# Notas Angular

## How Are They Different?

**1. Angular está orientado a móviles y tiene mejor rendimiento**

Angular 1.x no se hizo para soportar móviles, mientras que Angular está orientado desde el principio a dar buen rendimiento y funcionar bien en dispositivos móviles.

AngularJS se diseñó para crear apps Web [de tipo SPA](https://www.campusmvp.es/recursos/post/Video-que-son-las-Single-Page-Applications.aspx), con enlazado a datos bidireccional. No había soporte para móviles, aunque sí existen otras bibliotecas que hacen que Angular 1.x se ejecute en móviles.

**2. Angular ofrece más opciones a la hora de elegir lenguajes**

Angular ofrece la posibilidad de elegir entre varios lenguajes a la hora de programar. Para escribir código Angular puedes usar cualquiera de los lenguajes: ECMAScript 5 de toda la vida, ES6, TypeScript o incluso Dart (de Google). Mientras que con Angular 1.x puedes usar ES5, ES6 y Dart.

Poder usar **TypeScript** es un gran avance ya que [**es una forma genial de escribir JavaScript**](https://www.campusmvp.es/recursos/post/por-que-aprender-typescript.aspx). TypeScript es el lenguaje por defecto para desarrollar en Angular, y el que sin duda vas a querer utilizar. La mayor parte la documentación que encontrarás por ahí estará creada con TypeScript, así que debes aprenderlo.

**3. Los controladores y el $scope de Angular 1.x han desaparecido**

Podemos decir que en Angular los controladores se substituyen por **componentes**. Angular se basa en componentes web, con las ventajas que ello supone al adoptar un estándar de futuro, que cuando esté bien soportado por todos los navegadores ofrecerá más rendimiento todavía.

**4. La sintaxis de las directivas estructurales ha cambiado**

En Angular, la sintaxis de las directivas estructurales ha cambiado, ng-repeat se sustituye por \*ngFor, por ejemplo.

**Directivas estructurales Angular 1.x:**

<ul>

<li ng-repeat="prod in productos">

{{prod.name}}

</li>

</ul>

**Directivas estructurales Angular:**

<ul>

<li \*ngFor="prod of productos">

{{prod.name}}

</li>

</ul>

El signo Asterico(\*) se usa como prefijo para directivas estructurales, in se sustituye por of y se usa la sintaxis *camelCase*. Hay muchos más detalles de esta nueva sintaxis, pero de momento quédate con esto.

**5. Angular usa directamente las propiedades de los elementos y los eventos estándar del DOM**

Uno de los mayores cambios en Angular es, que usa directamente las propiedades de los elementos y los eventos estándar del DOM.

Por ello, muchas de las directivas integradas disponibles en Angular 1.x ya no se necesitan, como por ejemplo: ng-href, ng-src, ng-show o ng-hide. Angular usa directamente href, src y propiedades hidden para obtener el mismo resultado.

Y lo mismo se puede decir de las directivas basadas en eventos como ng-click o ng-blur.

En AngularJS:

<button ng-click="doSomething()">

En Angular simplemente tomas el evento estándar y lo envuelves entre paréntesis:

<button (click)="doSomething()">

Nuevamente aquí hay muchos otros detalles a tener en cuenta, pero quédate con esta idea principal.

**6. La directiva de datos enlazados en una sola dirección (*one-way data binding*) se sustituye por [property]**

En Angular 1.x, ng-bind se usa para enlazar datos en una sola dirección (*one-way data binding*), lo que quiere decir que sólo se modifica el enlace desde el código hacia la vista, pero no al revés, lo que permite un mayor control de flujo dentro de la aplicación.

Con Angular esto se reemplaza por [property], siendo "property" una propiedad válida del elemento del DOM sobre el que actuamos.

Por ejemplo, en Angular 1.x escribíamos:

<input ng-bind="prod.name"></input>

En Angular se utilizan simplemente corchetes sobre la propiedad estándar:

<input [value]="prod.name"></input>

Aunque existen otras variantes para lograr lo mismo.

**7. Enlaces de datos de doble dirección (*two-way data binding*): ng-model se sustituye por [(ngModel)]**

Este es el enlazado que todo el mundo conoce y usa en AngularJS. En Angular se retira esta sintaxis para lograr mayor seguridad, control y mejora del rendimiento.

En Angular 1.x hacíamos esto para tener un enlazado en dos direcciones:

<input ng-model="prod.name"></input>

En Angular la sintaxis equivalente sería:

<input [(ngModel)]="prod.name"></input>

Este *doble-binding* ofrece ventajas relevantes en la gestión avanzada de formularios.

**8. Ha cambiado la sintaxis de la inyección de dependencias**

Una de las grandes ventajas de Angular es la inyección de dependencias. Con Angular hay una manera distinta de llevar a cabo esto. Como en Angular todo son "clases", la inyección de dependencias se consigue mediante constructores.

Link: <https://www.campusmvp.es/recursos/post/las-10-principales-diferencias-entre-angularjs-y-angular.aspx>

**Performance**

AngularJS was originally developed for designers, not developers. Although there were a few evolutionary improvements in its design, they were not enough to fulfill developer requirements. The later versions, Angular 2 and Angular 4, have been upgraded to provide an overall improvement in performance, especially in speed and dependency injection.

**1. Speed**

By providing features like 2-way binding, AngularJS reduced the development effort and time. However, by creating more processing on the client side, page load was taking considerable time. Angular2 provides a better structure to more easily create and maintain big applications and a better change detection mechanism. Angular 4 is the fastest version yet.

**2. Dependency injection**

Angular implements unidirectional tree-based change detection and uses Hierarchical Dependency Injection system. This significantly boosts performance for the framework.

# Angular's template syntax

Common features of Angular's template syntax:

* \*[ngFor](https://angular.io/api/common/NgForOf) Structural directives are responsible for HTML layout. They shape or reshape the DOM's *structure*, typically by adding, removing, or manipulating elements. Structural directives are easy to recognize. An asterisk (**\***) precedes the directive attribute name as in this example.
* \*[ngIf](https://angular.io/api/common/NgIf)
* Interpolation {{ }} Para acceder a información del controlador y mostrarla como texto.
* Property binding [ ] Acceder a una propiedad del HTML y añadirle información del controlador.
* Event binding ( ) Para a un evento añadirle la función a llamar en la controladora

<https://angular.io/start>

# Instalación

<https://openwebinars.net/blog/instalacion-angular-8-requisitos/>

<https://www.agiratech.com/best-angular-code-editors-ide/>

**Instalacion angular y mi primera app**

https://www.c-sharpcorner.com/article/three-steps-to-install-angular-and-create-first-hello-world-angular-app/

## Comandos CMD para angular

node -v -> Para obtener la verison actual de node

npm -vdi-r-pena -> para obtener la version actual de npm

npm install -g typescript -> para instalar typescript

npm install -g @angular/cli -> instalar angular

npm install --save [bootstrap@3](mailto:bootstrap@3) → let you install boostrap version 3 to your project

**Angular commands:**

|  |  |
| --- | --- |
| Commands | Explenation |
| **ng --version** | To know the version |
| **ng new my-firt-app** | for create a new project |
| **ng serve** | For compile and execute the application. Before that you should be located in the directory of the app files are. Any change over the app will be refresh automatically. |
| **ng generate component name** | Command to generate a component automatically inside on the app folder. Also can be used the abbreviation **ng g c name** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Errors in Angular installation**

https://stackoverflow.com/questions/46623571/angular-ng-command-not-found/46623602

## Visual Studio Code

**Extensions to add**

This are good extension to add to the visual studio code IDE when you are working with angular.

<https://medium.com/frontend-coach/7-must-have-visual-studio-code-extensions-for-angular-af9c476147fd>

TODO: take a look, and test

# Course Angular

## Session 1

**Forum of the course**

As a student of this course you also get free access to our “Academind Community” on Discord: <https://discord.gg/gxvEWGU>

There, you can find like-minded people, discuss issues, help each other, share progress, successes and ideas and simply have a good time!

Note: my logging it the [leandrod.pty@gmail.com](mailto:leandrod.pty@gmail.com) account and password.

General Notes

* Angular its is SPA.
* Angular have a new version over 6 month, but all of they are compatible.
* If you want to dive deeper into the CLI and learn more about its usage, have a look at its official documentation: <https://github.com/angular/angular-cli/wiki>
* **app.module.t**s: In this file is where we tell to Angular the modules or package that going to be used by the app.
* **node\_modules**: in this directory inside every project the nmp dependencies installed will be. For example the version of typescroopt, bootstrap, etc. All that we install with NPM.
* **angular.json**: in this file is where all the configuration of the project will be. For example in this file we add the the specific version of bootstrap we going to use. The name of the project, etc.
* Start flow.

1. Angular execution start from the main.ts file where its passed as parameter the AppModule to platformBrowserDynamic().bootstrapModule(AppModule).

2. The file app.module.ts is imported in main.ts and paseed. In app.module.ts we inject the component AppComponent to Angular.

3. Then angular analyze it the component AppComponent and all its files: .html, .css, .ts.

4. Now the index.html file how is the main html can draw the component AppComponent.

<body>

  <app-root></app-root>

</body>

AppComponent have in its app.component.html file son html to be embedded in the index.html.

* Angular in the end is a Js framework changing your DOM(HTML) at run time.

## Session 2

### Components

* Angular is based on component with its own html, css if needed and ts file with the logic.
* Component .ts is not more than a typescript class that should have the name export. The export name let use this component, and be injected.
* After you create your component in its folder with its file, angular don't scant the component. You have to associate, register this component into a module. An angular module is used to encapsulate, group together different pieces like for example components into packages. For the most apps the app module will be enough.
* **ng generate component name** → command to generate a component automatically inside on the app folder. Also can be used the abbreviation **ng g c name**
* If you want in a component you can put instead of a **templateUrl** to a file, put the word **template,** and you are creating an inline template. Also if you want to put a lot of lines you have to use instead of quotes (‘’) the symbols (``).
* Just like template, for css we can do the same, we can use styleUrls for add in the line styles of css, instead of use styleUrls for adding the styles files.
* Selector. The selector function like a css styles. If you put the selector inside [] your directive will be used as a propertiy of an html tag. Also if a dot is used before the name of the directive, then the directive will be used as an html class.

### Databinding

its the communication between the typescript code (business logic) and Template (html) code.

**String interpolation:**

- When you put a variable or function front the logic (ts) between {{}} in the html template. Also expression could be written between the {{}}.

- At the end this element will be converted to string, angular will do it for you if the value can be converted to string.

Example:

<h4 class="list-group-item-heading">{{recipe.name}}</h4>

<img [src]="recipe.imagePath"

                     alt="{{recipe.imagePath}}"

                     class="img-responsive" style="max-height: 50px;">

**Property binding**

- When we put an html property between [] we are telling that this property will be binding and controlled by angular. Then we assign the value of a variable inside double quotes “”. Example:

<button

    class="btn btn-primary"

    [disabled]="allowNewServer">Add server</button>

**Even Biding**

- We have to put the even between () and also write the even clear, without prepositions. Then the assignation and double quotes “”. Example:

<button

    class="btn btn-primary"

    [disabled]="!allowNewServer"

    (click)="onServerCreation()">Add server</button>

**Bindable Properties and Events**

How do you know to which Properties or Events of HTML Elements you may bind? You can basically bind to all Properties and Events - a good idea is to console.log() the element you're interested in to see which properties and events it offers.

Important: For events, you don't bind to onclick but only to click (=> (click)).

The MDN (Mozilla Developer Network) offers nice lists of all properties and events of the element you're interested in. Googling for YOUR\_ELEMENT properties or YOUR\_ELEMENT events should yield nice results.

**Passing and Using Data with Event Binding**

How to get the even data of an HTML element ? If to an event of an input, for example we hava (clic)= "onServerFunction($event)"

If we pass the variable $event, we are sending to the TS the data and event contained in the object, in this case an input.

**32. Two-Way-Databinding**

Using two ways of data binding the value of an input for example associated to a variable in the TS (serverName in this example), any change in the TS is reflected in the template and vice versa.

Example:

<input type=”text” [(ngModel)]=”serverName”>

**Important: FormsModule is Required for Two-Way-Binding!**

Important: For Two-Way-Binding (covered in the next lecture) to work, you need to enable the ngModel  directive. This is done by adding the FormsModule  to the imports[]  array in the AppModule.

You then also need to add the import from @angular/forms  in the app.module.ts file:

import { FormsModule } from '@angular/forms';

### Understanding Directives

**Using ngIf to Output Data Conditionally**

Ngif is and structural directive, so should be used with asterisk before, this is because structural directives modify the DOM. If the value its true the element is draw in this example, else it not draw.

Example:

<p \*ngIf="serverCreated">Server was created, name is {{ serverName }}</p>

**Enhancing ngIf with an Else Condition**

After the \*ngIf you can put ; else, and do something in the other case.

For example:

<p \*ngIf="serverCreated ; else noServer">Server was created, server name is {{ serverName }}</p>

<ng-template #noServer>

    <p>No server was created!</p>

</ng-template>

<https://ultimatecourses.com/blog/angular-ngif-else-then>

**Styling Elements Dynamically with ngStyle**

The directive that are not of type structural directive are called attribute directive. They don’t add or remove elements, they only change the element they were placed on.

ngStyle let you change the style of an object dynamically.

Example:

<p [ngStyle]="{'background-color': getColor(1), 'color': getColor(2)}">{{numClick}}</p>

**Applying CSS Classes Dynamically with ngClass**

On the other hand ngClass let you add or remove dynamically a CSS class to an object if a certain condition is true. When we call it we have to pass a key:value, the key it’s the class name we going to add and the value is the condition determining whatever this class should be showed or not.

Example:

<p [ngClass]="{hidding : displayDetailsParagraph == true}">'Secret password = tuna'</p>

**Outputting Lists with ngFor**

\*ngFor it’s a structural directive that let you add elements to the DOM dynamically, just like a FOR of any language.

Example:

<div \*ngFor="let numClick of numClicks">

    <p>{{numClick}}</p>

</div>

if we want access to the index, current index in the for we can declare a variable an assing the reserved word in this context “index”:

<div \*ngFor="let numClick of numClicks; let i = index"

    [ngStyle]="{'background-color': getColor()}"

    [ngClass]="{'letters-color': i > 5}">{{numClick}}

</div>

## Session 3

### Components

# TypeScript

## Classes

**Constructor**

In TS the traditional way of declare a class and its constructor it is:

export class Ingrediente{

    public name: string;

    public amount: number;

    constructor(name: string, amount: number) {

        this.name = name;

        this.amount = amount;

    }

}

However there is a short way of declare the class with the attributes and it is:

export class Ingrediente{

    constructor(public name: string, public amount: number) {}

}

At the end TS compiler going to convert this second way to the first one.

# Various

## Sobre REACT

npm install → will get everything set up, and running

npm start → will let you run the application live.

React and Angular together

1. In this two pages are explained how integrate two applications of different frameworks through the NX framework.

<https://blog.nrwl.io/building-angular-and-react-applications-together-with-nx-78b5578de598>

<https://medium.com/angular-in-depth/how-to-talk-with-web-components-in-react-and-angular-8deb7d2fb92a>

2. In this other example, that seems to be what we need, the integration is done in a SPA using Micro Frontends integration.

<https://ivanjov.com/micro-frontends-how-i-built-a-spa-with-angular-and-react/>

Important notes:

package.json In this file are all dependencies need for React, Angular and other dependencies used to easy the integration and the communication.

root-application.js In this js, we're importing index files from react and angular directories and we initialize them on the page when the router hash starts with.

event-bus/index.js The communication between React and Angular apps can be tricky. It is done with Eev event bus librery. It's small, fast and zero-dependency event emitter that will help us to exchange information between React and Angular app.

react/index.js This is where the export of the React child app is done. In this code the React module single-spa-react is used to tell single-spa how to bootstrap, mount and unmount the React app.

angular/index.js In the folder angular it is where a component angular is created. In the file index.js , same as React, this code will tell single-spa how to bootstrap, mount and unmount the Angular app. The angular module single-spa-angular2 is used for that.

For execution: fto start the project first execute: npm install and then npm start. The url its: http://localhost:9090/#/

## Possible improvement for Angular 9

<https://profile.es/blog/angular9/>