

101 Panel Tech Days

Micael Gallego Carrillo
24/11/2016



Angular 2

para backend developers

Micael Gallego Carrillo
24/11/2016

panel.es
Panel Sistemas Informáticos, S.L.
Consultoría, servicios y soluciones TI.



¿Quién soy?



@micael_gallego

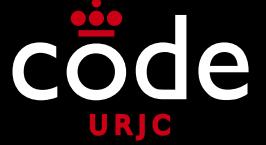


micael.gallego@urjc.es



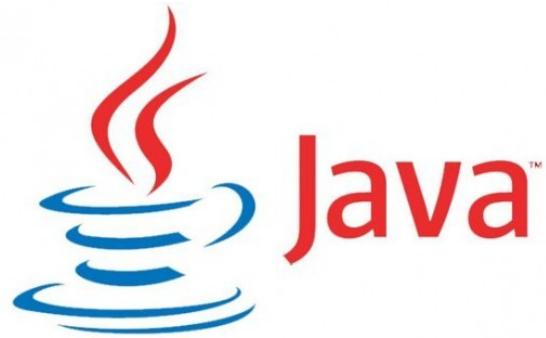
@micaelgallego

¿Quién soy?



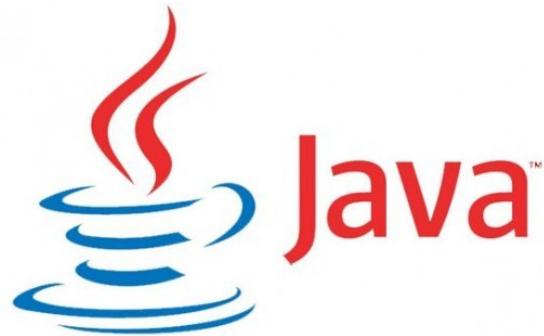
developer

¿Quién soy?



developer

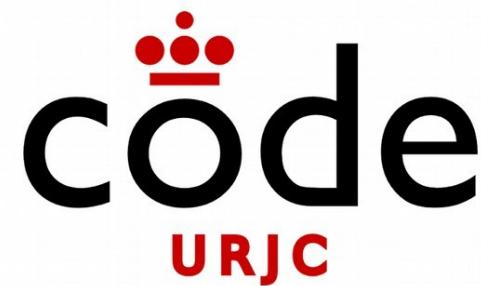
¿Quién soy?



developer



Universidad
Rey Juan Carlos



¿Quién soy?



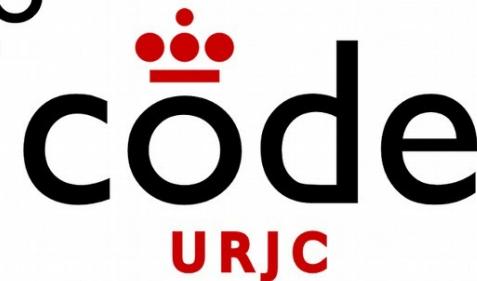
developer

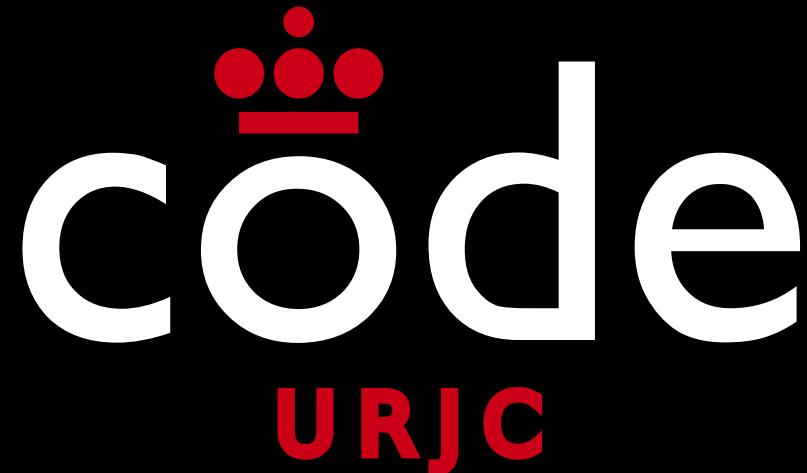


Universidad
Rey Juan Carlos



La Escuela de Negocios de la
Innovación y los Emprendedores





Consultoría y Formación en Desarrollo Software

**Contacta con nosotros para cursos
presenciales, online, in company**

Taxonomía de las webs

Existen varios tipos de webs

Dependiendo de cómo usan las tecnologías
de **cliente** y **servidor**

Página web

Servidor estático

El servidor web sirve
contenido guardado en el
disco duro

Aplicación web

Servidor dinámico

El servidor web sirve
contenido generado
mediante código

Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

Página web

Servidor estático

- Página web estática

- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

java.util.concurrent (Java I X)

download.java.net/jdk8/docs/api/

Java™ Platform Standard Ed. 8 DRAFT ea-b123

All Classes All Profiles Packages

java.applet
java.awt
java.awt.color

AnnotationValueVisitor
Any
AnyHolder
AnySeqHelper
AnySeqHelper
AnySeqHolder
AppConfigurationEntry
AppConfigurationEntry.LoginMo
Appendable
Applet
AppletContext
AppletInitializer
AppletStub
ApplicationException
Arc2D
Arc2D.Double
Arc2D.Float
Area
AreaAveragingScaleFilter
ARG_IN
ARG_INOUT
ARG_OUT
ArithmaticException
Array
Array
ArrayBlockingQueue

Please note that the specifications and other information contained herein are not final and are subject to change. The information is being made available to you solely for purpose of evaluation.

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java™ Platform Standard Ed. 8 DRAFT ea-b123

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package java.util.concurrent

Utility classes commonly useful in concurrent programming.

See: Description

Interface Summary

Interface	Description
BlockingDeque<E>	A Deque that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.
BlockingQueue<E>	A queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
Callable<V>	A task that returns a result and may throw an exception.
CompletableFuture.AsynchronousCompletionTask	A marker interface identifying asynchronous tasks produced by <code>async</code> methods.
CompletionService<V>	A service that decouples the production of new asynchronous tasks from the consumption of the results of completed tasks.

Maven – Maven in 5 Minu x

maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Apache Maven Project

http:// maven.apache.org

Apache > Maven > Maven in 5 Minutes

Last Published: 2015-01-11

Maven in 5 Minutes

Prerequisites

You must have an understanding of how to install software on your computer. If you do not know how to do this, please ask someone at your office, school, etc or pay someone to explain this to you. The Maven mailing lists are not the best place to ask for this advice.

Installation

Maven is a Java tool, so you must have Java installed in order to proceed.

First, [download Maven](#) and follow the [installation instructions](#). After that, type the following in a terminal or in a command prompt:

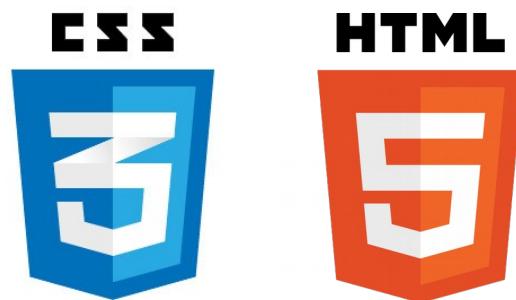
```
mvn --version
```

It should print out your installed version of Maven, for example:

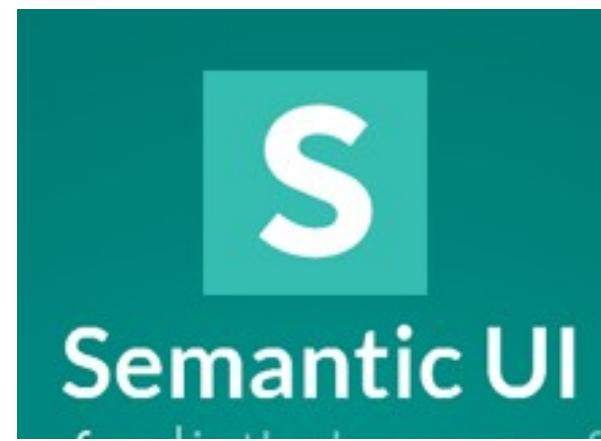
```
Apache Maven 3.0.5 (r01de14724cdef164cd33c7c8c2fe155faf9602da; 2013-02-19 14:51:28+0100)
Maven home: D:\apache-maven-3.0.5\bin\..
Java version: 1.6.0_25, vendor: Sun Microsystems Inc.
Java home: C:\Program Files\Java\jdk1.6.0_25\jre
Default locale: en_NL, platform encoding: Cp1252
```

Taxonomía de las webs

- Página web estática



- Se utilizan librerías de componentes CSS



Buttons and navigation

Use these vector based elements to mockup a Bootstrap website with form elements.

All elements are full customizable.

Buttons ▾ Navigation ▾ Labels Badges Typography Thumbnails Alerts Progress bars Miscellaneous

Default Primary Info Success Danger Warning Inverse 1
Default Primary Info Success Danger Warning Inverse 2
Default Primary Info Success Danger Warning Inverse 4
Large action Large action Default Success Warning Important Info Inverse 6
Large action Large action Default Success Warning Important Info Inverse 8
Large action Large action Default Success Warning Important Info Inverse 10

1 2 3 4 Left Middle Right Home Profile Messages Home Profile Messages

Home / Library / Data « 1 2 3 4 » ← Older Newer →

Seconday link
Something else here

Home
Profile
Messages

Home
Profile
Messages

Home
Library
Applications

Página web

Servidor estático

- Página web estática

- Página web interactiva

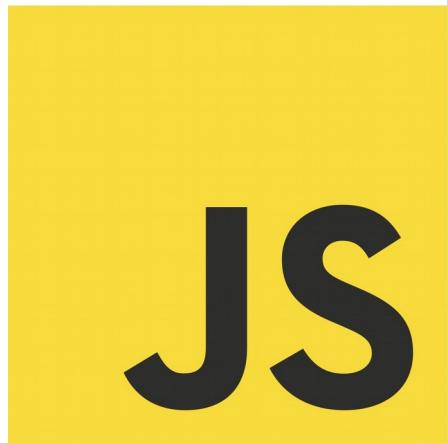
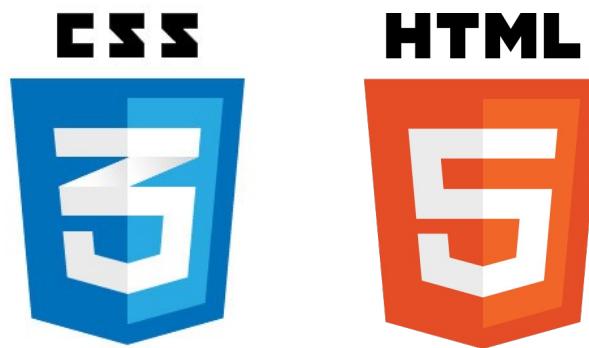
Aplicación web

Servidor dinámico

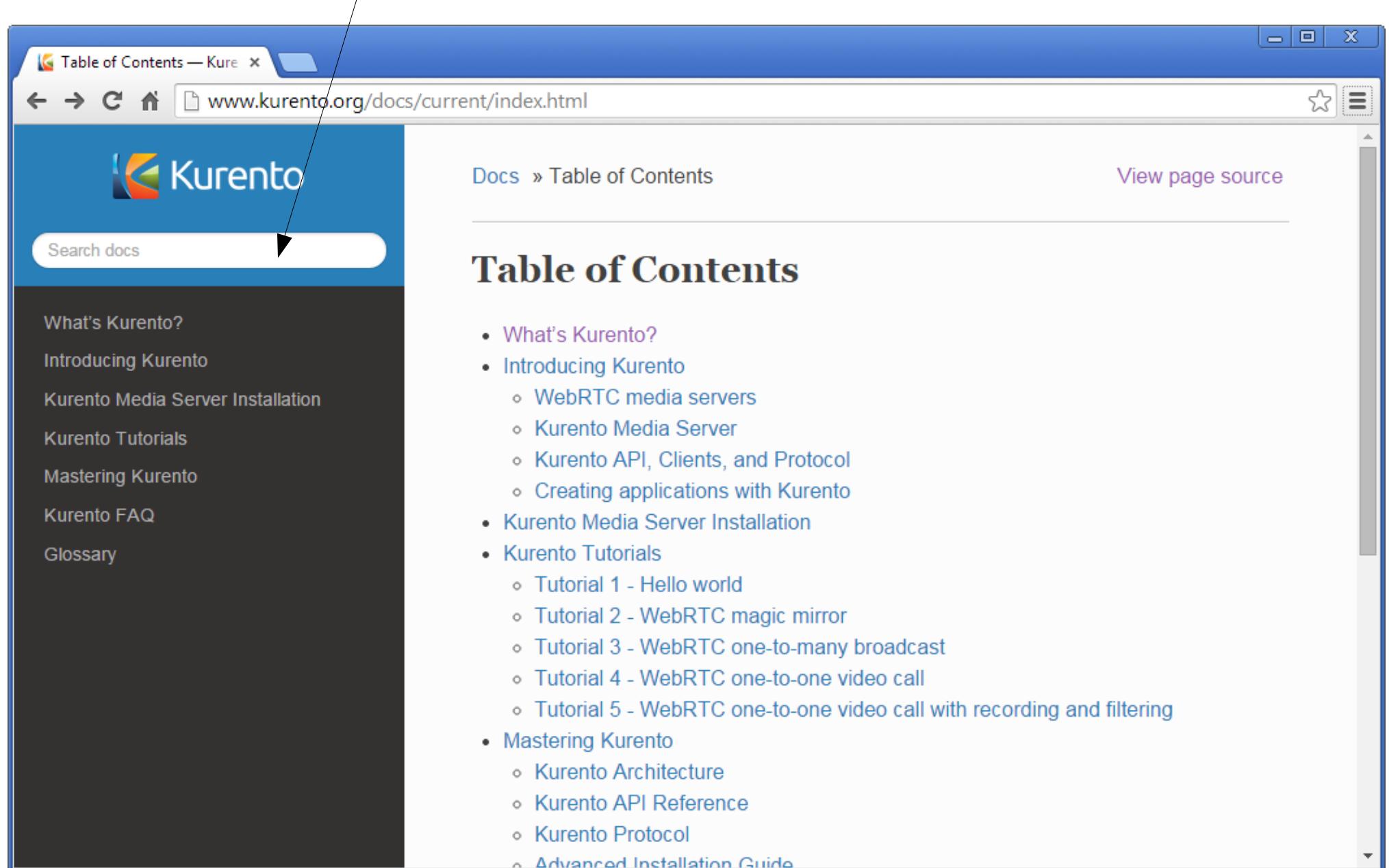
- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

Taxonomía de las webs

- Página web interactiva



Buscador implementado en JavaScript



The screenshot shows a web browser window with the title "Table of Contents — Kure". The address bar displays the URL www.kurento.org/docs/current/index.html. The page content is the "Table of Contents" for the Kurento documentation. On the left, there is a sidebar with links to various sections: "What's Kurento?", "Introducing Kurento", "Kurento Media Server Installation", "Kurento Tutorials", "Mastering Kurento", "Kurento FAQ", and "Glossary". A search bar labeled "Search docs" is also present in the sidebar. The main content area shows the "Table of Contents" heading and a list of topics:

Table of Contents

- [What's Kurento?](#)
- [Introducing Kurento](#)
 - [WebRTC media servers](#)
 - [Kurento Media Server](#)
 - [Kurento API, Clients, and Protocol](#)
 - [Creating applications with Kurento](#)
- [Kurento Media Server Installation](#)
- [Kurento Tutorials](#)
 - [Tutorial 1 - Hello world](#)
 - [Tutorial 2 - WebRTC magic mirror](#)
 - [Tutorial 3 - WebRTC one-to-many broadcast](#)
 - [Tutorial 4 - WebRTC one-to-one video call](#)
 - [Tutorial 5 - WebRTC one-to-one video call with recording and filtering](#)
- [Mastering Kurento](#)
 - [Kurento Architecture](#)
 - [Kurento API Reference](#)
 - [Kurento Protocol](#)
 - [Advanced Installation Guide](#)

Página web

Servidor estático

- Página web estática
- Página web interactiva

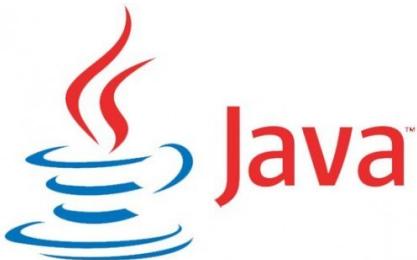
Aplicación web

Servidor dinámico

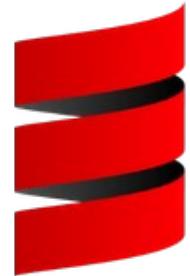
- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

Evolución de la web

Servidor dinámico



express



symfony



Portal de servicios Micael

miportal.urjc.es/portal/page/portal/portal_inicio/inicio

Inicio | Mis favoritos | Desconectar

 Universidad
Rey Juan Carlos

Portal de Servicios

Bienvenido/a: Micael Gallego Carrillo

Servicios Alumno

- [Mi Correo de Alumno - \(Ayuda\)](#)
- [Mis datos personales](#)
- [Mis notas](#)
- [Mis encuestas](#)
- [Mis traslados](#)
- [Aula Virtual](#)
- Expediente**
- [Mi expediente](#)
- [Mi nota media del expediente](#)
- [Mi progreso académico](#)

Alumnos

 **Centro de atención telefónica al alumno (C.A.T.A.)**

Fecha de Actualización: 06-FEB-2014

El C.A.T.A es un servicio de la Universidad Rey Juan Carlos cuyo objetivo es asistir a los alumnos y futuros alumnos en su búsqueda de información, al tiempo que asesora y orienta en las posibles cuestiones que puedan surgir en su relación con nuestra institución.

Horario del servicio:

Horario del servicio: de lunes a viernes de 9:00 a 20:00 horas

Teléfono: 91 488 93 93

Enlaces Generales

[Acceso DreamSpark Premium MSDN](#)

Enlaces Biblioteca

[Recordar pin](#)

[Acceso remoto a los recursos electrónicos de la biblioteca](#)

Enlaces Alumnos

[Centro de Atención Telefónica para el Alumno \(CATA\)](#)

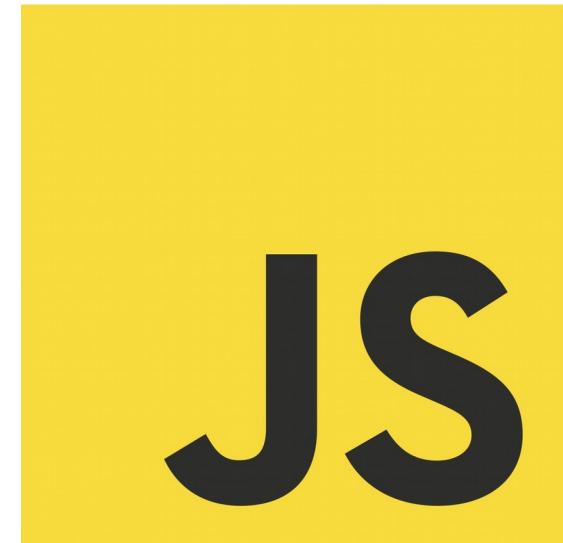
[Apple on Campus](#)

[CONVIVE - PROGRAMA](#)



- **Aplicaciones web con JavaScript**

- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA



Página web

Servidor estático

- Página web estática
- Página web interactiva

Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

[Sign Up](#)[Log In](#)

create a new account

Your Email

Please use a valid email address.

Confirm Email**Country****Zip Code**

Please provide a valid Zip Code

Password

Your password must be between 6-16 characters long.

Confirm Password

This does not match the password entered above.

Yes, I agree to the [Terms of Use](#)

[Sign Up](#)

Why you'll love it

- See all your accounts in one place
- Set a budget and pay down your debt
- Find the best ways to grow your money
- Stay safe and secure

Página web

Servidor estático

- Página web estática
- Página web interactiva

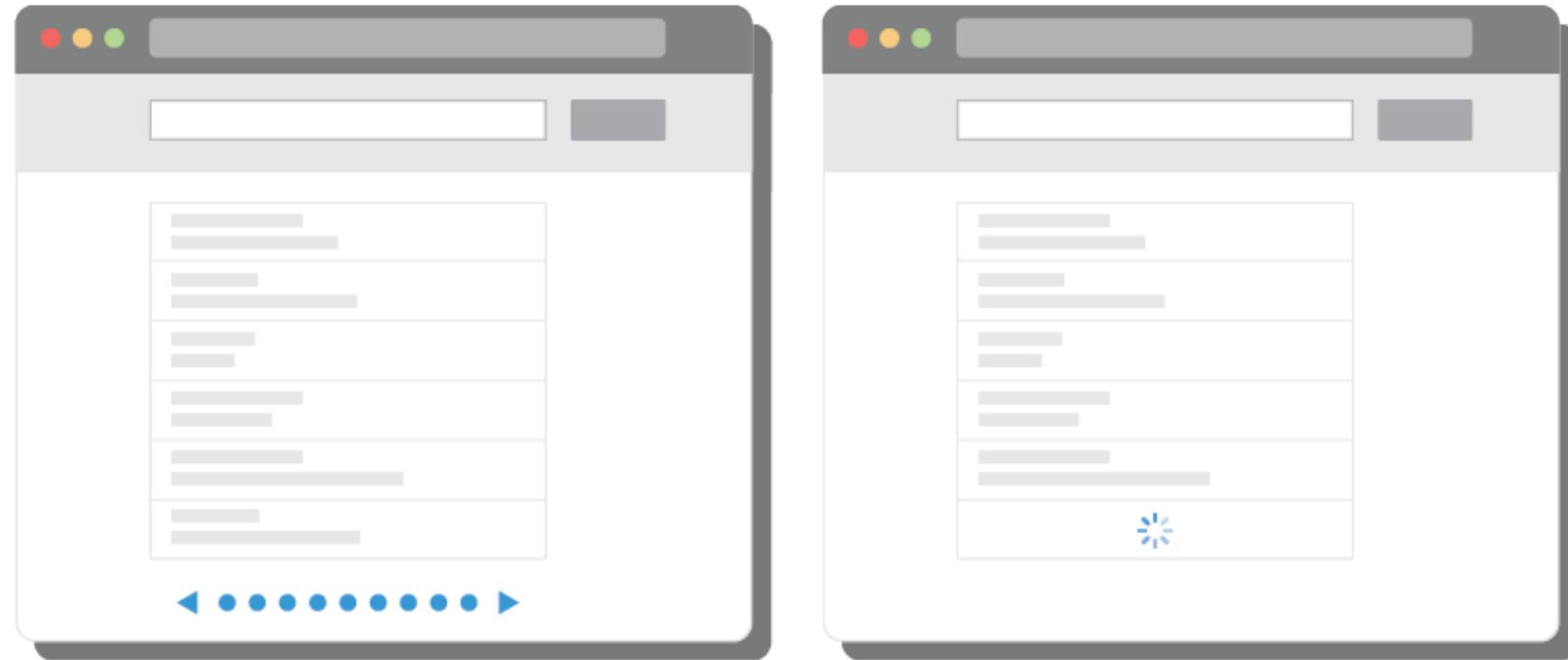
Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

Taxonomía de las webs

- Aplicación web con AJAX



- Aplicación web con AJAX



AJAX Username Verification

NOTE: Please type an username and continue filling the other fields. You'll see the result immediately.

Already existing members in this demo: **john, michael, terry, steve**

Username: ✓

Password:

Confirm Password:

Página web

Servidor estático

- Página web estática
- Página web interactiva

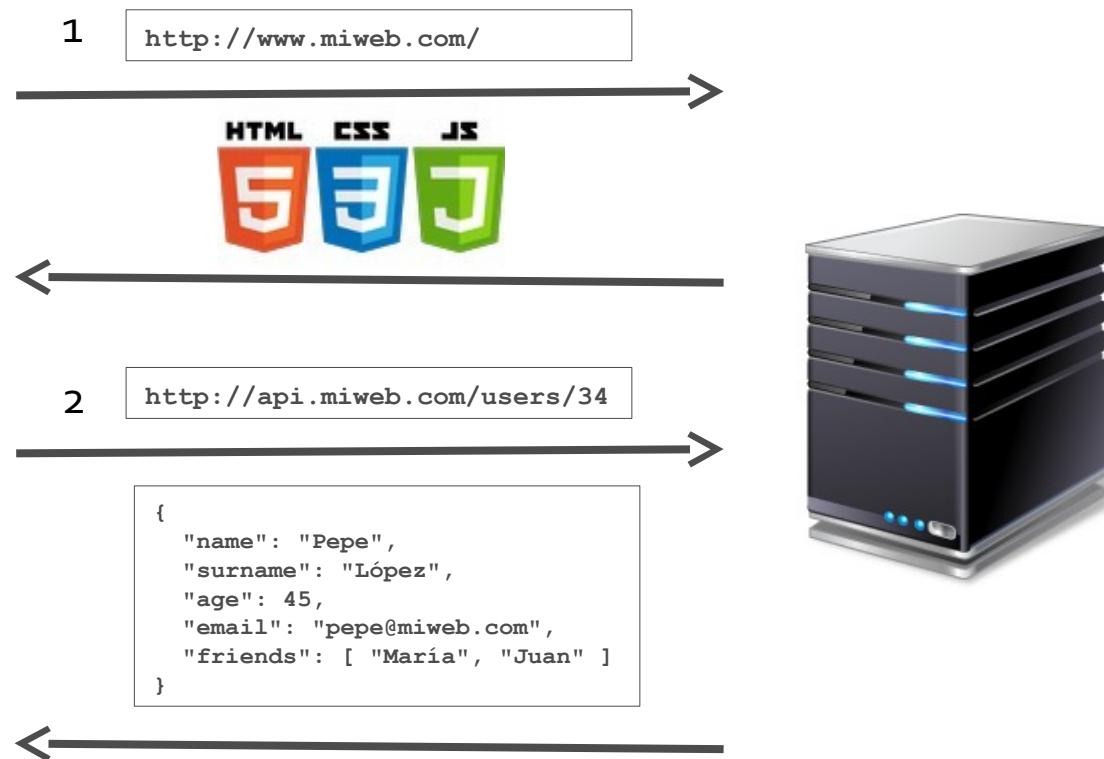
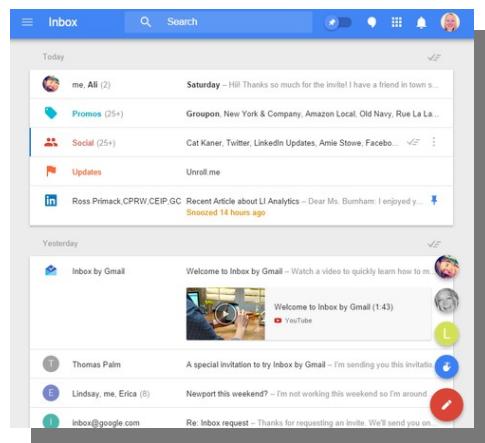
Aplicación web

Servidor dinámico

- Aplicación web con cliente estático
- Aplicación web interactiva
- Aplicación web con AJAX
- Aplicación web SPA

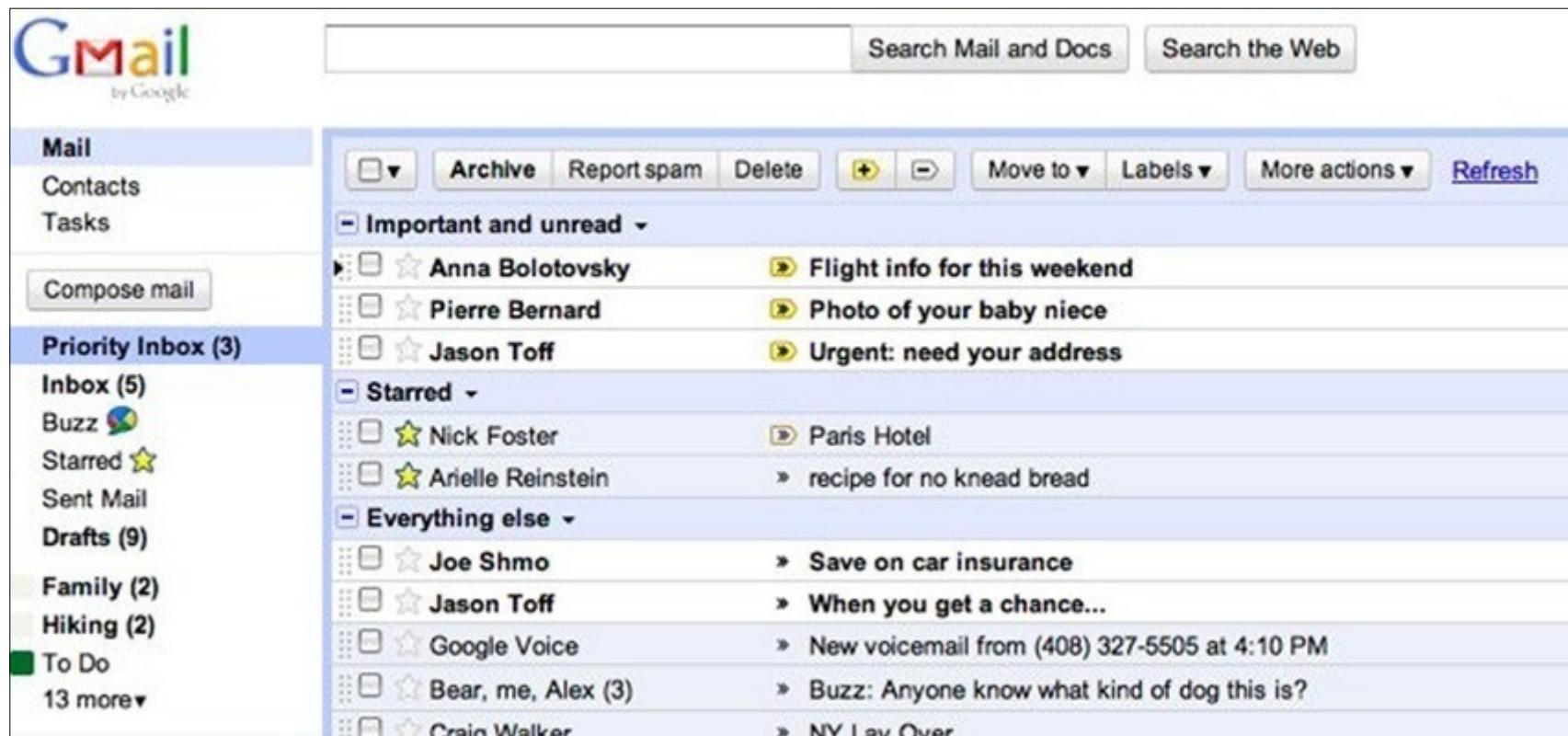
Taxonomía de las webs

- Aplicación web SPA



Taxonomía de las webs

- Aplicación web SPA



Google Maps

Universidad Rey Juan Carlos - Google Maps - Mozilla Firefox

Universidad Rey Juan Car... + New Tab

https://www.google.es/maps/place/Universidad+Rey+Juan+Carlos/@40.3347481,-3.8755547,15z | Buscar

Escuela Superior De Ciencias...

Micael

1

Universidad Rey Juan Carlos, Calle Tulipán, Móstoles

Universidad Rey Juan Carlos

3,8 ★★★★☆ 66 reseñas

Cómo llegar

Universidad

GUARDAR LUGARES CERCANOS ENVIAR A TU TELÉFONO COMPARTIR

Calle Tulipán, s/n, 28933 Móstoles, Madrid

urjc.es

916 65 50 60

Cerrado hoy

Escuela Oficial de Idiomas de Móstoles

Instituto de Educación Secundaria Benjamín Rúa

IES Los Rosales

Universidad Rey Juan Carlos

CAFETERIA HARVARD CAFÉ

Av. del Alcalde de Móstoles

Calle Alonso Cano

Av. de los Abogados de Atocha

Calle Gran Capitán

Ceil Alonso Cano

Tierra

Datos del mapa ©2016 Google, Inst. Geogr. Nacional, Fundación Espacio, Condiciones, Privacidad, Danos tu opinión, 100 m

Soundcloud

That's How I Feel by FIXED FETISH - Mozilla Firefox

Thats How I Feel by Fl... x +

https://soundcloud.com/groups/speedsound

SOUND CLOUD Charts Search for artists, bands, tracks, podcasts Sign in or Create account Upload ...

SPEEDSOUND

FIXED FETISH Thats How I Feel 2 hours #Electronic

0:07 3:39

Write a comment

Heart Share Embed Copy ▶ 7

ZEN PARTY TriWALKER Zen Party (LISTEN. 1 month #EDM

MFers) (FREE DL) 5:49

TZ CAT Buy ▶ 132 □ 2

Moderators View all

Speedsound REC. 28K 24 Follow

BLUE EYES KILL 19.9K Follow

164K members View all

Legal - Privacy - Cookies - Imprint - Popular searches

Language: English (US)

Playing from :: Electronic Music Lovers Network (...
That's How I Feel

Slides: Edit - Mozilla Firefox

Slides: Edit

https://slides.com/micaelgallego/deck-2/edit

Buscar

Shape

Line

Iframe

Table

Code

Math

Upgrade

+

Universidad
Rey Juan Carlos

grafo
RESEARCH

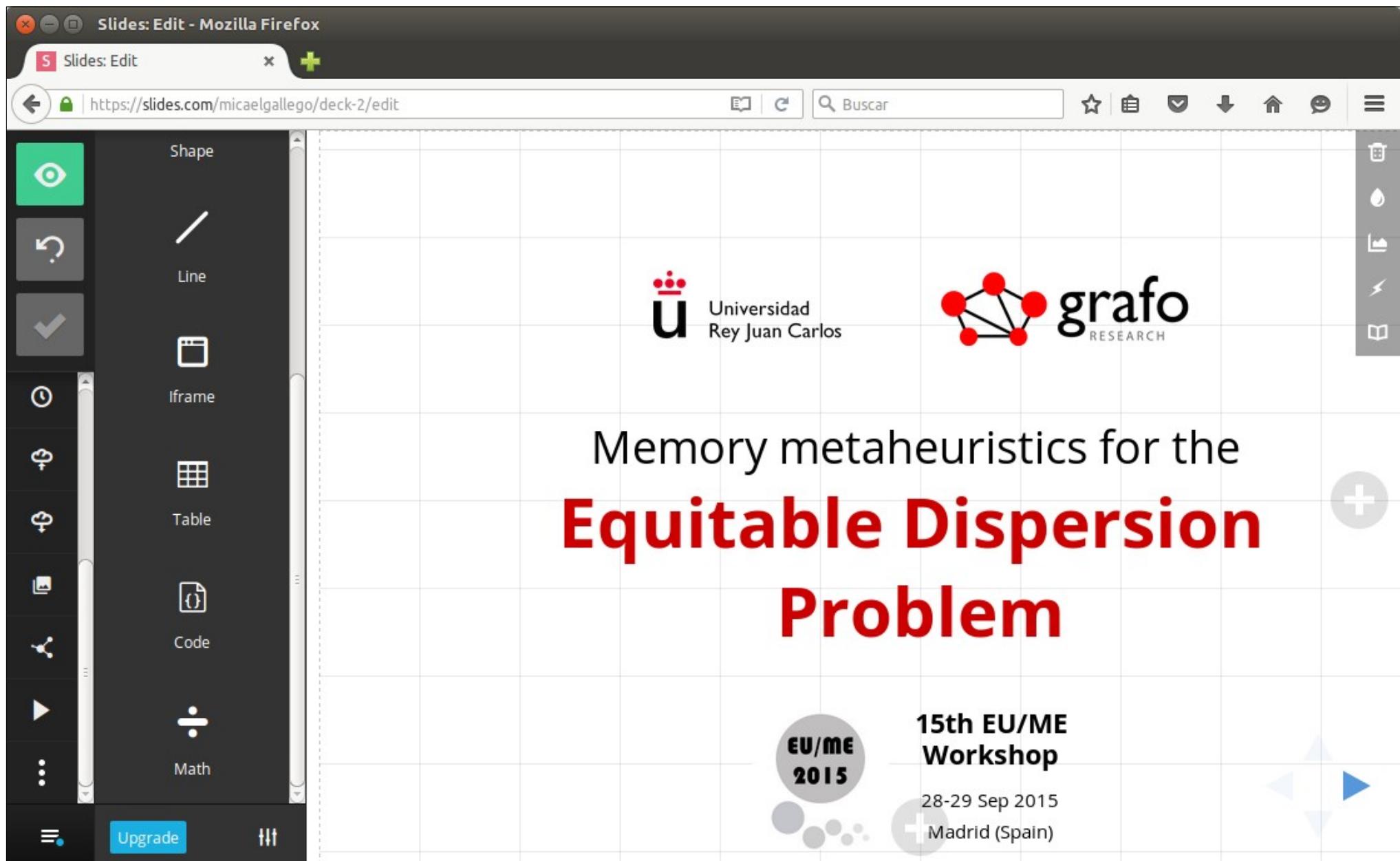
Memory metaheuristics for the
**Equitable Dispersion
Problem**

EU/ME
2015

15th EU/ME
Workshop

28-29 Sep 2015

Madrid (Spain)



GitHub

micaelgallego (Micael Gallego) - Mozilla Firefox

micaelgallego (Micael Gall...)

GitHub, Inc. (US) | https://github.com/micaelgallego

Buscar

Pull requests Issues Gist

Search GitHub

Contributions Repositories Public activity Edit profile


Micael Gallego
micaelgallego

Universidad Rey Juan Carlos
Madrid (Spain)
micael.gallego@gmail.com
<http://about.me/micael.gallego>
Joined on 23 Sep 2011

16 Followers 5 Starred 1 Following

Popular repositories

-  **simpleconcurrent** 1 ★
This repository contains a library for teaching ...
-  **spring-boot** 0 ★
Spring Boot
-  **kurento-client-core-js** 0 ★
Generated javascript api from kms-core project
-  **kurento-client-elements-js** 0 ★
Generated javascript api from kms-elements ...
-  **kurento-client-filters-js** 0 ★
Autogenerated code for js from kms-filters pro...

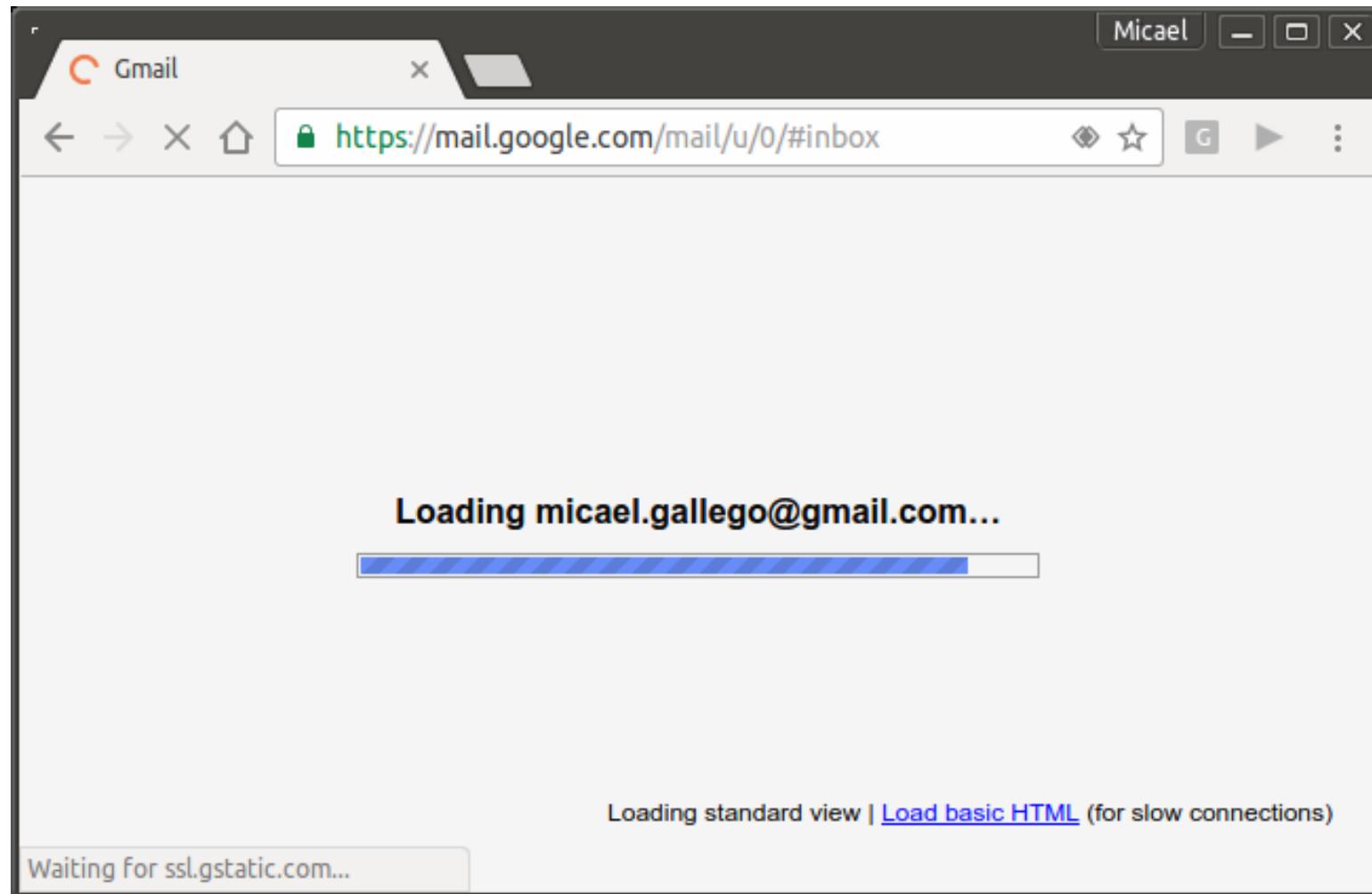
Repositories contributed to

-  **Kurento/kurento-java** 30 ★
Apis for using Kurento server from java
-  **Kurento/kurento-tree** 0 ★
-  **codeurjc/codeurjc.github.io** 0 ★
CodeUrjc web page
-  **sergiobanegas/perseus** 0 ★
-  **Kurento/kurento-room** 14 ★
Kurento room related projects

Contributions



Taxonomía de las webs



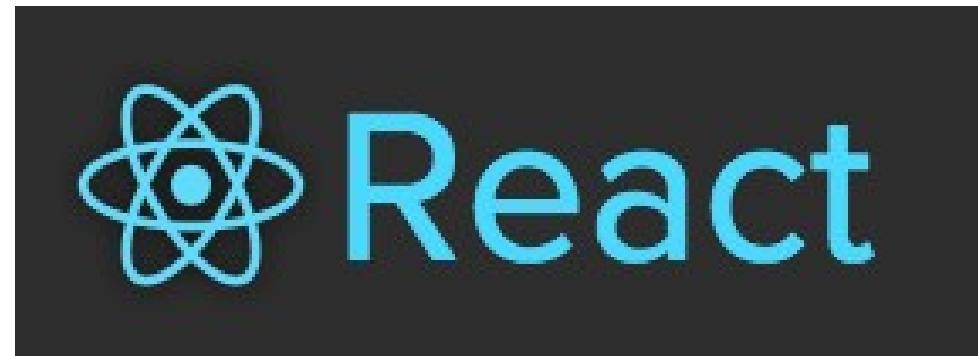
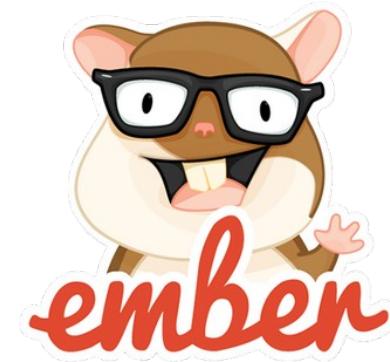


Angular 2

Frameworks / librerías SPA



BACKBONE.JS



Características



- Angular es un framework para desarrollo **SPA**
- Permite extender el **HTML con etiquetas propias**
 - Con contenido personalizado (HTML)
 - Con comportamiento personalizado (JavaScript)
- Interfaz basado en **componentes** (no en páginas)
- Se recomienda usar con **TypeScript** (aunque se puede con ES5 y ES6)

<https://angular.io/>



Angular 2

Funcionalidades

- Composición de componentes
- Inyección de dependencias
- Servicios (sin vista)
- Navegación por la app (Router)
- Cliente http (APIs REST)
- Animaciones
- Internacionalización
- Soporte para tests unitarios y e2e
- Librerías de componentes: Material Design
- Renderizado en el servidor (SEO friendly)
- ...

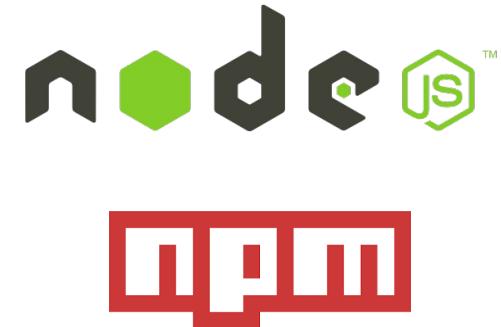
Angular 2 vs Angular 1

- Versión 2.2 (Noviembre 2016)
- Está implementado desde cero, no como una evolución de Angular 1
- Angular 2 **no es compatible con Angular 1**
- Cuidado, la **documentación de Angular 1 no sirve** para Angular 2

\$scope



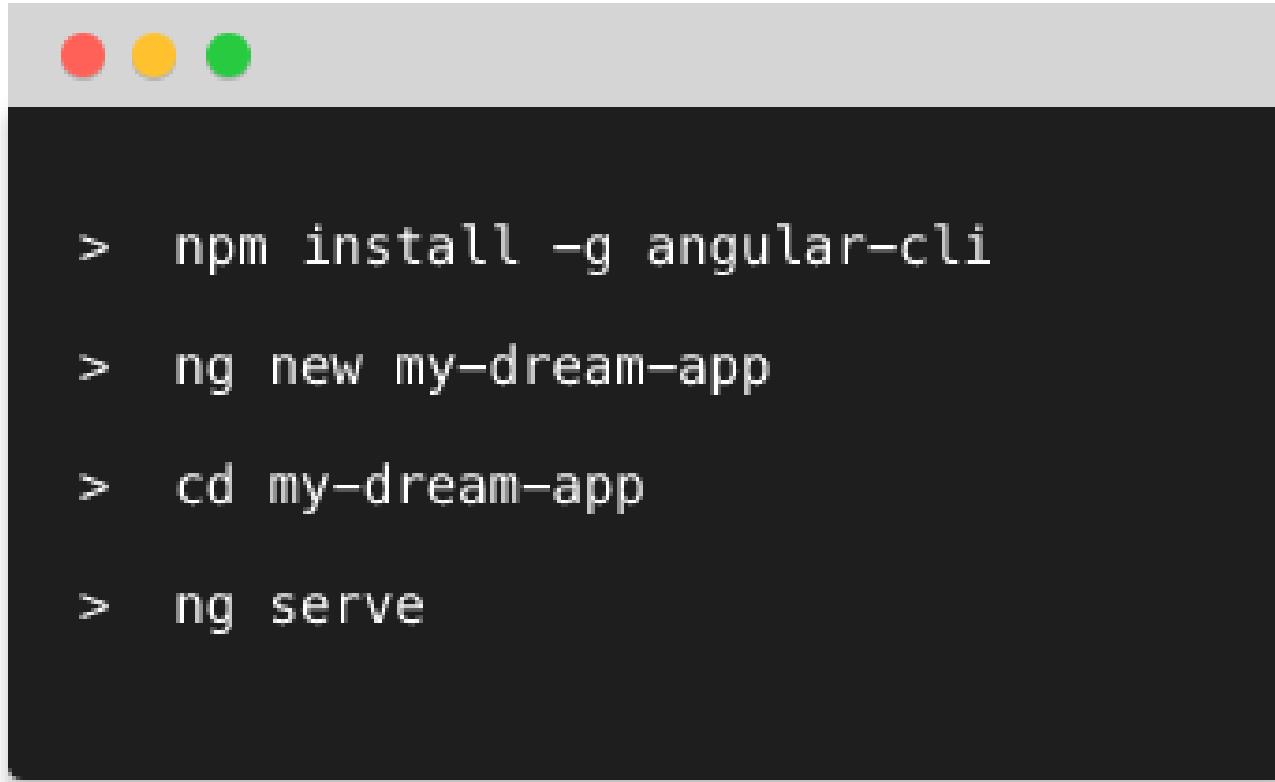
Angular 2



- Generación del proyecto **inicial**
- Generación de partes **posteriormente**
- Compilado automático de **TypeScript** y actualización del **navegador**
- Construcción del proyecto para distribución (*build*)

<https://github.com/angular/angular-cli>

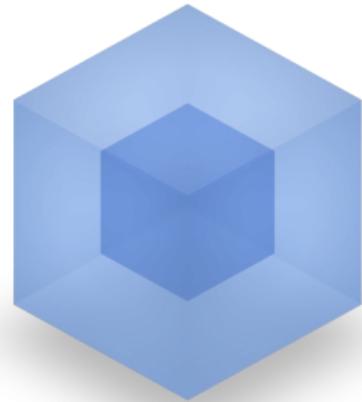
Angular 2



```
> npm install -g angular-cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Un proyecto ocupa unos **250Mb** en disco (se puede reutilizar las librerías entre varios proyectos)

Angular 2



webpack
MODULE BUNDLER

<https://webpack.github.io/>

**Construcción
del proyecto**

 ARMA
<https://karma-runner.github.io>

 Jasmine
<http://jasmine.github.io/>

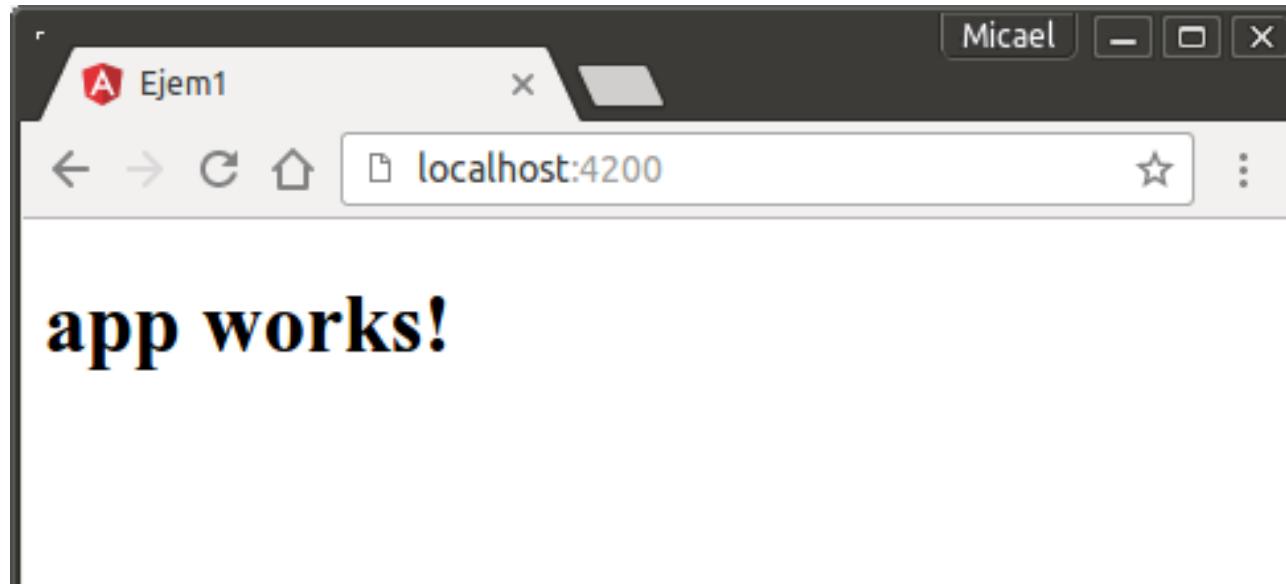
 Protractor
end to end testing for AngularJS

<http://www.protractortest.org/>

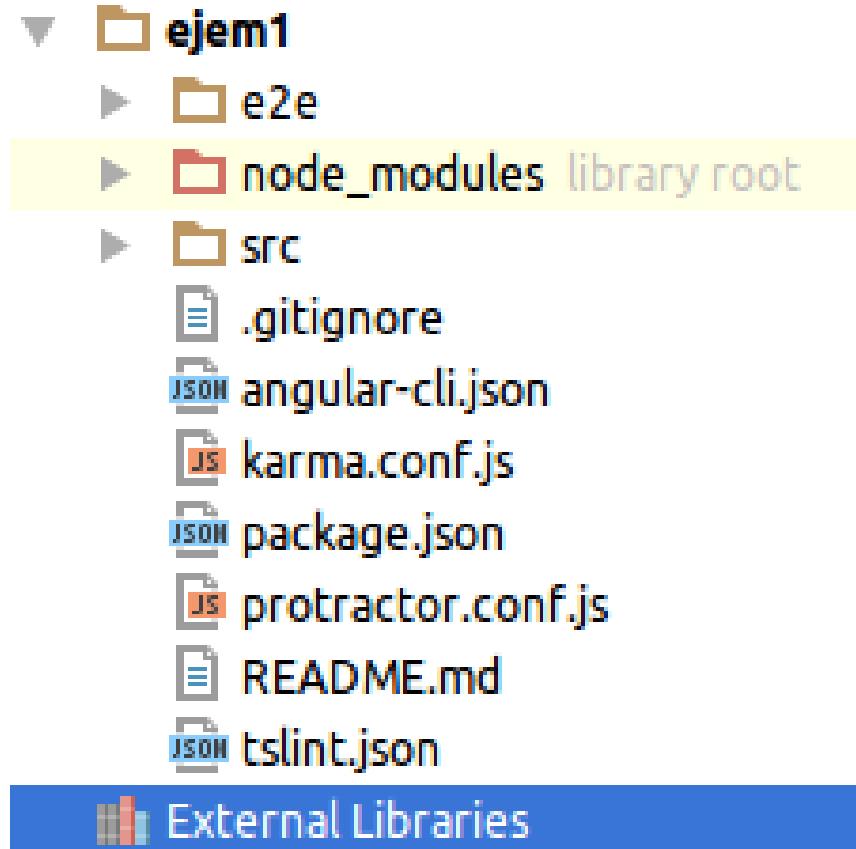
Herramientas de testing

Angular 2

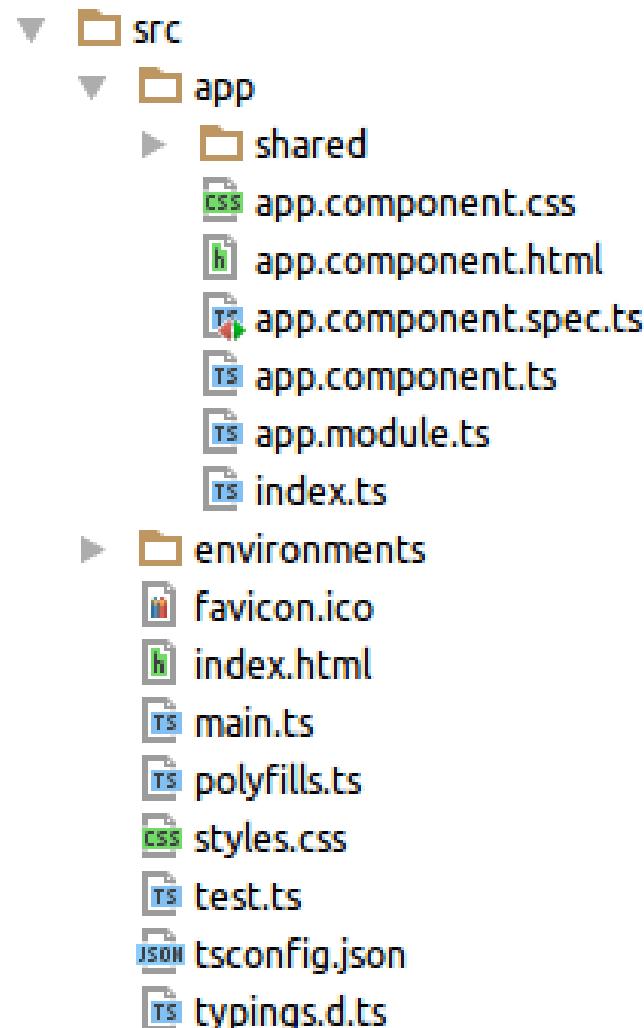
- **Ejecutar servidor web en modo desarrollo**
 - Se iniciará un servidor web en <http://localhost:4200>
 - Hay que abrir el navegador para ejecutar la app
 - Al guardar un fichero fuente, la aplicación se **recargará automáticamente**



Angular 2



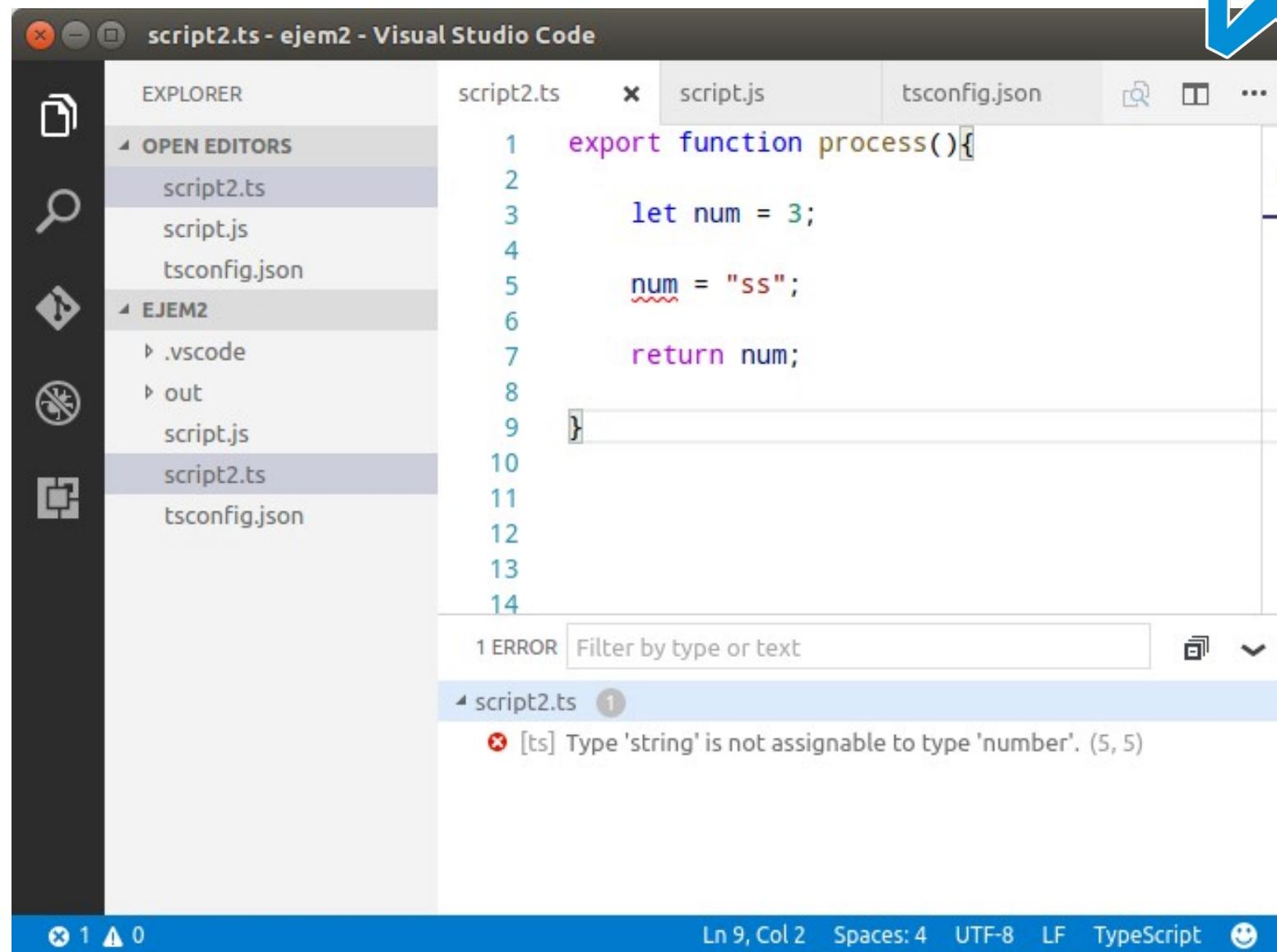
- **Ficheros/Carpetas generadas**
 - **src:** Fuentes de la aplicación
 - **package.json:** Configuración de librerías y herramientas
 - **dist:** Recursos que hay que publicar en el servidor web
 - **node_modules:** Librerías y herramientas descargadas
 - **e2e:** Testing end to end



- ## Ficheros/Carpetas generadas

 - **src/app:**
 - Carpeta que contiene los ficheros fuente principales de la aplicación.
 - Borraremos su contenido y le sustituiremos por los ejemplos
 - **main.ts:** Fichero principal de la aplicación. No es necesario modificarle
 - **favicon.ico:** Icono de la aplicación
 - **index.html:** Página principal. Se editará para incluir CSS globales en la web (bootstrap)
 - **tsconfig.json:** Configuración del compilador TS

Visual Studio Code



The screenshot shows the Visual Studio Code interface with the title bar "script2.ts - ejem2 - Visual Studio Code". The Explorer sidebar on the left lists files: "script2.ts", "script.js", and "tsconfig.json" under "OPEN EDITORS"; ".vscode", "out", "script.js", and "script2.ts" under "EJEM2". The main editor area contains the following TypeScript code:

```
1  export function process(){
2
3      let num = 3;
4
5      num = "ss";
6
7      return num;
8
9 }
10
11
12
13
14
```

A status bar at the bottom indicates "1 ERROR" and shows the file path "script2.ts" with a count of 1 error. The error message is: "[ts] Type 'string' is not assignable to type 'number'. (5, 5)".



<https://code.visualstudio.com>

Angular 2

- ## Descargar los ejemplos

- Descargar de github, descargar librerías y compartir librerías entre todos los proyectos

```
git clone https://github.com/codeurjc/ng2-panelsistemas  
cd ng2-panelsistemas  
cd ejem1  
npm install  
mv node_modules ../  
ln -s ../node_modules node_module
```

- Ejecutar un ejemplo

```
cd ejem1  
ng serve
```



TypeScript

Características

- JavaScript ES6 con tipos opcionales
- Las herramientas
 - Avisan de potenciales **errores**
 - Generan código JavaScript ES5 compatible con los browsers
- Integrado en todos los **editores / IDEs**

<http://www.typescriptlang.org/>

<https://www.gitbook.com/book/basarat/typescript/details>



Ventajas frente a JavaScript

- Con los **tipos**, las herramientas pueden avisar de potenciales problemas mucho mejor que un **linter**
- Los programas grandes son **menos propensos a errores**
- Los **IDEs** y **editores** pueden: Autocompletar, Refactorizar, Navegar a la definición
- Se puede programar muy parecido a **Java o C#** (pero no estás forzado a ello)

Facilidad de adopción para JavaScripters

- Existe mucha inferencia de tipos (evita tener que escribir los tipos constantemente)
- No tiene que cambiarse la forma de programar (no te obliga a programar con clases)
- Un mismo proyecto puede **combinar JS y TS**, lo que facilita migrar un proyecto existente

TypeScript

```
export class Empleado {          TypeScript

    private nombre:string;
    private salario:number;

    constructor(nombre:string,
                salario:number) {
        this.nombre = nombre;
        this.salario = salario;
    }

    getNombre() {
        return this.nombre;
    }

    toString() {
        return "Nombre:"+this.nombre+
               ", Salario:"+this.salario;
    }
}
```

TypeScript vs JavaScript ES5

Clase en TypeScript

```
export class Empleado {  
  
    private nombre:string;  
    private salario:number;  
  
    constructor(nombre:string,  
               salario:number) {  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    getNombre() {  
        return this.nombre;  
    }  
  
    toString() {  
        return "Nombre:"+this.nombre+  
               ", Salario:"+this.salario;  
    }  
}
```

Simulación de clase en JS ES5 con prototipos

```
function Empleado(nombre, salario) {  
  
    this.nombre = nombre;  
    this.salario = salario;  
}  
  
Empleado.prototype.getNombre = function() {  
    return nombre;  
};  
  
Empleado.prototype.toString = function() {  
    return "Nombre:"+this.nombre+,  
           Salario:"+this.salario;  
};
```

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {  
  
    private String nombre;  
    private double salario;  
  
    public Empleado(String nombre,  
                    double salario){  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    public String getNombre(){  
        return nombre;  
    }  
  
    public String toString(){  
        return "Nombre:"+nombre+  
               ", Salario:"+salario;  
    }  
}
```

Clase en TypeScript

```
export class Empleado {  
  
    private nombre:string;  
    private salario:number;  
  
    constructor(nombre:string,  
               salario:number){  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    getNombre(){  
        return this.nombre;  
    }  
  
    toString(){  
        return "Nombre:"+this.nombre+  
               ", Salario:"+this.salario;  
    }  
}
```

TypeScript

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {
    private String nombre;
    private double salario;

    public Empleado(String nombre,
                    double salario) {
        this.nombre = nombre;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

    public String toString() {
        return "Nombre:"+nombre+
               ", Salario:"+salario;
    }
}
```

Clase en TypeScript

```
export class Empleado {
    private nombre:string;
    private salario:number;

    constructor(nombre:string,
                salario:number) {
        this.nombre = nombre;
        this.salario = salario;
    }

    getNombre() {
        return this.nombre;
    }

    toString() {
        return "Nombre:"+this.nombre+
               ", Salario:"+this.salario;
    }
}
```

En Java las clases son públicas. En TypeScript las clases se exportan

TypeScript

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {
    private String nombre;
    private double salario;

    public Empleado(String nombre,
                    double salario) {
        this.nombre = nombre;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

    public String toString() {
        return "Nombre:"+nombre+
               ", Salario:"+salario;
    }
}
```

Clase en TypeScript

```
export class Empleado {
    private nombre:string;
    private salario:number;

    constructor(nombre:string,
                salario:number) {
        this.nombre = nombre;
        this.salario = salario;
    }

    getNombre() {
        return this.nombre;
    }

    toString() {
        return "Nombre:"+this.nombre+
               ", Salario:"+this.salario;
    }
}
```

En TS los tipos se ponen tras el nombre con : (dos puntos)

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {  
  
    private String nombre;  
    private double salario;  
  
    public Empleado(String nombre,  
                    double salario){  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    public String getNombre(){  
        return nombre;  
    }  
  
    public String toString(){  
        return "Nombre:"+nombre+  
               ", Salario:"+salario;  
    }  
}
```

Clase en TypeScript

```
export class Empleado {  
  
    private nombre:string;  
    private salario:number;  
  
    constructor(nombre:string,  
                salario:number){  
        this.nombre = nombre;  
        this.salario = salario;  
    }  
  
    getNombre(){  
        return this.nombre;  
    }  
  
    toString(){  
        return "Nombre:"+this.nombre+  
               ", Salario:"+this.salario;  
    }  
}
```

En TS el constructor es "constructor"

TypeScript

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {

    private String nombre;
    private double salario;

    public Empleado(String nombre,
                    double salario) {
        this.nombre = nombre;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

    public String toString() {
        return "Nombre:"+nombre+
               ", Salario:"+salario;
    }
}
```

Clase en TypeScript

```
export class Empleado {

    private nombre:string;
    private salario:number;

    constructor(nombre:string,
                salario:number) {
        this.nombre = nombre;
        this.salario = salario;
    }

    getNombre() {
        return this.nombre;
    }

    toString() {
        return "Nombre:"+this.nombre+
               ", Salario:"+this.salario;
    }
}
```

En TS los tipos se infieren (si quieras), incluso en el tipo que devuelven los métodos

TypeScript

Clases Java vs TypeScript

Clase en Java

```
public class Empleado {

    private String nombre;
    private double salario;

    public Empleado(String nombre,
                    double salario) {
        this.nombre = nombre;
        this.salario = salario;
    }

    public String getNombre() {
        return nombre;
    }

    public String toString() {
        return "Nombre:"+nombre+
               ", Salario:"+salario;
    }
}
```

Clase en TypeScript

```
export class Empleado {

    private nombre:string;
    private salario:number;

    constructor(nombre:string,
                salario:number) {
        this.nombre = nombre;
        this.salario = salario;
    }

    getNombre() {
        return this.nombre;
    }

    toString() {
        return "Nombre:"+this.nombre+
               ", Salario:"+this.salario;
    }
}
```

En TS siempre que quieras acceder a un elemento de la clase tendrás que usar this.

TypeScript

TypeScript

```
import { Empleado } from "./Empleado";

let emps = new Array<Empleado>();

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps) {
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
}) ;
```

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();  
  
emps.add(new Empleado('Pepe', 500));  
emps.add(new Empleado('Juan', 200));  
  
for(Empleado emp : emps){  
    System.out.println(emp.getNombre());  
}  
  
empleados.forEach(emp -> {  
    System.out.println(emp);  
});
```

TypeScript

```
import { Empleado } from './Empleado';  
  
let emps = new Array<Empleado>();  
  
emps.push(new Empleado('Pepe', 500));  
emps.push(new Empleado('Juan', 200));  
  
for(let emp of emps){  
    console.log(emp.getNombre());  
}  
  
empleados.forEach(emp => {  
    console.log(emp);  
});
```

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps) {
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps = new Array<Empleado>();

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps) {
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

En Java las clases de la misma carpeta son del mismo paquete.

En TS cada fichero es un módulo, por eso hay que importar desde otros ficheros

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps){
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps: Array<Empleado> = [];

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps) {
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

En TS las variables se declaran con let y tienen ámbito de bloque y no se pueden declarar dos veces. Se podría usar var (como JS), pero el ámbito sería la función y se podrían redeclarar

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps){
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps: Array<Empleado> = [];

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps){
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

En TS las variables se pueden declarar con tipo (después de :), pero es opcional porque el tipo se infiere de la inicialización. En Java no lo veremos hasta Java 9 o Java 10

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps){
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps = new Array<Empleado>();

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps){
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

En Java usamos List y ArrayList generificados del API.
 En TS usamos el Array nativo de JS generificado por TS.

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();
emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps) {
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps = new Array<Empleado>();
emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps) {
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

En Java List el método es “add”
 En el array de JS el método es “push”

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps) {
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps = new Array<Empleado>();

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps) {
    console.log(emp.getNombre());
}

empleados.forEach(emp => {
    console.log(emp);
});
```

La sintaxis del foreach es muy parecida en Java y TS.
En TS está basado en iteradores (como en Java)

TypeScript

Imports / Listas / foreach / lambdas

Java

```
List<Empleado> emps = new ArrayList<>();

emps.add(new Empleado('Pepe', 500));
emps.add(new Empleado('Juan', 200));

for(Empleado emp : emps){
    System.out.println(emp.getNombre());
}

empleados.forEach(emp -> {
    System.out.println(emp);
});
```

TypeScript

```
import { Empleado } from './Empleado';

let emps = new Array<Empleado>();

emps.push(new Empleado('Pepe', 500));
emps.push(new Empleado('Juan', 200));

for(let emp of emps){
    console.log(emp.Nombre());
}

empleados.forEach(emp => {
    console.log(emp),
});
```

Las expresiones lambda de Java se llaman arrow function en TS.
 Se diferencian en la “flecha” con – o con =

TypeScript

Definición de atributos en el constructor

```
class Animal {  
    private name:string;  
    constructor(name: string) {  
        this.name = name;  
    }  
}
```

TypeScript

Definición de atributos en el constructor

```
class Animal {  
    private name:string;  
    constructor(name: string) {  
        this.name = name;  
    }  
}
```



```
class Animal {  
    constructor(private name: string) {  
    }  
}
```

TypeScript

Definición de atributos en el constructor

```
class Animal {  
    private name:string;  
    constructor(name: string) {  
        this.name = name;  
    }  
}
```

```
class Animal {  
    constructor(private name: string) {  
    }  
}
```

En TS se puede declarar un atributo e inicializar su valor desde el constructor declarando ese atributo como parámetro del constructor y usando el modificador de visibilidad

TypeScript

Objetos literales “tipados”

```
interface SquareConfig {  
    color: string;  
    width?: number;  
}
```

```
let config: SquareConfig;  
  
config = {color: "black"};  
config = {color: "black", width: 20};
```

Objetos literales “tipados”

```
interface SquareConfig {  
    color: string;  
    width?: number;  
}
```

```
let config: SquareConfig;  
  
config = {color: "black"};  
config = {color: "black", width: 20};
```

TypeScript

Tipos unión

```
let id: string | number;  
  
id = 3;  
...  
Id = "3";  
  
if (typeof id === "string") {  
    ...  
}
```

En JS es habitual que ciertas variables puedan tener la misma información aunque se represente con varios “tipos”. TS permite definir variables con varios tipos

{codemotion}



(codemotion)
MADRID · NOV 18-19 - 2016



TypeScript: Un
lenguaje aburrido
para programadores
torpes y tristes
by Micael Gallego

<https://www.youtube.com/watch?v=32cPEcX3Qao>

{ the *Code* }

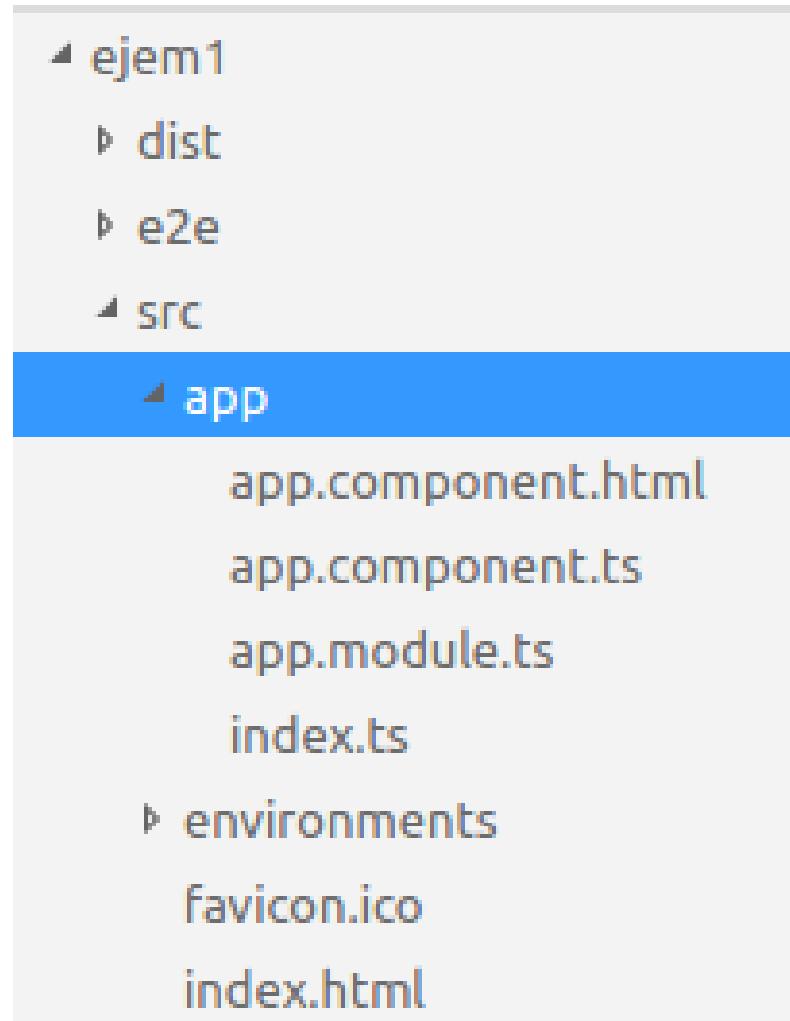
SHOW ME

Hola mundo

Componentes

- Las aplicaciones están basadas en **componentes**
- Un componente es una **nueva etiqueta HTML** con una **vista** y una **lógica** definidas por el desarrollador
 - La **vista** es una plantilla (*template*) en HTML con elementos especiales
 - La **lógica** es una clase TypeScript vinculada a la vista

Hola Mundo



Hola Mundo

ejem1

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
```

app.component.html

```
<h1>My First Angular 2 App</h1>
```

Lógica

Vista

Hola Mundo

ejem1

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
```

app.component.html

```
<h1>My First Angular 2 App</h1>
```

Lógica

Vista

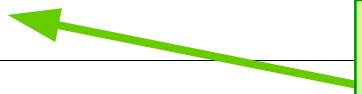
Hola Mundo

ejem1

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
```



Este componente no tiene ninguna lógica

app.component.html

```
<h1>My First Angular 2 App</h1>
```

Lógica

Vista

Hola Mundo

ejem1



Hola Mundo

ejem1

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: `
    <h1>
      My First Angular 2 App
    </h1>
  `
})
export class AppComponent {
```

Se puede incluir la **vista** (HTML del template) directamente en la **clase**. Si se usa la tildes invertidas (`) (grave accent), se puede escribir HTML multilínea

ejem2

Visualización de una variable

La vista del componente (HTML) se genera en función de su estado (**atributos de la clase**)

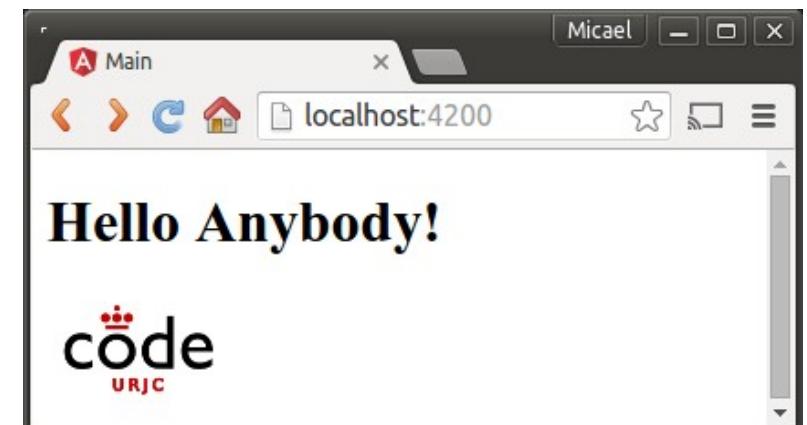
`app.component.ts`

```
import { Component } from
'@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  name = 'Anybody';
  imgUrl = "assets/img.png";
}
```

`app.component.html`

```
<h1>Hello {{name}}!</h1>
<img [src]="imgUrl"/>
```



ejem2

Visualización de una variable

La vista del componente (HTML) se genera en función de su estado (**atributos de la clase**)

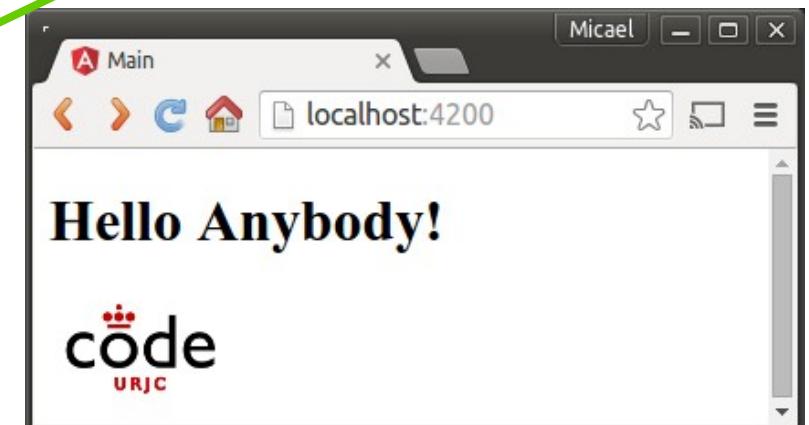
`app.component.ts`

```
import { Component } from
'@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  name = 'Anybody';
  imgUrl = "assets/img.png";
}
```

`app.component.html`

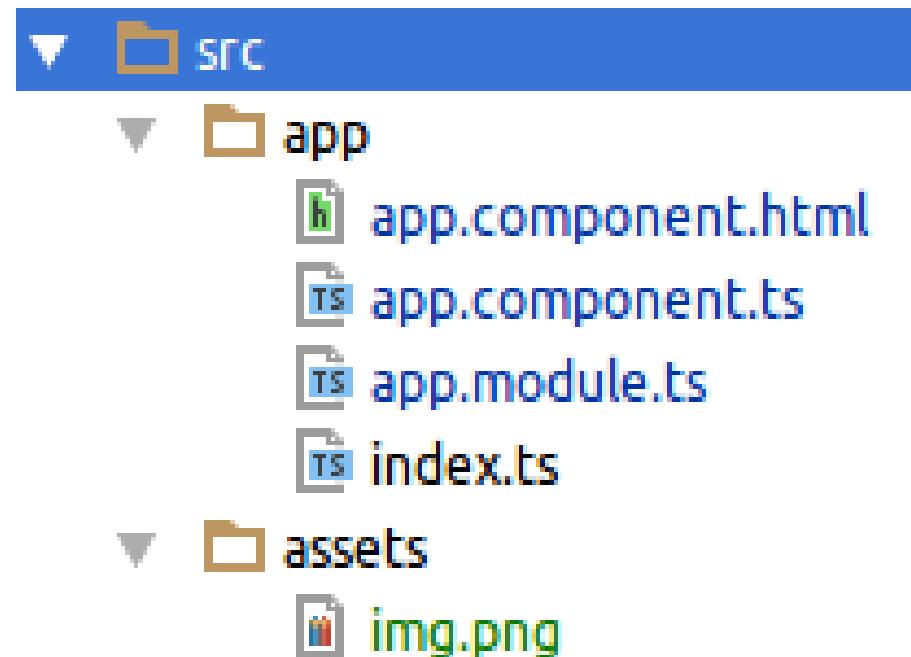
```
<h1>Hello {{name}}!</h1>
<img [src]="imgUrl"/>
```



ejem2

Recursos de la app

Los recursos (imágenes, fonts..) deben colocarse en una carpeta **src/assets** para que se copien en el **build**



Interactividad

Interactividad

ejem3

Ejecución de lógica

Se puede ejecutar un método ante un evento producido en la vista del componente

`app.component.ts`

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {
  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

`app.component.html`

```
<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
  Hello John
</button>
```

Interactividad

ejem3

Ejecución de lógica

Se puede ejecutar un método ante un evento producido en la vista del componente

`app.component.ts`

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {

  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

`app.component.html`

```
<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
  Hello John
</button>
```

Interactividad

ejem3

Ejecución de lógica

Se puede ejecutar un método ante un evento producido en la vista del componente

`app.component.ts`

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {

  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

Se puede definir cualquier **evento** disponible en el **DOM** para ese elemento

`app.component.html`

```
<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
  Hello John
</button>
```



Interactividad

ejem3

Ejecución de lógica

Se puede ejecutar un método ante un evento producido en la vista del componente



Interactividad

ejem4

Datos enlazados (*data binding*)

Un campo de texto se puede “enlazar” a un atributo
Atributo y componente están sincronizados

`app.component.ts`

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {
  name = 'Anybody';
  setName(name:string){
    this.name = name;
  }
}
```

`app.component.html`

```
<input type="text" [(ngModel)]="name">

<h1>Hello {{name}}!</h1>

<button (click)="setName('John')">
  Hello John
</button>
```

Interactividad

ejem4

Datos enlazados (*data binding*)

Un campo de texto se puede “enlazar” a un atributo
Atributo y componente están sincronizados

`app.component.ts`

```
import {Component} from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html'
})
export class AppComponent {
  name = 'Anybody';

  setName(name:string){
    this.name = name;
  }
}
```

`app.component.html`

```
<input type="text" [(ngModel)]="name">

<h1>Hello {{name}}!</h1>

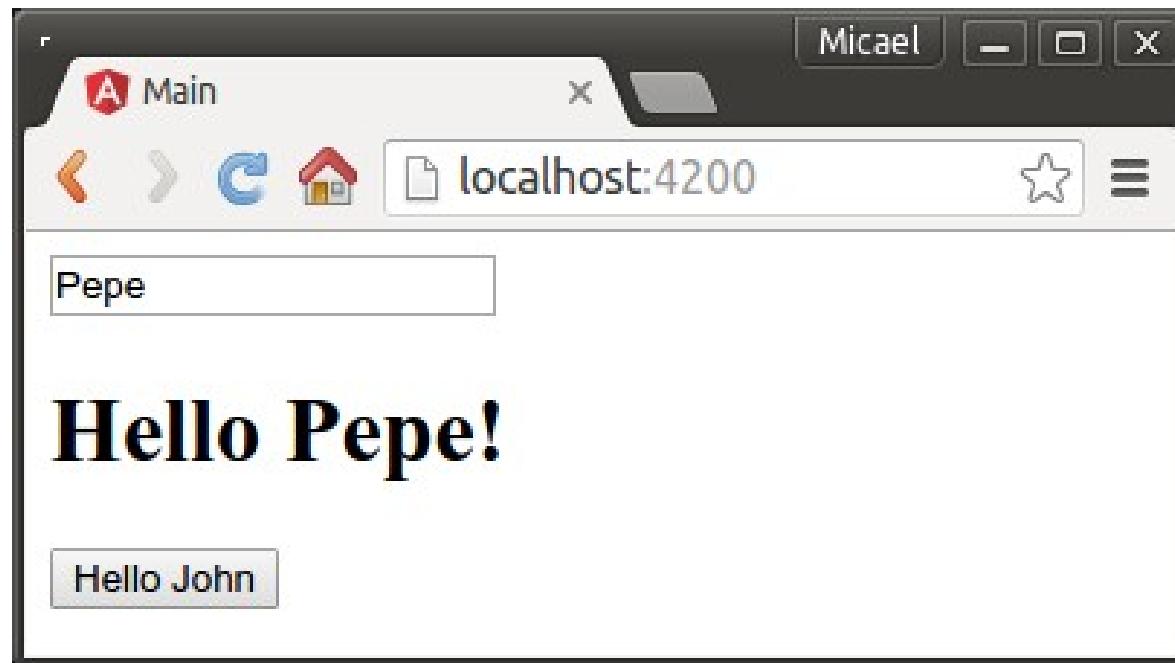
<button (click)="setName('John')">
  Hello John
</button>
```

Interactividad

ejem4

Datos enlazados (*data binding*)

Un campo de texto se puede “enlazar” a un atributo
Atributo y componente están sincronizados



Templates

Templates

- Los **templates** permiten definir la vista en función de la información del componente
 - Visualización condicional
 - Repetición de elementos
 - Estilos
 - Formularios

<https://angular.io/docs/ts/latest/guide/template-syntax.html>

Templates

• Visualización condicional

- Por ejemplo dependiendo del **valor del atributo booleano visible**

ejem5

```
<p *ngIf="visible">Text</p>
```

- También se puede usar una **expresión**

```
<p *ngIf="num == 3">Num 3</p>
```

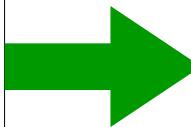
Templates

- Repetición de elementos

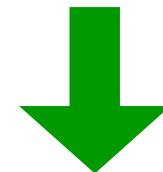
ejem5

```
<div *ngFor="let elem of elems">{{elem.desc}} </div>
```

```
items = [  
  { desc: 'Elem1', check: true },  
  { desc: 'Elem2', check: true },  
  { desc: 'Elem3', check: false }  
]
```



```
<div>Elem1</div>  
<div>Elem2</div>  
<div>Elem3</div>
```



Templates

ejem5

- **Estilos**

- Existen varias formas de definir un CSS
 - Globalmente asociado al index.html
 - Local al componente:
 - En la propiedad styles o styleUrls de @Component
 - En el template

Templates

ejem5

- **Estilos**
 - Globalmente asociado al index.html
 - Podemos modificar manualmente el fichero **index.html** y asociar un CSS (como haríamos con cualquier HTML)
 - Con angular-cli, si creamos un fichero **src/styles.css** se incluirá de forma automática en el **index.html**

<https://github.com/angular/angular-cli#global-styles>

Templates

ejem5

- ## Estilos

- En la propiedad styles o styleUrls de @Component

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styles: [`  
    .red { color: red; }  
    .blue { color: blue; }  
  `]  
})  
export class AppComponent {  
  ...  
}
```

Se suelen usar los strings multilínea con tildes invertidas

Templates

ejem5

- ## Estilos

- En la propiedad styles o styleUrls de @Component

```
@Component({  
  selector: 'app',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})  
export class AppComponent {  
  ...  
}
```



Con angular-cli no se puede usar styles y styleUrls a la misma vez en un component

Templates

ejem5

- Estilos

- En el template

```
<style>
  .orange {
    color: orange;
  }
</style>

<h1 [class]="className">Hello {{name}}!</h1>

<button (click)="setClass('blue')">Blue</button>
...
```

Templates

ejem5

- ## Estilos

- Hay muchas formas de controlar los estilos de los elementos
 - Asociar la **clase** de un elemento a un atributo de tipo string
 - Activar una **clase concreta** con un atributo boolean
 - Asociar la **clase** de un elemento a un atributo de tipo mapa de string a boolean
 - Asociar un **estilo concreto** de un elemento a un atributo

Estilos

ejem5

- Asociar la clase de un elemento a un atributo string
 - Cambiando el valor del atributo se cambia la clase del elemento
 - Por ejemplo, la clase del elemento h1 se cambia modificando el atributo string **className**

```
<h1 [class]="className">Title!</h1>
```

Estilos

- Activar una clase concreta con un atributo boolean

- Activa o desactiva una clase red con el valor de redActive

```
<h1 [class.red]="redActive">Title!</h1>
```

- Se puede usar para varias clases

```
<h1 [class.red]="redActive"
      [class.yellow]="yellowActive">
    Title!
</h1>
```

Estilos

• Asociar la clase de un elemento a un mapa ejem5

- Para gestionar varias clases es mejor usar un mapa de string (nombre de la clase) a boolean (activa o no)



```
<p [ngClass]="pClasses">Text</p>
```

```
pClasses = {  
  "red": false,  
  "bold": true  
}
```

```
changeParagraph () {  
  this.pClasses.bold = true;  
}
```

Estilos

ejem5

- Asociar un estilo concreto a un atributo
 - En algunos casos es mejor cambiar el estilo directamente en el elemento

```
<p [style.backgroundColor]="pColor">Text</p>
```

- Con unidades

```
<p [style.fontSize.em]="pSizeEm">Text</p>
```

```
<p [style.fontSize.%]="pSizePerc">Text</p>
```

Estilos

ejem5

- Asociar un estilo concreto a un atributo
 - Usando mapas de propiedad a valor

```
<p [ngStyle]="getStyles () ">Text</p>
```

```
getStyles () {  
  return {  
    'font-style':this.canSave? 'italic':'normal',  
    'font-weight':!this.isUnchanged? 'bold':'normal',  
    'font-size':this.isSpecial? '24px':'8px',  
  }  
}
```

Formularios

- Existen diversas formas de controlar formularios en Angular 2
 - Vincular un control del formulario a un atributo del componente (*data binding*)
 - Acceso a los controles desde el **código** para leer y modificar su estado
 - Mecanismos **avanzados** con validación con **ngControl** (no los veremos en el curso)

<https://angular.io/docs/ts/latest/guide/forms.html>

Formularios

ejem6

• Campo de texto

- Se vincula el control a un atributo del componente con [(ngModel)]
- Cualquier cambio en el control se refleja en el valor del atributo (y viceversa)

```
<input type="text" [(ngModel)]="name">  
<p>{{ name }}</p>
```

name:string

Formularios

ejem6

- **Acceso a los controles desde el código**

- Un elemento del template puede asociarse a una variable
- Podemos usar esa variable en el código del template para manejar ese elemento

```
<input #cityInput type="text">
```

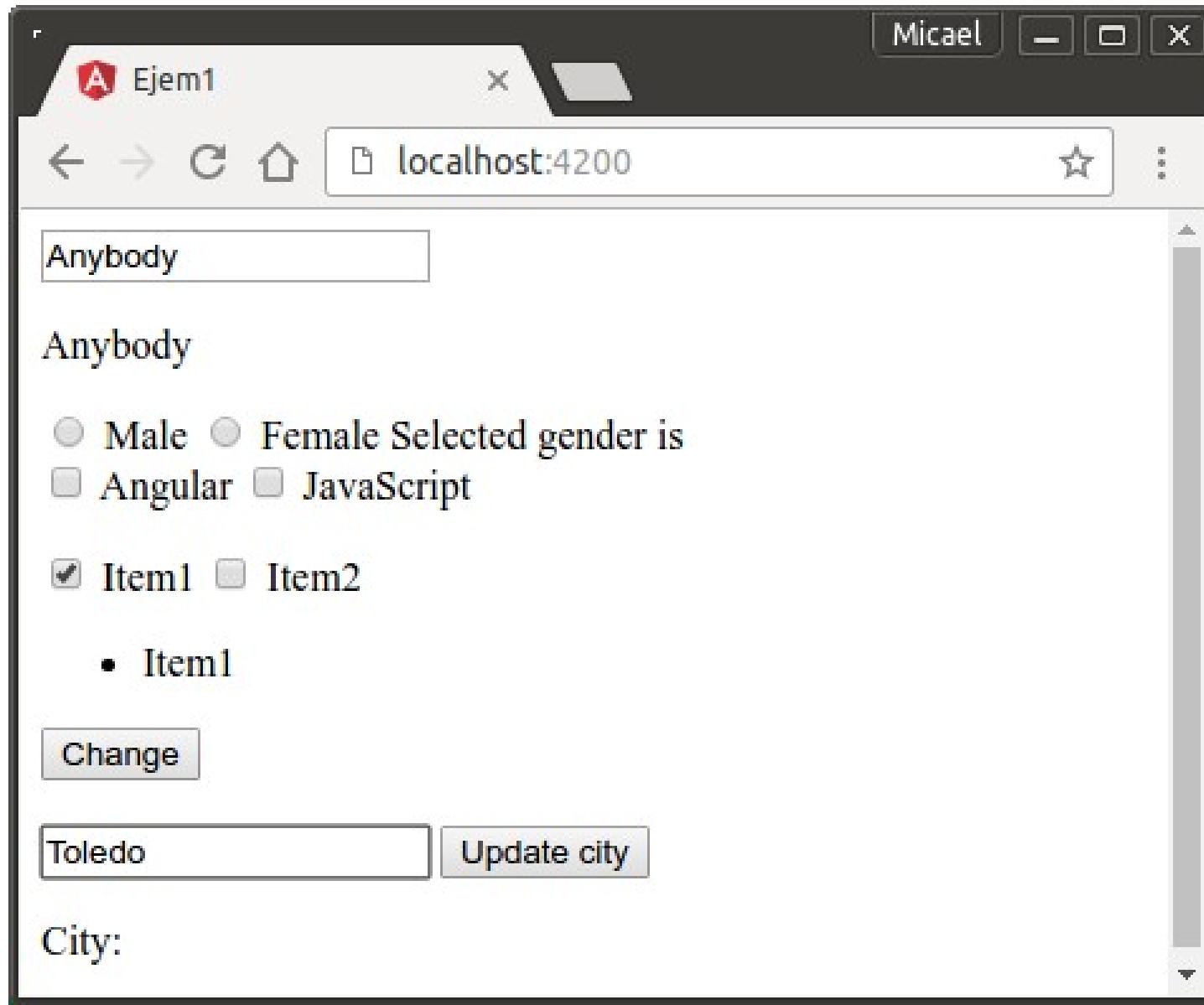
Este control input puede referenciarse con la variable cityInput

```
<button (click)="update(cityInput.value); cityInput.value=''">  
    Update city  
</button>
```

Podemos usar las propiedades y métodos del elemento definidos en su API DOM

Formularios

ejem6

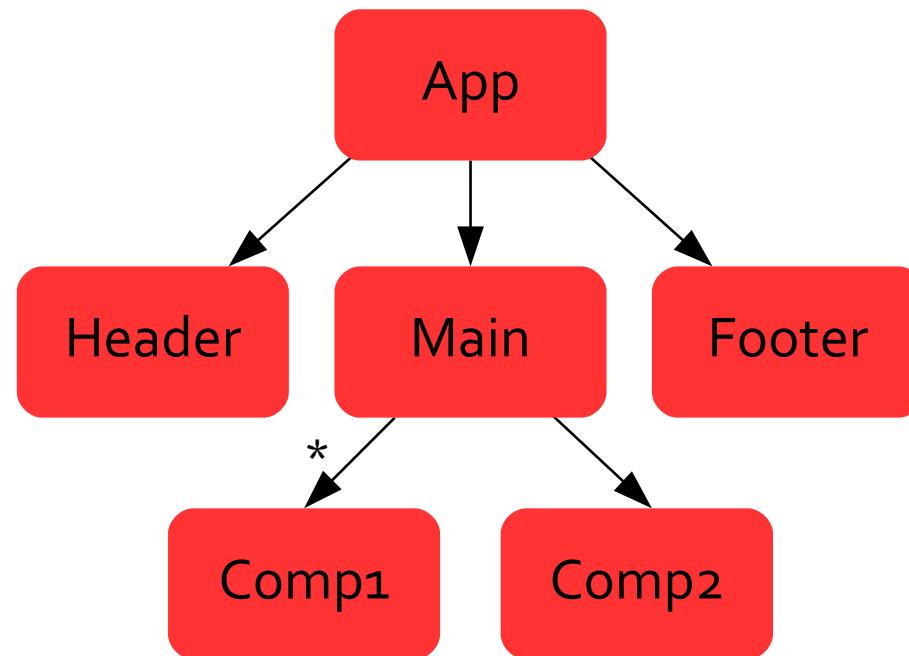


Composición de componentes

Composición de componentes

Árboles de componentes

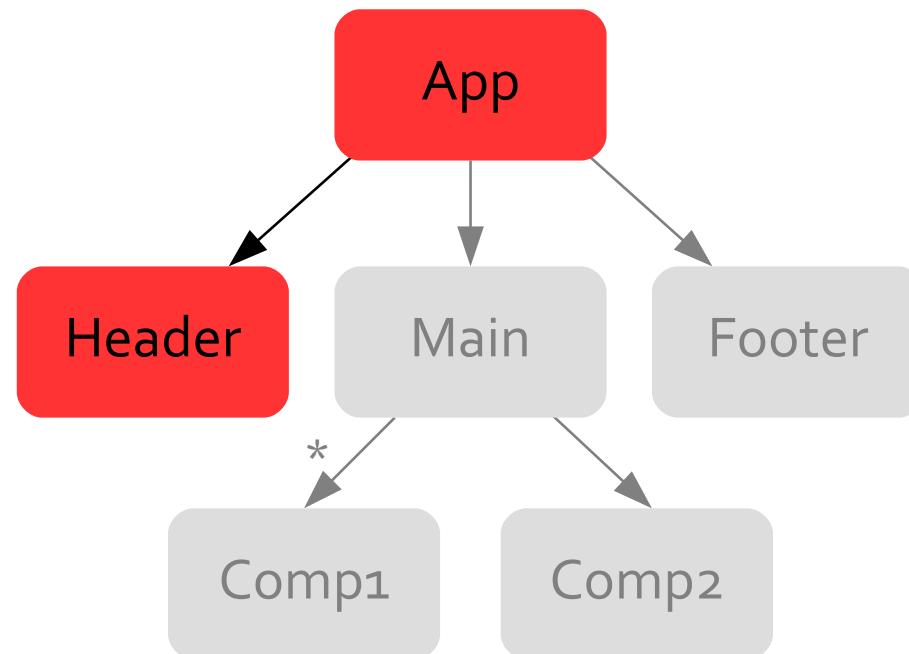
En Angular 2 un componente puede estar formado por más componentes formando un árbol



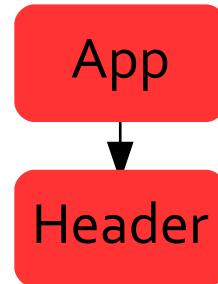
Composición de componentes

Árboles de componentes

En Angular 2 un componente puede estar formado por más componentes formando un árbol



Composición de componentes



Árboles de componentes

```
<h1>Title</h1>
<p>Main content</p>
```



```
<header></header>
<p>Main content</p>
```

```
<header>
```

```
<h1>Title</h1>
```

Composición de componentes



Árboles de componentes

ejem7

`app.component.ts`

```

import {Component} from
  '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl:
    './app.component.html'
})
export class AppComponent {}
```

`app.component.html`

```

<header></header>
<p>Main content</p>
```

`header.component.ts`

```

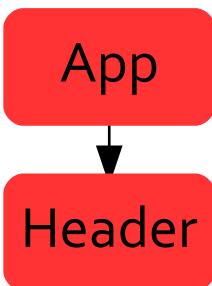
import {Component} from
  '@angular/core';

@Component({
  selector: 'header',
  templateUrl:
    './header.component.html'
})
export class HeaderComponent {}
```

`header.component.html`

```
<h1>Title</h1>
```

Composición de componentes



app.component.ts

```
import {Component} from
  '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl:
    './app.component.html'
})
export class AppComponent {}
```

app.component.html

```
<header></header>
<p>Main content</p>
```

Para incluir un componente se usa su selector

ejem7

Árboles de componentes

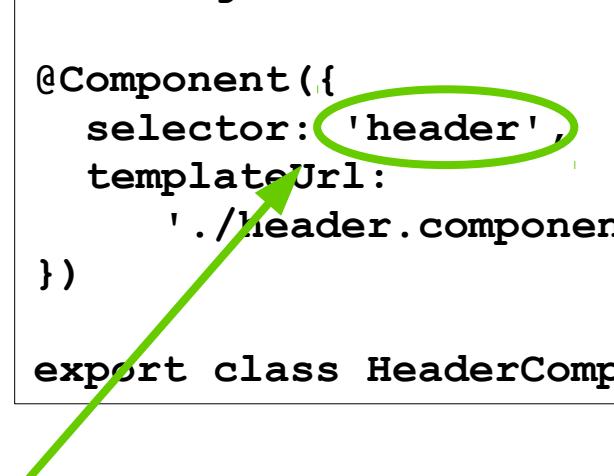
header.component.ts

```
import {Component} from
  '@angular/core';

@Component({
  selector: 'header',
  templateUrl:
    './header.component.html'
})
export class HeaderComponent {}
```

header.component.html

```
<h1>Title</h1>
```



Composición de componentes

ejem7

Árboles de componentes

- Al cargar la app en el navegador, en el árbol DOM cada componente incluye en su **elemento** el contenido de la **vista** (HTML)

```
└<body>
  └<app>
    └<header>
      <h1>Title</h1>
    </header>
    └<p>
      "Main content"
    </p>
  </app>
```

Composición de componentes

- Comunicación entre un **componente padre** y un **componente hijo**
 - Configuración de propiedades (Padre → Hijo)
 - Envío de eventos (Hijo → Padre)
 - Invocación de métodos (Padre → Hijo)
 - Con variable template
 - Inyectando hijo con `@ViewChild`
 - Compartiendo el mismo servicio (Padre ↔ Hijo)

Composición de componentes

ejem8

Configuración de propiedades (Padre → Hijo)

- El componente padre puede especificar **propiedades** en el componente hijo como si fuera un elemento **nativo HTML**

El título de <header> será el valor del atributo appTitle

```
<header [title]='appTitle'></header>
<p>Main content</p>
```

Composición de componentes

Configuración de propiedades (Padre → Hijo)

ejem8

app.component.ts

```
...
export class AppComponent {
  appTitle = 'Main Title';
}
```

header.component.ts

```
import {Component, Input} from
  '@angular/core';
...
export class HeaderComponent {

  @Input()
  private title: string;
}
```

app.component.html

```
<header [title]='appTitle'></header>
<p>Main content</p>
```

header.component.html

```
<h1>{{title}}</h1>
```

Composición de componentes

ejem8

Configuración de propiedades (Padre → Hijo)

app.component.ts

```
...
export class AppComponent {
  appTitle = 'Main Title';
}
```

app.component.html

```
<header [title]='appTitle'></header>
<p>Main content</p>
```

header.component.ts

```
import {Component, Input} from
  '@angular/core';
...
export class HeaderComponent {
  @Input()
  private title: string;
}
```

header.component.html

```
<h1>{{title}}</h1>
```

Composición de componentes

ejem8

Configuración de propiedades (Padre → Hijo)

app.component.ts

```
...
export class AppComponent {
  appTitle = 'Main Title';
}
```

header.component.ts

```
import {Component, Input} from
  '@angular/core';
...
export class HeaderComponent {

  @Input()
  private title: string;
}
```

app.component.html

```
<header [title]='appTitle'></header>
<p>Main content</p>
```

header.component.html

```
<h1>{{title}}</h1>
```

Composición de componentes

Envío de eventos (Hijo → Padre)

ejem9

- El componente hijo puede generar eventos que son atendidos por el padre como si fuera un elemento **nativo HTML**

La variable \$event apunta al evento generado

```
<header (hidden)='hiddenTitle($event)'></header>
<p>Main content</p>
```

Composición de componentes

ejem9

Envío de eventos (Hijo → Padre)

`app.component.ts`

```
...
export class AppComponent {
  hiddenTitle(hidden: boolean) {
    console.log("Hidden:" + hidden)
  }
}
```

`app.component.html`

```
<header (hidden)='hiddenTitle($event)'></header>
<p>Main content</p>
```

Composición de componentes

ejem9

Envío de eventos (Hijo → Padre)

app.component.ts

```
...
export class AppComponent {
  hiddenTitle(hidden: boolean) {
    console.log("Hidden:" + hidden)
  }
}
```

app.component.html

```
<header (hidden)='hiddenTitle($event)'></header>
<p>Main content</p>
```

Composición de componentes

ejem9

Envío de eventos (Hijo → Padre)

`app.component.ts`

```
...
export class AppComponent {
  hiddenTitle(hidden: boolean) {
    console.log("Hidden:" + hidden)
  }
}
```

Los eventos pueden tener valores que se capturan con `$event`

`app.component.html`

```
<header (hidden)='hiddenTitle($event)'></header>
<p>Main content</p>
```

Composición de componentes

Envío de eventos (Hijo → Padre)

ejem9

`header.component.ts`

```
import {Component, Output, EventEmitter} from '@angular/core';
...
export class HeaderComponent {

  @Output()
  hidden = new EventEmitter<boolean>();

  visible = true;

  click() {
    this.visible = !this.visible;
    this.hidden.next(this.visible);
  }
}
```

`header.component.html`

```
<h1 *ngIf="visible">Title</h1>
<button (click)='click()'>Hide/Show</button>
```

Composición de componentes

ejem9

Envío de eventos (Hijo → Padre)

`header.component.ts`

```
import {Component, Output, EventEmitter} from '@angular/core';
...
export class HeaderComponent {
    ...
    @Output()
    hidden = new EventEmitter<boolean>();
    visible = true;

    click() {
        this.visible = !this.visible;
        this.hidden.emit(this.visible);
    }
}
```

Se declara un atributo de tipo **EventEmitter** con la anotación **@Output**

Para **lanzar un evento** se invoca el método **emit(valor)**

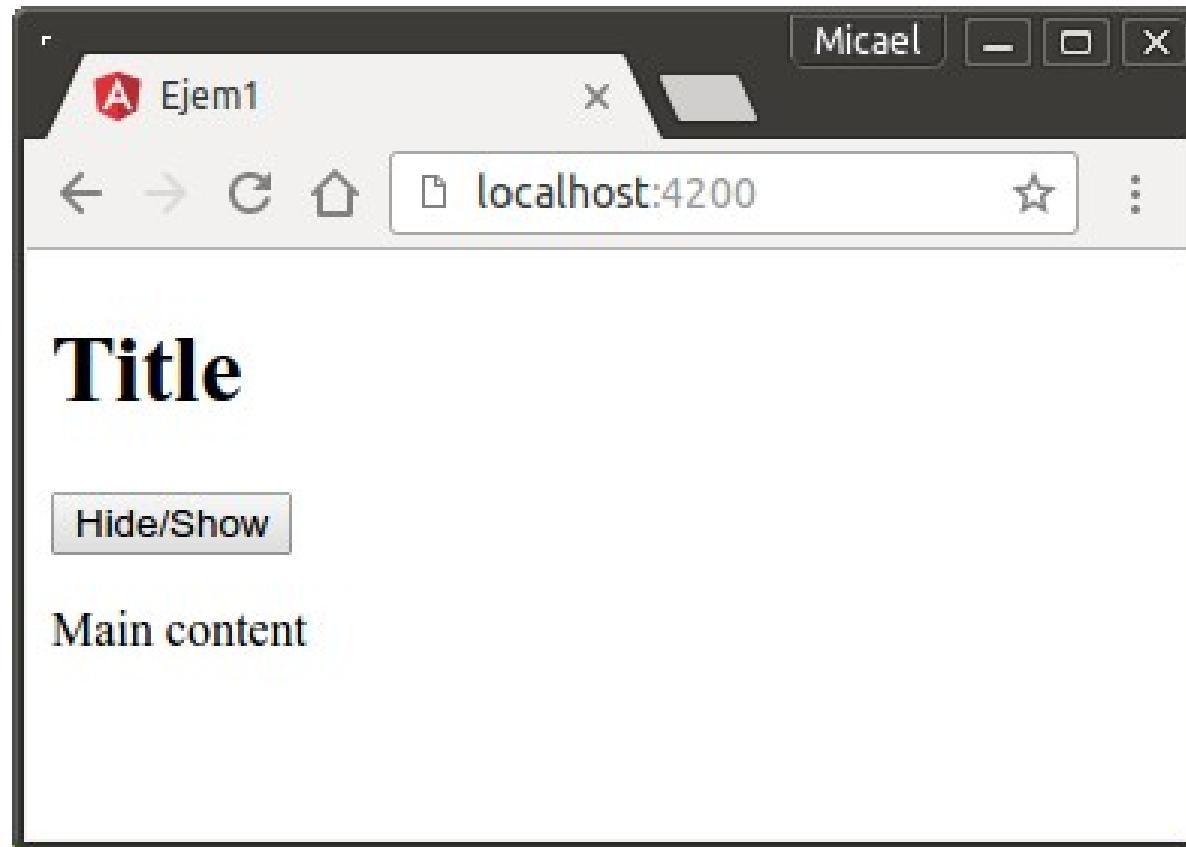
`header.component.html`

```
<h1 *ngIf="visible">Title</h1>
<button (click)='click()'>Hide/Show</button>
```

Composición de componentes

ejem9

Envío de eventos (Hijo → Padre)



Cliente REST e inyección de dependencias

Cliente REST e inyección de dependencias

- Angular 2 dispone de su propio **cliente de API REST**
- Es un objeto de la clase **Http**
- Usa **RxJS**



```
Http http = ...  
  
http.get(url).subscribe(  
  response => console.log(response.json()),  
  error => console.error(error)  
);
```

<https://angular.io/docs/ts/latest/guide/server-communication.html>
<https://angular.io/docs/ts/latest/api/http/index/Http-class.html>

Cliente REST e inyección de dependencias

- Angular 2 dispone de su propio **cliente de API REST**
- Es un objeto de la clase **Http**

```
Http http = ...  
  
http.get(url).subscribe(  
  response => console.log(response.json()) ,  
  error => console.error(error)  
);
```

El método **subscribe** recibe dos parámetros:
1) La función que se ejecutará cuando la petición sea **correcta**
2) La función que se ejecutará cuando la petición sea **errónea**

<https://angular.io/docs/ts/latest/guide/server-communication.html>
<https://angular.io/docs/ts/latest/api/http/index/Http-class.html>

Cliente REST e inyección de dependencias

- Angular 2 dispone de su propio **cliente de API REST**
- Es un objeto de la clase **Http**

```
Http http = ...
```

```
http.get(url).subscribe(  
  response => console.log(response.json()),  
  error => console.error(error)  
) ;
```

Para obtener la respuesta del servidor usamos el método **json()** del objeto **response**

<https://angular.io/docs/ts/latest/guide/server-communication.html>
<https://angular.io/docs/ts/latest/api/http/index/Http-class.html>

- ¿Cómo podemos acceder al objeto http?
 - Creando un objeto
 - Podríamos crear un objeto nuevo cada vez que nos haga falta
 - Esta opción **dificulta hacer tests automáticos** unitarios porque necesitaríamos que el servidor REST estuviese disponible y sería más difícil probar diferentes situaciones

- ¿Cómo podemos acceder al objeto http?
 - Pidiendo al framework que proporcione el objeto
 - Cuando la **aplicación** se ejecute, el objeto http sería un **objeto real** que hace peticiones al servidor REST
 - Cuando ejecutemos los **tests**, el objeto http puede ser un **sustituto (mock)** que se comporte como queramos en el test pero no haga peticiones reales

- ## Inyección de dependencias

- La técnica que permite solicitar objetos al **framework** se denomina inyección de dependencias
- Las **dependencias** que un módulo necesita son **inyectadas** por el sistema
- Esta técnica se ha hecho muy popular en el desarrollo de *back-end* en frameworks como **Spring** o **Java EE**

<https://angular.io/docs/ts/latest/guide/dependency-injection.html>

Cliente REST e inyección de dependencias

- Para usar un objeto `http` tenemos que usar la inyección de dependencias

```
import { Component } from '@angular/core';
import { Http } from '@angular/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  constructor(private http: Http) { }

  search(title: string) {
    ....
  }
}
```

Cliente REST e inyección de dependencias

- Para usar un objeto **http** tenemos que usar la **inyección de dependencias**

```
import { Component } from '@angular/core';
import { Http } from '@angular/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  constructor(private http: Http) { }

  search(title: string) {
    ....
  }
}
```

Definimos un **parámetro** **Http** en el **constructor** de un componente

Cuando Angular construya el componente, **inyectará** el objeto http solicitado

ejem10

- Ejemplo de buscador libros en Google Books



ejem10

- Ejemplo de buscador libros en Google Books

app.component.html

```
<h1>Google Books</h1>

<input #title type="text">

<button
(click)="search(title.value); title.value=' '">
Buscar</button>

<p *ngFor="let book of books">{ {book} }</p>
```

app.component.ts

```
import { Component } from '@angular/core';
import { Http } from '@angular/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  private books: string[] = [];

  constructor(private http: Http) { }

  search(title: string) {

    this.books = [];

    let url = "https://www.googleapis.com/books/v1/volumes?q=intitle:" + title;
    this.http.get(url).subscribe(
      response => {
        let data = response.json();
        for (var i = 0; i < data.items.length; i++) {
          let bookTitle = data.items[i].volumeInfo.title;
          this.books.push(bookTitle);
        }
      },
      error => console.error(error)
    );
  }
}
```

ejem10

```
import { Component } from '@angular/core';
import { Http } from '@angular/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  private books: string[] = [];

  constructor(private http: Http) { }

  search(title: string) {

    this.books = [];

    let url = "https://www.googleapis.com/books/v1/volumes?q=intitle:"+title;
    this.http.get(url).subscribe(
      response => {
        let data = response.json();
        for (var i = 0; i < data.items.length; i++) {
          let bookTitle = data.items[i].volumeInfo.title;
          this.books.push(bookTitle);
        }
      },
      error => console.error(error)
    );
  }
}
```

ejem10

- Peticiones POST

```
let data = { ... }
this.http.post(url, data).subscribe(
  response => console.log(response),
  error => console.error(error)
);
```

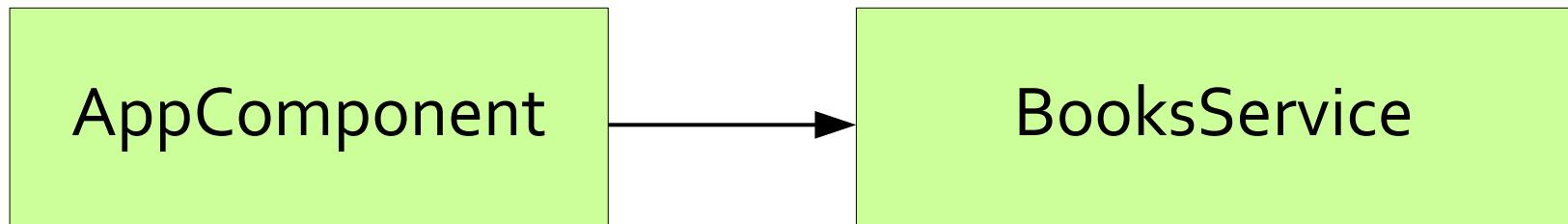
- Peticiones PUT

```
let data = { ... }
this.http.put(url, data).subscribe(
  response => console.log(response),
  error => console.error(error)
);
```

Servicios

Servicios

- En Angular2 se puede **modularizar** una aplicación
- Cada módulo es una **clase**
- Las dependencias entre **módulos** se gestionan con la inyección de dependencias
- A los módulos se les llama “**Servicios**”



Servicios

ejem11

¿Cómo se implementa un servicio?

`books.service.ts`

```
Import { Injectable } from '@angular/core';

@Injectable()
export class BooksService {

  getBooks(title: string) {
    return [
      'Aprende Java en 2 días',
      'Java para torpes',
      'Java para expertos'
    ];
  }
}
```

Servicios

ejem11

¿Cómo se implementa un servicio?

`books.service.ts`

```
Import { Injectable } from '@angular/core';

@.Injectable()
export class BooksService {

  getBooks(title: string) {
    return [
      'Aprende Java en 2 días',
      'Java para torpes',
      'Java para expertos'
    ];
  }
}
```

Servicios

ejem11

¿Cómo se implementa un servicio?

app.component.ts

```
import { Component } from '@angular/core';
import { BooksService } from './books.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  private books: string[] = [];

  constructor(private booksService: BooksService) {}

  search(title: string) {
    this.books = this.booksService.getBooks(title);
  }
}
```

ejem11

¿Cómo se implementa un servicio?

app.component.ts

```
import { Component } from '@angular/core';
import { BooksService } from './books.service';

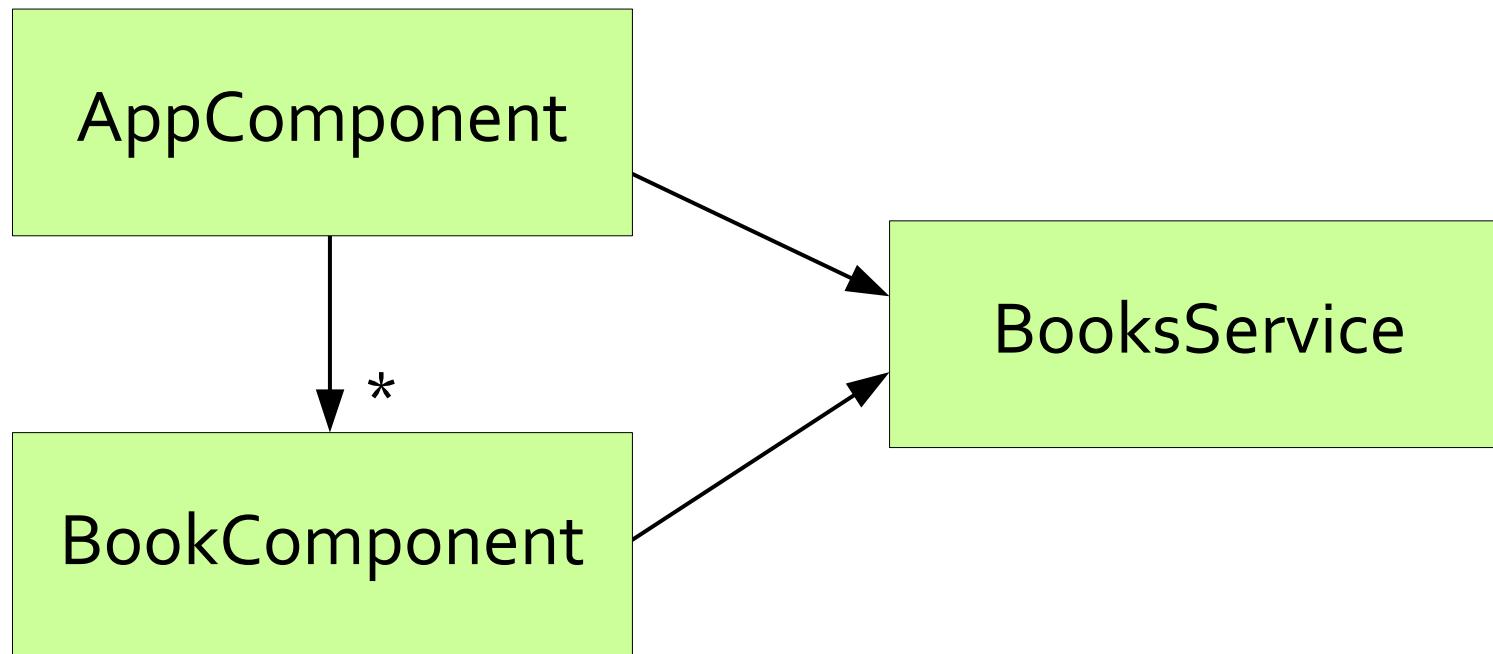
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {

  private books: string[] = [];

  constructor(private booksService: BooksService) {}

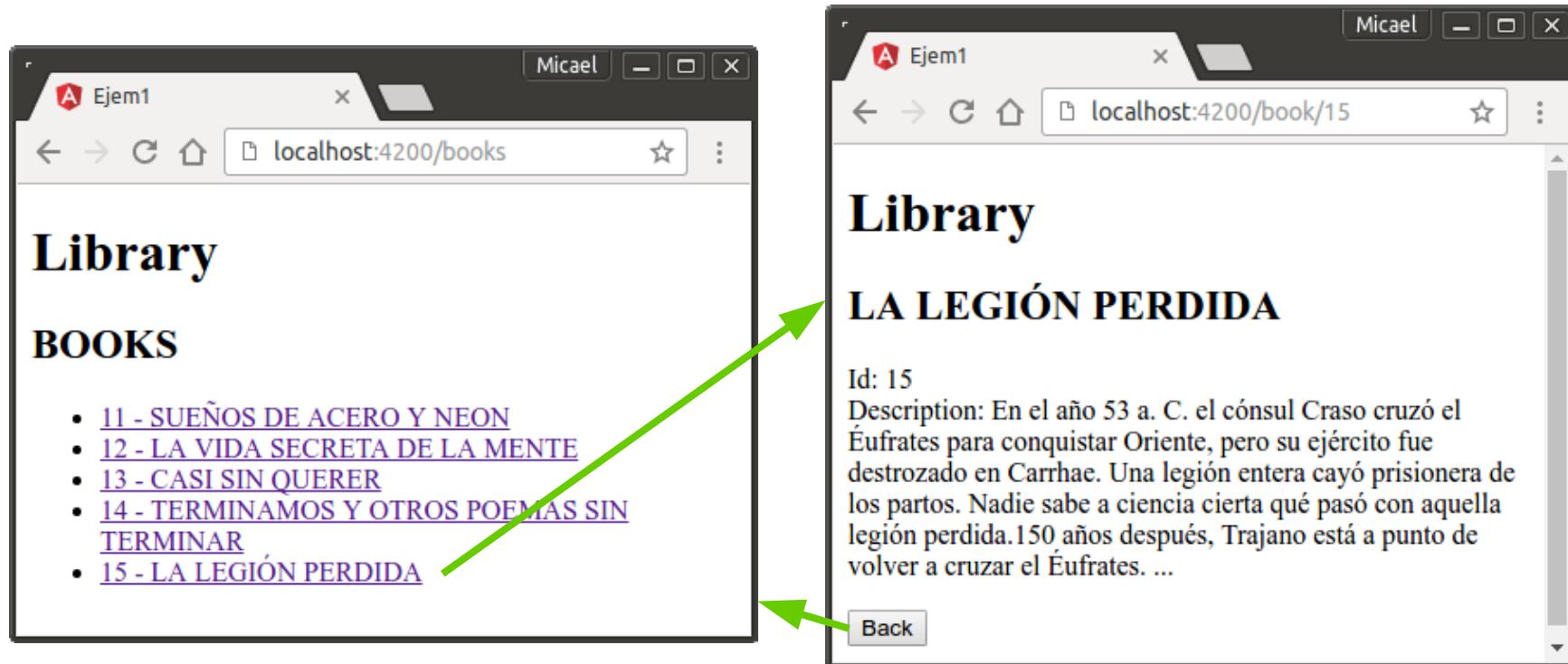
  search(title: string) {
    this.books = this.booksService.getBooks(title);
  }
}
```

Compartir servicios entre componentes



Aplicaciones Multipágina Router

Aplicaciones multipágina: Router



Aplicaciones multipágina: Router

- El **componente principal**
 - Parte fija
 - Parte que cambia (**<router-outlet>**)
- En **app.routing.ts** se define qué **componente** se muestra para cada **URL**
- Existen **links especiales** para navegar dentro de la aplicación web (**[routerLink]**)
- Desde el código se puede navegar (**Router**)

<https://angular.io/docs/ts/latest/guide/router.html>

ejem14

Componente principal

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1 class="title">Library</h1>
    <router-outlet></router-outlet>
  `
})
export class AppComponent { }
```

ejem14

Componente principal

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template:
    `\
      <h1 class="title">Library</h1>
      <router-outlet></router-outlet>
    `)
export class AppComponent { }
```

Zona que cambia en función de la URL

ejem14

Configuración de las rutas

app.routing.ts

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

import { BookListComponent } from './book-list.component';
import { BookDetailComponent } from './book-detail.component';

const appRoutes = [
  { path: 'book/:id', component: BookDetailComponent, },
  { path: 'books', component: BookListComponent },
  { path: '', redirectTo: 'books', pathMatch: 'full' }
]

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

Aplicaciones multipágina: Router

ejem14

Configuración de las rutas

app.routing.ts

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

import { BookListComponent } from './book-list.component';
import { BookDetailComponent } from './book-detail.component';

const appRoutes = [
  { path: 'book/:id', component: BookDetailComponent },
  { path: 'books', component: BookListComponent },
  { path: '', redirectTo: 'books', pathMatch: 'full' }
]

export const routes: Routes = [
  ...appRoutes
]
```

Para cada URL se indica el **componente** que será visualizado

ejem14

Configuración de las rutas

app.module.ts

```
...
import { routing } from './app.routing';

@NgModule({
  declarations: [AppComponent,
    BookDetailComponent, BookListComponent],
  imports: [BrowserModule, FormsModule,
    HttpModule, JsonpModule, routing],
  bootstrap: [AppComponent],
  providers: [BookService]
})
export class AppModule { }
```

Aplicaciones multipágina: Router

BookListComponent

ejem14

book-list.component.ts

```
...
@Component({
  template: `
    <h2>BOOKS</h2>
    <ul>
      <li *ngFor="let book of books">
        <a [routerLink]=["'/book',book.id]">
          {{book.id}}-{{book.title}}
        </a>
      </li>
    </ul>`
})
export class BookListComponent {

  books: Book[];

  constructor(service: BookService) {
    this.books = service.getBooks();
  }
}
```

Aplicaciones multipágina: Router

BookListComponent

ejem14

book-list.component.ts

```
...
@Component({
  template:
    <h2>BOOKS</h2>
    <ul>
      <li *ngFor="let book of books">
        <a [routerLink]=["'/book',book.id"]>
          {{book.id}}-{{book.title}}
        </a>
      </li>
    </ul>
})
export class BookListComponent {
  books: Book[];
  constructor(service: BookService) {
    this.books = service.getBooks();
  }
}
```

En vez de `href`, los links usan `[routerLink]`. La URL se puede indicar como un string (**completa**) o como un array de strings si hay **parámetros**

Aplicaciones multipágina: Router

BookDetailComponent

ejem14

book-detail.component.ts

```
...
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button></p>`})
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {

    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Aplicaciones multipágina: Router

ejem14

book-detail.component.ts

BookDetailComponent

```
...
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button></p>`,
})
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {
    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Para acceder a los parámetros desde el componente usamos el servicio **ActivatedRoute**

Aplicaciones multipágina: Router

ejem14

book-detail.component.ts

BookDetailComponent

```
...
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button> <a href="#">Edit</a> <a href="#">Delete</a></p>
`)
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {
    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Obtenemos un **snapshot** de los parámetros y accedemos al parámetro **id**



Aplicaciones multipágina: Router

BookDetailComponent

ejem14

book-detail.component.ts

```
...
import { Router, ActivatedRoute } from '@angular/router';

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button></p>`})
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, ser...
```

let id = activatedRoute.snapshot.params['id'];
this.book = service.getBook(id);
}
gotoBooks() { this.router.navigate(['/books']);}

Para navegar desde código usamos la dependencia **Router** y el método **navigate**

Aplicaciones multipágina: Router

• Funcionalidades avanzadas

- Rutas para un componente concreto (no para toda la app)
- Ejecutar código al salir de una pantalla
 - Si el usuario navega a otra página “sin guardar” se le puede preguntar si realmente desea descartar los cambios o abortar la navegación
- Verificar si se puede ir a una nueva pantalla
 - Generalmente se comprueba si el usuario tiene permisos para hacerlo
- Carga perezosa de componentes (*lazy loading*)
- Animaciones

<https://angular.io/docs/ts/latest/guide/router.html>

Library

Books

- [SUEÑOS DE ACERO Y NEON](#)
- [LA VIDA SECRETA DE LA MENTE](#)
- [CASI SIN QUERER](#)
- [TERMINAMOS Y OTROS POEMAS SIN TERMINAR](#)
- [LA LEGIÓN PERDIDA](#)

New book

Library

Book "SUEÑOS DE ACERO Y NEON"

Los personajes que protagonizan este relato sobreviven en una sociedad en decadencia a la que, no obstante, lograrán devolver la posibilidad de un futuro. En un mundo dominado por las grandes corporaciones, solo un hombre, Thompson, detective privado deslenguado y vividor, pero de gran talento y sentido d...

Remove Edit Back

localhost:4200 dice:

Do you want to remove this book?

Evita que esta página cree cuadros de diálogo adicionales.

Cancelar Aceptar

Library

Book ""

Title:

Description:

Cancel Save

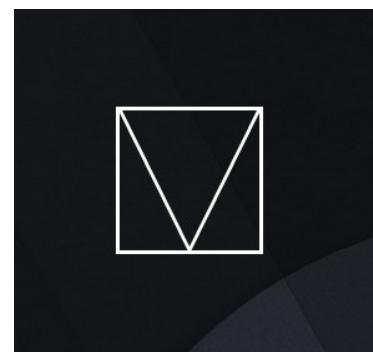
Librerías de Componentes

Librerías de componentes

- Angular 2 **no proporciona** componentes de alto nivel, usa HTML y CSS
- Se pueden usar cualquier librería de componentes que **sólo tienen CSS**: Bootstrap, Semantic ui, Google Material Design Lite...



<http://getbootstrap.com/>



<http://www.getmdl.io/>



<http://semantic-ui.com/>

Librerías de componentes

- **No se recomienda** usar directamente librerías gráficas JavaScript con Angular 2:
 - **JQuery:** Es mejor modificar el DOM con plantillas u otros mecanismos avanzados de Angular2
 - **JavaScript de Bootstrap:** Está basado en jQuery. Es mejor usar **ng2-bootstrap**, componentes bootstrap adaptados a Angular2
 - **Otras librerías:** Es mejor usar aquellas con diseñadas para Angular 2 o con adaptadores para Angular2 para evitar problemas de rendimiento y en la construcción de la app

Librerías de componentes

- **ng2-bootstrap:** Componentes de bootstrap reimplementados en Angular 2 (Valor software)



<http://valor-software.com/ng2-bootstrap/>

Librerías de componentes

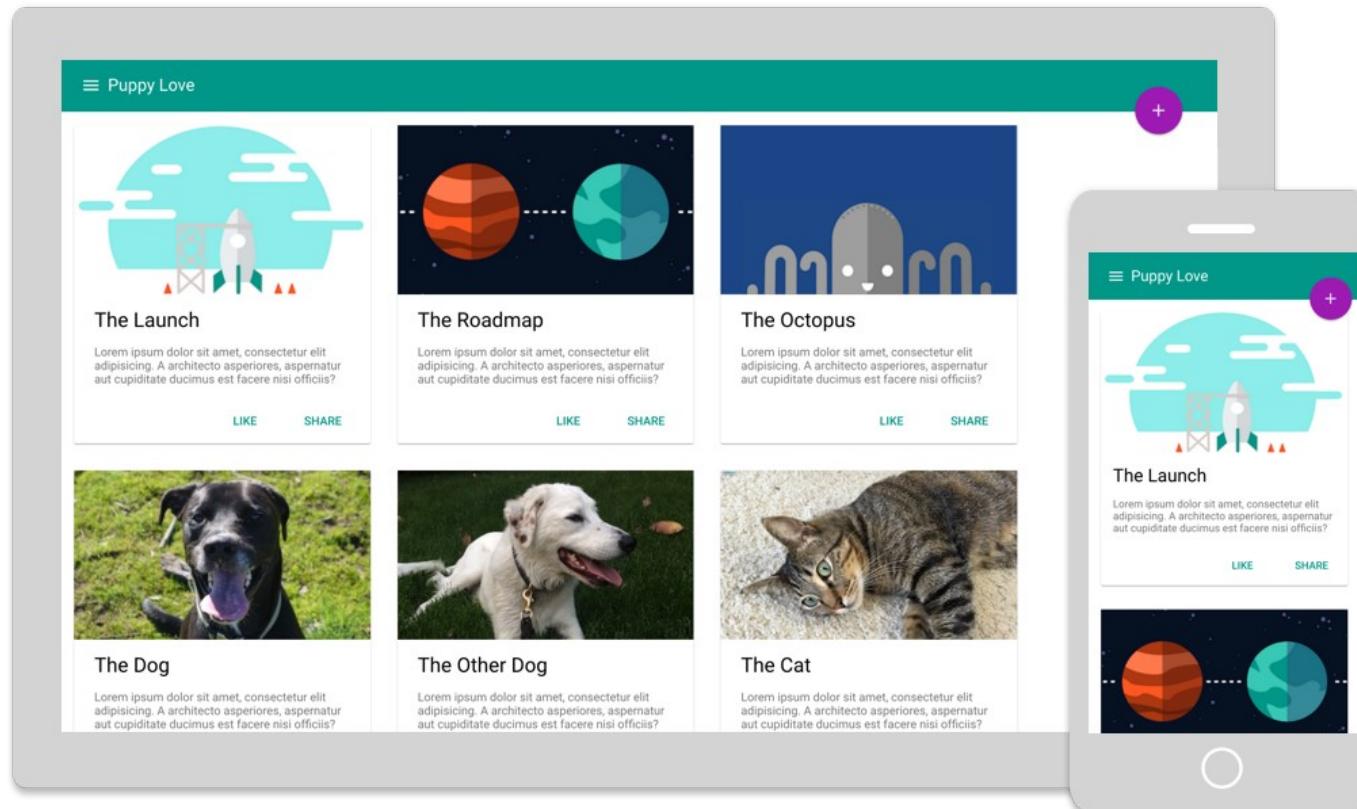
- **ng-bootstrap:** Componentes de bootstrap reimplementados en Angular 2 (equipo de Angular)



<https://ng-bootstrap.github.io/>

Librerías de componentes

- Material Design Angular 2: Librería de componentes



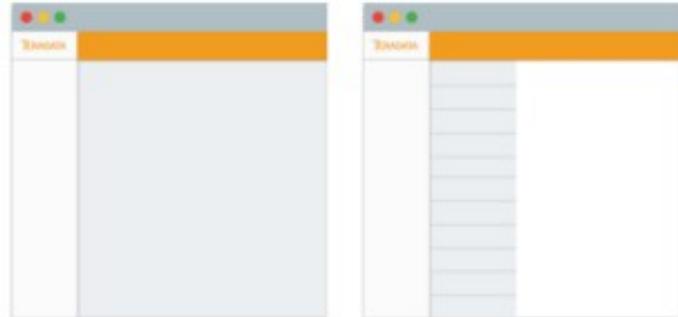
<https://github.com/angular/material2>

Librerías de componentes

- Covalent

Layout Options

We offer 4 Material Design layouts for you



Nav View

Nav List

Teradata Components

Stepper (Wizard)
A sequence of logical & numbered steps

Change Event: Changed from step: 1 to step: 2
Active/Deactive Event for Step 1: Deactive event emitted.

1 Basic Usage
Toggle between active and inactive and emit events.

A Required Step
Toggle between active and inactive while in required state.

STEP CONTENT GOES HERE

STEP SUMMARY GOES HERE

Complete State
Toggle between active and inactive while in complete state.

Toggle Disable All Steps Toggle Active Step 1

TdStepsComponent
How to use this component



<https://teradata.github.io/covalent/>

Librerías de componentes

- ag-grid: Tabla con controles avanzados

Type text to filter...

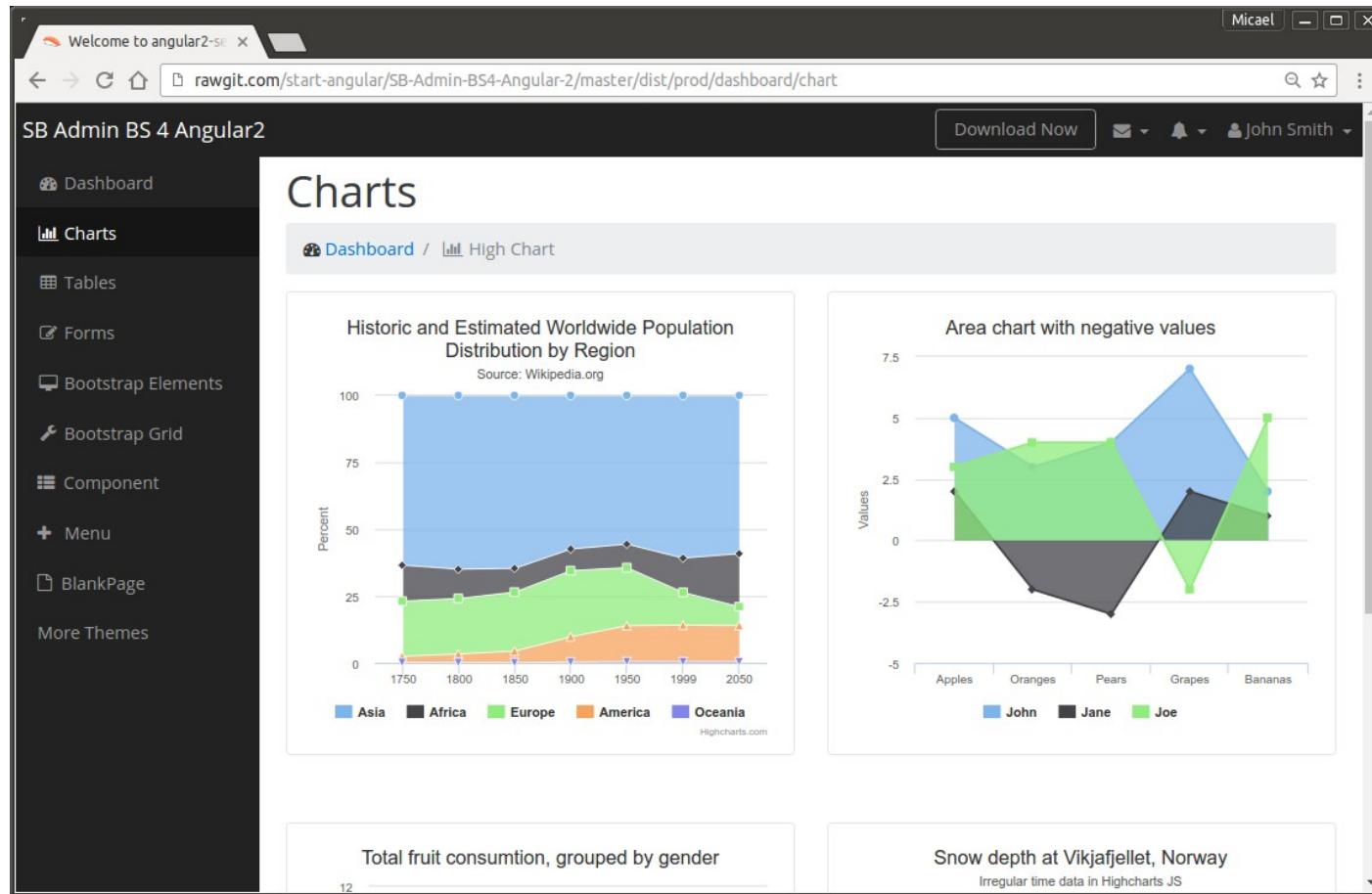
Example Dataset Showing 10.000 / 10.000 rows(s)

#	Employee		IT Skills		Contact		
	Name	▼ Country	Skills	Proficiency	Mobile	Land-line	
<input type="checkbox"/>	Jessica Kade	Uruguay		76%	+873 117 227 477	+141 627 522 455	4819 Honey Treasure Park, A
<input type="checkbox"/>	Chloe Brennan	Uruguay		86%	+333 840 805 1062	+464 546 279 888	5313 Clear Willow Route, Am
<input type="checkbox"/>	Mia Corbin	Uruguay		61%	+224 1092 327 055	+783 3410 5610 816	7390 Harvest Crest, Mosquito
<input checked="" type="checkbox"/>	Emily Hudson	Uruguay		99%	+886 659 691 561	+634 112 676 1059	6918 Cotton Pine Corner, Ken
<input type="checkbox"/>	Isabelle Donovan	Uruguay		78%	+241 221 394 845	+097 641 829 585	7619 Tawny Carrefour, Senlac
<input checked="" type="checkbox"/>	Isla Dalton	Uruguay		15%	+338 085 844 1010	+326 734 1102 156	7619 Tawny Carrefour, Senlac
<input checked="" type="checkbox"/>	Ava Cadwell	Uruguay		44%	+1102 171 194 147	+145 153 485 213	6686 Lazy Ledge, Two Rock,
<input checked="" type="checkbox"/>	Evie Griffin	Uruguay		16%	+6108 968 379 632	+799 044 511 914	3685 Rocky Glade, Showtuck
<input type="checkbox"/>	Poppy Jett	Uruguay		49%	+037 9104 647 7110	+243 9910 258 929	9218 Crystal Highway, Picklev
<input type="checkbox"/>	Lily Jagger	Uruguay		10%	+413 527 778 358	+7910 465 373 918	7619 Tawny Carrefour, Senlac
<input type="checkbox"/>	Lucy Cole	Uruguay		90%	+641 937 319 379	+664 143 819 999	9218 Crystal Highway, Picklev
<input type="checkbox"/>	Chloe Corbin	Uruguay		10%	+722 489 590 185	+536 566 115 237	6683 Colonial Street, Swan Ri
<input type="checkbox"/>	Poppy Chandler	Uruguay		17%	+557 530 1020 238	+974 945 987 804	7619 Tawny Carrefour, Senlac
<input type="checkbox"/>	Freya Cadwell	Uruguay		27%	+626 876 473 799	+809 203 617 347	6918 Cotton Pine Corner, Ken
<input type="checkbox"/>	Poppy Cadwell	Uruguay		73%	+1056 876 820 921	+242 674 779 705	2347 Indian Boulevard, Frisbe
<input type="checkbox"/>	Lucy Hudson	Uruguay					

<https://www.ag-grid.com/>

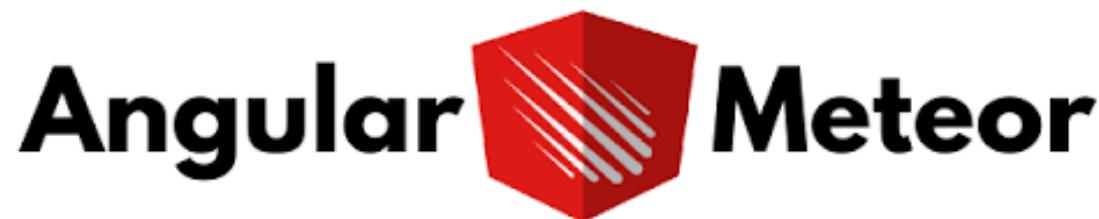
Librerías de componentes

- **SB Admin 2.0 ng2:** Tema completo para admin

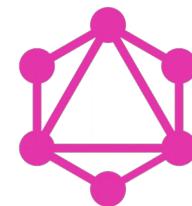


<https://github.com/start-angular/SB-Admin-BS4-Angular-2>

Ecosistema



NativeScript



Firebase

Conclusiones

Conclusiones

- **Introducción a Angular 2...**

- Un framework de desarrollo apps **SPA**
- Se recomienda **TypeScript**, más preparado para grandes aplicaciones (pero puede usar ES5, ES6 y Dart)
- **Orientado a componentes**, con **inyección de dependencias y templates**
- **No es compatible** con Angular 1, pero comparte su filosofía
- Mucho mejor **rendimiento** que Angular 1
- Seguramente será uno de los frameworks **más usados** para desarrollo web en los **próximos años**

Conclusiones

- **Angular 2 es mucho más..**
 - **Validación de formularios** (con NgForm y NgControl)
 - **Testing unitario y de integración** (Jasmine, Karma, Protractor, Inyección de dependencias de testing...)
 - **Carga bajo demanda de componentes** (para acelerar la carga inicial de la aplicación)
 - **Gestión del estado al estilo Redux** (ngrx)
 - **Animaciones**

Conclusiones

- **Angular 2 es mucho más..**
 - **Aplicaciones con múltiples @NgModule** (para estructurar componentes y dependencias)
 - **Optimización de la app en producción:** Compilador de templates a TS (angular-compiler), eliminación de funcionalidades de la librería no usadas... (tree shaking)
 - **Angular Universal:** Renderizado en el servidor para optimizar la descarga inicial

Conclusiones



Micael Gallego

2.0

CURSO GRATUITO



CAMPUS MADRID

26 de Septiembre
11:00 a 17:30

<https://www.youtube.com/watch?v=YVVjXpquzBE>