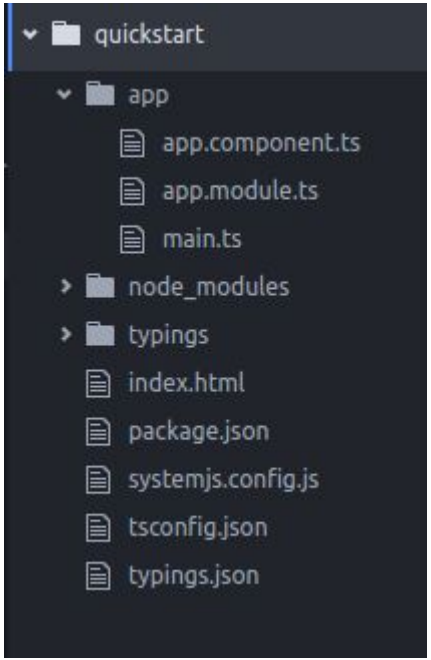
```
3   @Component({
4     moduleId: module.id,
5     selector:    'hero-list',
6     templateUrl: 'hero-list.component.html',
7     providers:  [ HeroService ]
8   })
9
```

1. Archivos de configuración

2. Carpeta node_modules

3. Carpeta app

Quickstart



Archivos de configuración

- package.json: identifica las dependencias npm del proyecto.
- tsconfig.json: define cómo el compilador de TypeScript genera JS.
- typings.json: provee información adicional para archivos que no reconoce nativamente el compilador de TS.
- systemjs.config.js: provee información a la carga de módulos sobre la localización de los módulos de la aplicación.

node_modules

La carpeta node_modules contiene las dependencias a diferentes librerías, incluyendo Angular.

Carpeta app

Contiene:

- app.component.ts: componente raíz o principal.
- app.module.ts: módulo principal o raíz.
- main.ts: le indica a Angular cómo comenzar la aplicación.

Routing

El router de Angular puede interpretar una URL del browser como una instrucción para navegar hacia una vista generada en el cliente y pasar parametros opcionales al componente para ayudarle a decidir que contenido mostrar

Routing

El router no es parte del core de Angular, está en su propio paquete, el import mínimo es:

```
import { Routes, RouterModule } from '@angular/router';
```

Esto debe ser importado tanto en el archivo de configuración como en el *app.module*

Además, en el </head> de nuestro *index.html*, debemos declarar la ruta base:

```
<base href="/">
```

Routing

1. El router de angular

2. Base href y router imports

3. El archivo de configuración

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const appRoutes: Routes = [
  { path: 'hero/:id', component: HeroDetailComponent },
  { path: 'crisis-center', component: CrisisCenterComponent },
  {
    path: 'heroes',
    component: HeroListComponent,
    data: {
      title: 'Heroes List'
    }
  },
  { path: '', component: HomeComponent },
  { path: '**', component: PageNotFoundComponent }
];

export const appRoutingProviders: any[] = [

];

export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

app.routing.ts

1. Imports

2. Routes Array

3. RouterModule.forRoot

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const appRoutes: Routes = [
  { path: 'hero/:id', component: HeroDetailComponent },
  { path: 'crisis-center', component: CrisisCenterComponent },
  {
    path: 'heroes',
    component: HeroListComponent,
    data: {
      title: 'Heroes List'
    }
  },
  { path: '', component: HomeComponent },
  { path: '**', component: PageNotFoundComponent }
];

export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

Routing

1. El router de angular

2. Base href y router imports

3. El archivo de configuración

```
import { ModuleWithProviders } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const appRoutes: Routes = [
  { path: 'hero/:id', component: HeroDetailComponent },
  { path: 'crisis-center', component: CrisisCenterComponent },
  {
    path: 'heroes',
    component: HeroListComponent,
    data: {
      title: 'Heroes List'
    }
  },
  { path: '', component: HomeComponent },
  { path: '**', component: PageNotFoundComponent }
];

export const appRoutingProviders: any[] = [

];

export const routing: ModuleWithProviders = RouterModule.forRoot(appRoutes);
```

HTTP

Cliente provisto por Angular para comunicarse mediante HTTP

GET, POST, PUT, PATCH, DELETE, ...

HTTP

Importar en AppModule:

```
import { HttpModule }    from '@angular/http';

imports: [
    // ...
    HttpModule
]
```

HTTP

1. Comunicación con servidores

2. Importar

3. Crear un servicio dedicado

4. Utilizar el servicio en el componente

hero.service.ts:

```
@Injectable()
export class HeroService {

    getHeroes(): Observable<Hero[]> {
        return this.http.get('app/heroes')
            .map(this.extractData)
            .catch(this.handleError);
    }

    private extractData(res: Response) {
        let body = res.json();
        return body.data || { };
    }

    private handleError (error: any) {
        // ...
    }
```

HTTP

hero-list.component.ts:

```
export class HeroListComponent implements OnInit {
    errorMessage: string;
    heroes: Hero[];

    constructor (private heroService: HeroService) {}
```

1. Comunicación con servidores

2. Importar

3. Crear un servicio dedicado

4. Utilizar el servicio en el componente

HTTP

```
ngOnInit() { this.getHeroes(); }

getHeroes() {
        this.heroService.getHeroes()
            .subscribe(
                heroes => this.heroes = heroes,
                error => this.errorMessage = <any>error);
}
```

Demo

(mostrar)

Apreciaciones

1. **Usar el proyecto de quickstart en github (base proyect)**
2. Learning Curve
3. Documentación
4. Actividad
5. Typescript

Apreciaciones

1. Usar el proyecto de quickstart en github (base proyect)
2. **Learning Curve**
3. Documentación
4. Actividad
5. Typescript

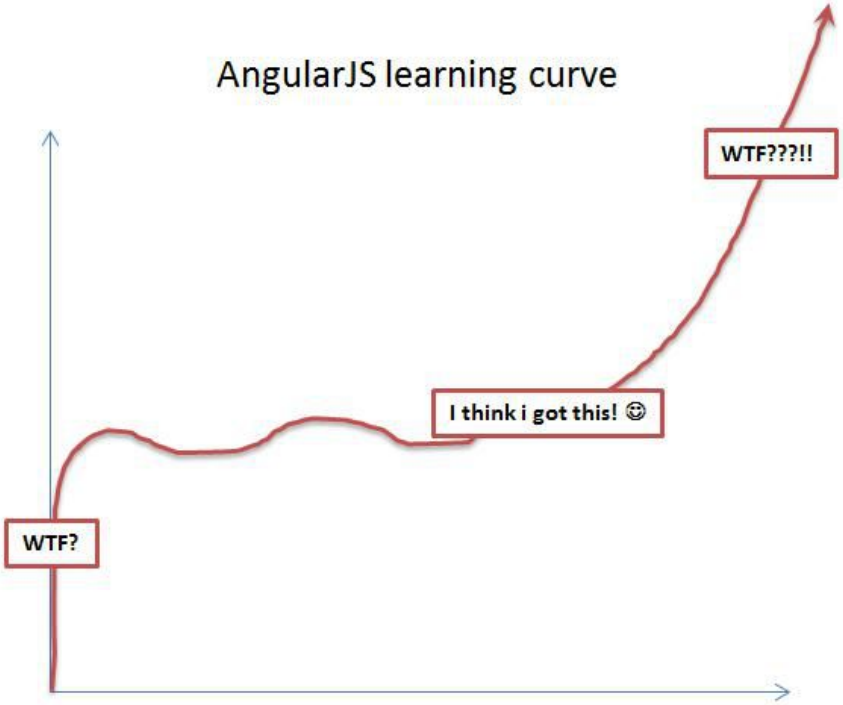
Apreciaciones y consejos

Curva de aprendizaje:

1. ¿Es mejor que Angular JS?

2. ¿Cómo es en comparación a otros Frameworks?

Learning Curve



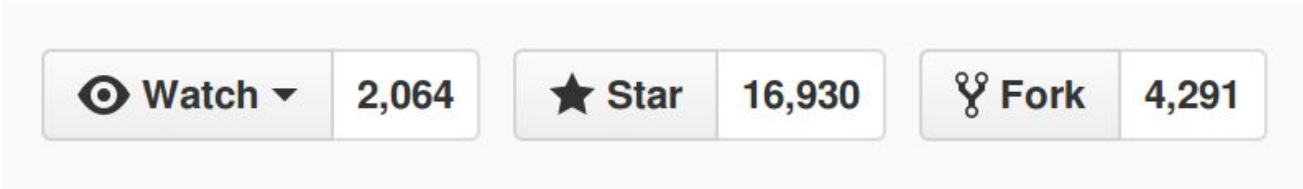
Apreciaciones

1. Usar el proyecto de quickstart en github (base proyect)
2. Learning Curve
3. **Documentación**
4. Actividad
5. Typescript

Apreciaciones

1. Usar el proyecto de quickstart en github (base proyect)
2. Learning Curve
3. Documentación
4. **Actividad**
5. Typescript

Actividad



Actividad hasta el 1 de Oct. 2016

Apreciaciones

1. Usar el proyecto de quickstart en github (base proyect)
2. Learning Curve
3. Documentación
4. Actividad
5. **Typescript**

Referencias

- http://eisenbergeffect.bluespire.com/all-about-angular-2-0/
- https://github.com/angular/quickstart
- https://www.tutorialspoint.com/angular2/index.htm
- https://github.com/angular
- https://angular.io/docs
- https://edm00se.io/web/angular-2-learning-curve/
- https://www.bennadel.com/blog/2439-my-experience-with-angularjs---the-super-heroic-javascript-mvw-framework.htm
- https://github.com/angular/angular.dart
- https://angular.io/docs/ts/latest/guide/
- https://angular.io/docs/ts/latest/tutorial/toh-pt5.html
- https://www.sitepoint.com/getting-started-with-angular-2-using-typescript/
- https://www.infoq.com/articles/Angular2-TypeScript-High-Level-Overview
- https://books.ninja-squad.com/public/samples/Become_a_ninja_with_Angular2_sample.html