

Agenda: Angular (2.x/4.x)

sábado, 9 de septiembre de 2017 13:07



The banner features a large teal 'J' shape on the right side. In the center, there's a white rectangular box containing the text 'ANGULAR 2,0' in bold black letters. Above this text is the date '11, 12, 13, 14 y 15 de Septiembre de 2017'. Below the date are two logos: 'indra' (with a circular icon) and 'icono' (with a stylized 'o'). To the right of the 'ANGULAR 2,0' text is a logo for 'ESCUELA TECNOLOGÍA' featuring a gear icon.

ÍNDICE

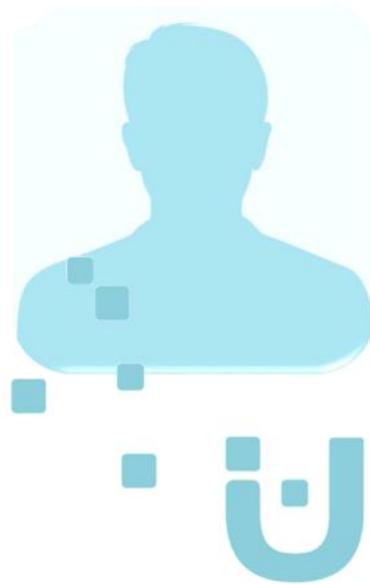
Duración: 30 horas

Objetivo: Los participantes serán capaces de desarrollar y entender una aplicación básica realizada en Angular v.2.

Requisitos:

Conocimientos de JavaScript, HTML, CSS y JQUERY.

- Introducción a Angular 2
- Introducción a *TypeScript*
- Herramientas de Desarrollo
- Módulos
- Plantillas
- Formularios
- Servicios
- Acceso al servidor
- Enrutamiento y navegación



■ PROFESOR
Alejandro Cerezo Lasne

■ VER PERFIL COMPLETO:

 <https://www.linkedin.com/in/alejandrocerezo/>

■ CONTACTO

 training@iconotc.com
alce65@hotmail.es



indra



Avda. de Bruselas 35
28108 Alcobendas,
Madrid España
T +34 91 480 50 00
F +34 91 480 50 80
www.indracompany.com

Desarrollo de la agenda

sábado, 9 de septiembre de 2017 14:09

Contenido:

- Introducción a Angular 2
- Introducción a TypeScript
- Herramientas de Desarrollo

- Módulos
- Plantillas
- Formularios
- Servicios

- Acceso al servidor
- Enrutamiento y navegación



[Icono Training Consulting: iconotc.com](http://iconotc.com)

INTRODUCCIÓN A Angular 2

- Características de Angular 2

TECNOLOGÍAS IMPLICADAS Y *TypeScript*

- *TypeScript*
- Preparando el entorno: *npm*, *web pack* (*), transpilación,...
- Gestión de la configuración: angular-cli

ELEMENTOS PRINCIPALES

- Módulos
- **Componentes**
- Directivas
- Pipes
- *Data binding*
- Servicios e inyección de dependencias
- Formularios
- ...Uso del enrutador (*router*)
- ...Conexión con el servidor

Single Page Applications

- Configuración, acción y presentación
- Enrutado en base a componentes
- Rutas anidadas y con parámetros

Comunicaciones HTTP con APIs REST

- Operaciones HTTP asíncronas con promesas
- Observables con Rx.js
Subscripción, transformación y cancelación de streams
- Seguridad e infraestructura de comunicaciones

Angular. Primera parte

sábado, 9 de septiembre de 2017 13:22

- Introducción a Angular
 - *Frameworks en JS*
 - Presentación de AngularJS y Angular
- Tecnologías implicadas
 - Entorno de trabajo
 - Instalación e inicio. Angular cli
 - ES6
 - *TypeScript*

Navegadores.
Editores de código
Gestión de versiones. GIT
NodeJS y npm. Empaquetado

Formas de creación de proyectos
Arquitectura del proyecto (*Scaffolding*). Angular-cli

Frameworks en JS

sábado, 9 de septiembre de 2017 13:32

Bibliotecas o frameworks

- facilitan el desarrollo
- automatizan procesos
- aumentan la eficacia
- mejoran el producto

*jQuery
Underscore.js
MooTools
Prototype
Google Web Toolkit (de Java a JS)
YUI*

*Ext JS
Vue.js
SAP - OpenUI5*

AngularJS / Angular
*BackboneJS
Ember.js
React.js*

*AccDC
Ample SDK
Atoms.js
DHTMLX
Dojo
Echo3
Enyo
Handlebars
Kendo UI
Knockout.js
D3.js - Kinetic.js*

*Meteor
PhoneJS
Pyjamas
qooxdoo
Rialto
SmartClient & SmartGWT
Socket.IO
SproutCore
Wakanda
ZK
Webix*



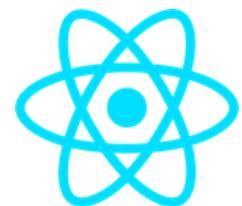
BackboneJS

<http://backbonejs.org/>



Ember.js

<http://emberjs.com/>



React.js

<http://emberjs.com/>



AngularJS

<https://angularjs.org>

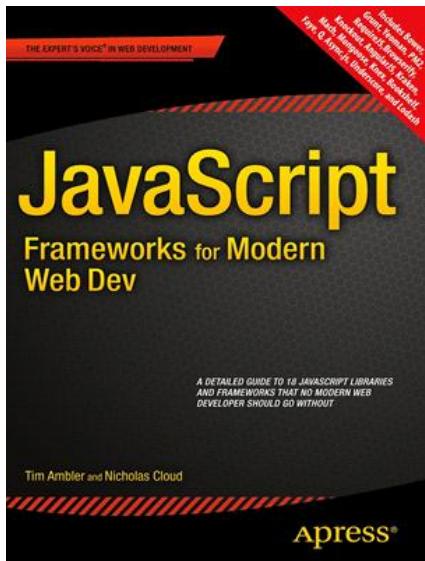


La elección de uno de ellos depende de los objetivos de cada proyecto



<http://todomvc.com/>

Frameworks interrelacionados



JavaScript Frameworks for Modern Web Dev
Tim Ambler & Nicholas Cloud
Apress, 2015

Contents at a Glance

About the Authors.....	xix
About the Technical Reviewer.....	xxi
Acknowledgments	xxiii
Introduction	xxv
■ Chapter 1: Bower	1
■ Chapter 2: Grunt	11
■ Chapter 3: Yeoman	37
■ Chapter 4: PM2	53
■ Chapter 5: RequireJS	73
■ Chapter 6: Browserify.....	101
■ Chapter 7: Knockout.....	121
■ Chapter 8: AngularJS	155
■ Chapter 9: Kraken.....	191
■ Chapter 10: Mach	251
■ Chapter 11: Mongoose.....	297
■ Chapter 12: Knex and Bookshelf	345
■ Chapter 13: Faye.....	381



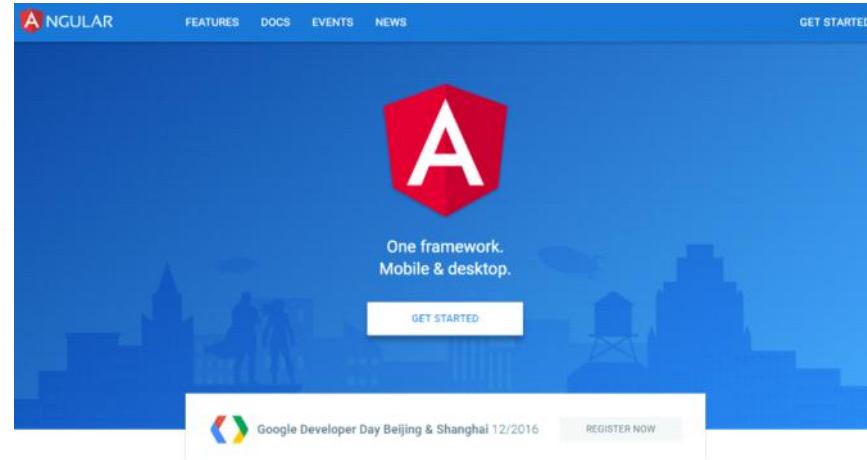
Angular

Introducción a Angular

sábado, 9 de septiembre de 2017 16:43



<https://angularjs.org/>



<https://angular.io/>

Orígenes y desarrollo

Misko Hevery

Adam Abrons



- proyecto de **código abierto**, realizado íntegramente en JavaScript
- creado en **2009**, por Misko Hevery de *Brat Tech LLC* y Adam Abrons
- está mantenido por **Google** y junto con una amplia y creciente **comunidad**.
- puede coexistir con **otros frameworks** (e.g JQuery, Bootstrap, *Material Design*)
- se ha hecho muy popular desde finales de 2012 hasta ahora,
- especialmente en asociación con otras tecnologías, dando lugar a **MEAN**

MongoDB
ExpressJS
AngularJS
NodeJS



<http://mean.io/#!/>

Se habla de una nueva *technology fullstack* como antes era *xAMP* (Apache + MySQL + PHP)

Se traduce a aplicaciones **JavaScript de principio a fin (End-to-End)**



MEAN is an opinionated fullstack javascript framework - which simplifies and accelerates web application development.

Get MEAN by running...

```
$ sudo npm install -g mean-cli
$ mean init yourNewApp
```

LATEST RELEASE: v0.5.5 | LATEST COMMIT: Nov 23, 2015 | FORKS: 2296

Ejemplos de uso

madewithANGULAR

<https://www.madewithangular.com/#/>

Communication



Education

SEE ALL





AngularJS 1.x

- extiende directamente las funcionalidades **HTML**
 - utiliza como patrón arquitectónico **MVC** (Modelo, Vista, Controlador) o similares (estrictamente sería MV* o MVW (*Model-View-Whatever*))
 - está especialmente orientado a la creación de aplicaciones **SPA** (*Single-Page Applications*).
 - es muy eficiente: promueve el uso **patrones** de diseño de software
 - permite crear **tests unitarios** y *End-to-End* de forma sencilla empleando *Jasmine* y *Karma*
 - al estar exclusivamente orientado a la lógica, es un *framework* muy liviano: no incluye elementos gráficos ni CSS.
- Se complementa muy bien con *Bootstrap* o *Material Design*

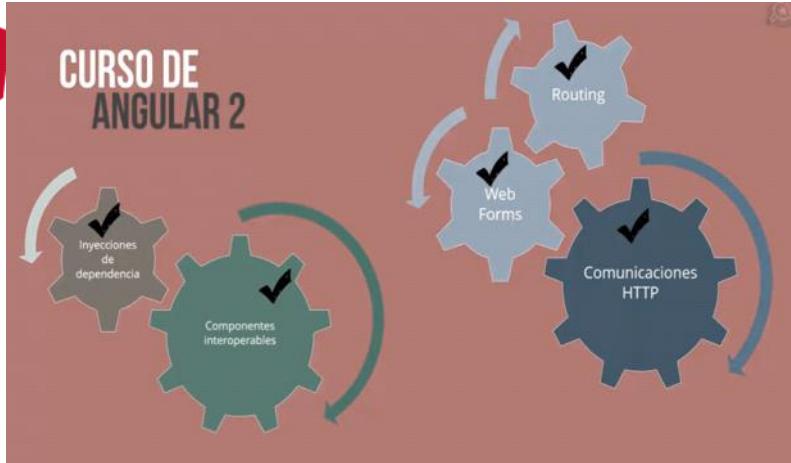
Aplicaciones cada vez más ambiciosas

Angular 2.x
Angular 4.x

- amplia el modelo de **extender** las funcionalidades **HTML** empleando **componentes**
 - Árboles de componentes Web
 - Interconexiones entre ellos
 - Cada uno su propia interface I/O
 - Inyección de dependencias totalmente renovado
- con ello se modifica la forma de emplear el patrón arquitectónico **MVC** (Modelo, Vista, Controlador) o similares (estrictamente sería MV* o MVW (*Model-View-Whatever*))
- sigue estando especialmente orientado a la creación de aplicaciones **SPA** (*Single-Page Applications*).



CURSO DE ANGULAR 2



- Angular 2 Versión final: Septiembre 2016
Angular 4 : Abril 2017
- Está implementado desde cero no como una evolución de Angular 1
- Angular 2 no es compatible con Angular 1: no existe el `$scope`
- La documentación de Angular 1 no sirve para Angular 2

Tabula Rasa



Funcionalidades en Angular 2

- Inyección de dependencias
- Servicios
- Cliente http (APIs REST)
- Navegación por la app (*Router*)
- Animaciones
- Internacionalización
- Soporte para tests unitarios y e2e
- Librerías de componentes
- Renderizado en el servidor (Angular Universal)





Características: MVC

Patrón arquitectónico (según otros autores **patrón de diseño**)
separación del código de los programas dependiendo de su responsabilidad

Se basa en las ideas claves
en el desarrollo de la
ingeniería del software

- **reutilización de código**
(*code reuse*, Douglas McIlroy, 1968)
- **separación de conceptos**
(*separation of concerns*,
Edsger W. Dijkstra, 1974)

Fue introducido por el científico
noruego Trygve Reenskaug
cuando trabajaba con Smalltalk-76 en el
Xerox Palo Alto Research Center (PARC)

Más tarde fue re implementado
en Smalltalk-80

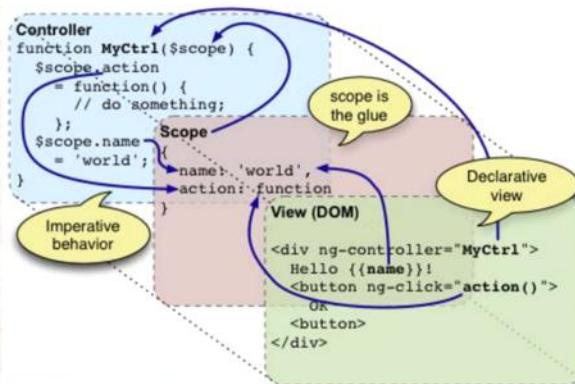


Model - View - Whatever



- **Vistas:** Será la representación de los datos o la información, es decir, el código del interfaz de usuario, básicamente el DOM/HTML/CSS .
- **Controladores:** Se encargarán de la lógica de la aplicación, incluyendo "Factorías" y "Servicios" para mover datos contra servidores o memoria local en HTML5.
- **Modelo o Modelo de la vista,** según se emplee la variante del patrón MVC o MVVC.

El modelo es la estructura lógica que subyace a los datos. Asociado a él se define el **scope**, responsable de detectar los cambios en el modelo y proporciona el contexto a las plantillas.



Futuro de HTML: Web Components

Web Components

conjunto de **estándares** que, permiten crear y utilizar elementos HTML personalizados.

Se puede así ampliar el “vocabulario” de HTML con elementos propios

En ellos está trabajando la W3C. Ya se soportan en algunos navegadores (Chrome) y esta disponible un *polyfile* para que puedan usarse en otros.

Constan de cuatro especificaciones:

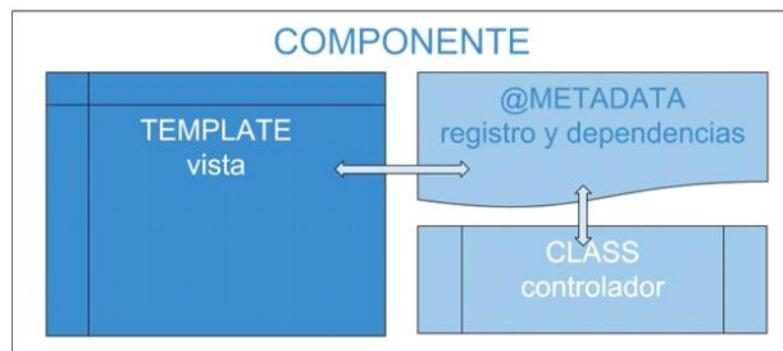
- **Custom elements:** Nos permite definir nuevos elementos HTML.
- **Templates:** Sistema de plantillas nativas en el navegador.
- **Shadow DOM:** DOM scope.
- **HTML Imports:** Carga de documentos HTML.

Continuando con Web Components

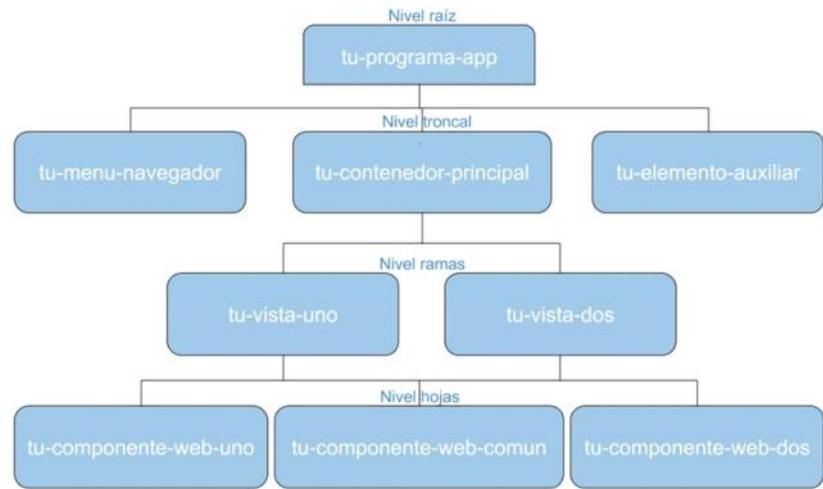


Componentes en Angular

- **Elemento personalizado:** Nos permite definir nuevos elementos HTML.
- Cada uno de ellos con su **template**: Sistema de plantillas nativas en el navegador.
- **Shadow DOM**: contenedor no visible
- **ciclo de vida** bien definido
- evolución de los componente definidos en Angular 1.5

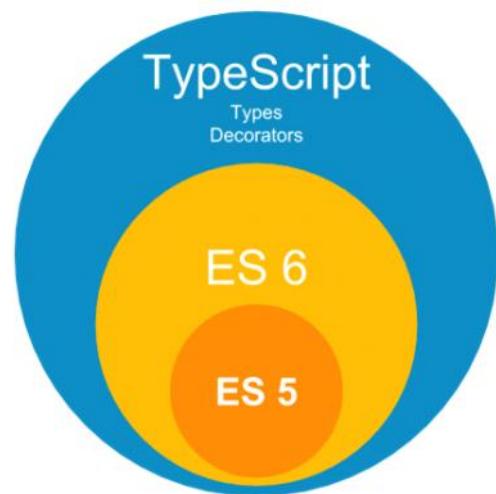


Árbol de componentes



Programación en ES6 + Typescript

- ES6
 - let (variables con ámbito)
 - clases (class)
 - módulos (import y export)
 - funciones arrow
- TypeScript
 - tipos
 - anotaciones



TypeScript is a javascript superset

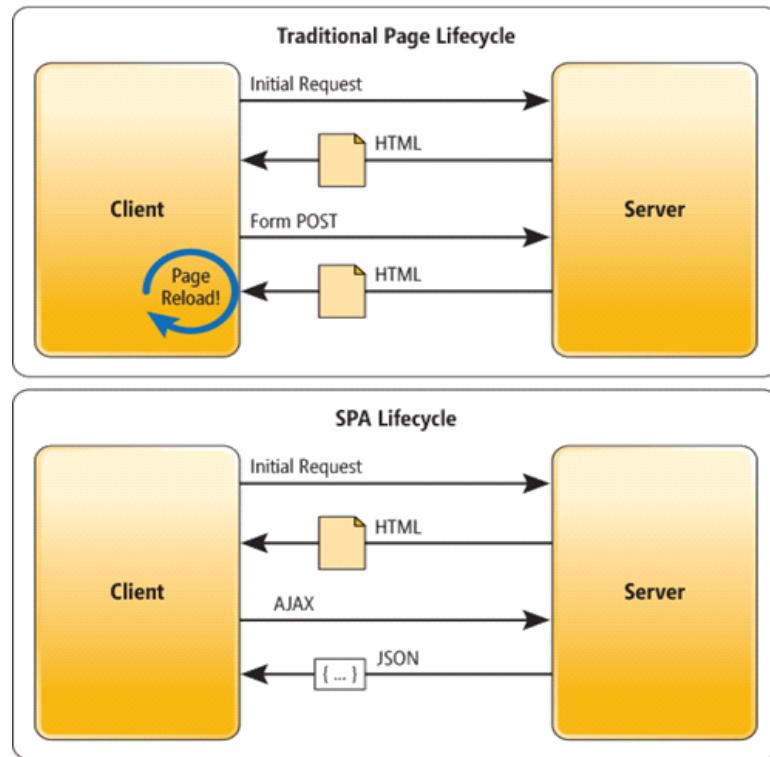
Transpilación

resultados en ES5 / ES6
(compatibilidad)

SPA: Single-Page Applications

sábado, 9 de septiembre de 2017 17:13

- carga completa en **una sola página**
- **Asincronicidad**
- limitada dependencia del **servidor**
- aproximación a las **aplicaciones de escritorio**





Elementos de AngularJS 1.x

Al margen de MVC, en términos prácticos, se distingue

- HTML
- JS



Scope

objeto normal de JavaScript que almacena los datos de un modelo
 recibe su nombre porque sustituye a *window* como ámbito global a nivel de las vistas



Elemento: HTML

(Vistas)

AngularJS
extiende el vocabulario
del código HTML

proporcionarnos la introducción
de lógica en la representación
de nuestra información



Directivas

- Son atributos HTML **ng-....**
- Suponen la posibilidad de modificar los elementos del DOM
- Pueden ser predefinidas o propias

Expresiones {{}}

- permite el uso de los elementos del modelo a nivel de las vistas
- También permite expresiones, e.g. operaciones matemáticas
- Lenguaje de plantillas : Similar a otros motores, como *Mustache*, *Smarty* o *Jade*



Elemento: JavaScript

(Controlador / Modelo)

JS

Ctrl

Factory

Service

Controlador (controller)

Es el código con la lógica que:

- define el modelo
- lo comunica con la vista mediante el *scope*

Modelos

Son la representación de los datos de la aplicación.

Se pueden definir de dos maneras

- en el código JS del **controlador**
- directamente en el HTML (la vista)

ng-init: tareas de inicialización de la aplicación, también permite definir el modelo directamente en el HTML

NADA RECOMENDABLE (anti-patrón)

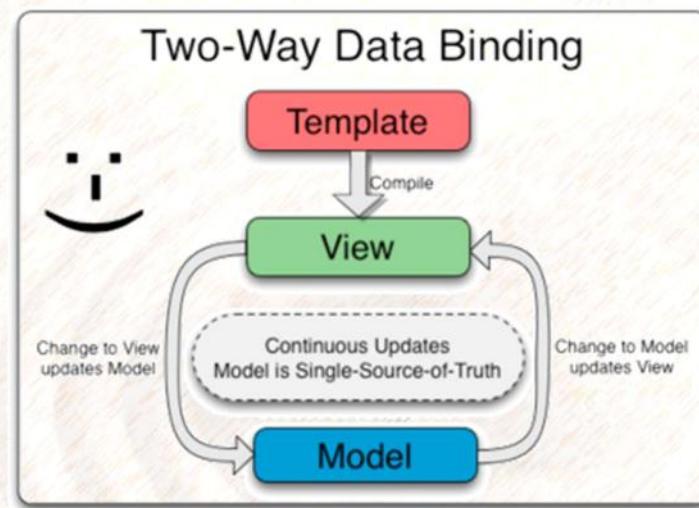


Elemento: Doble binding

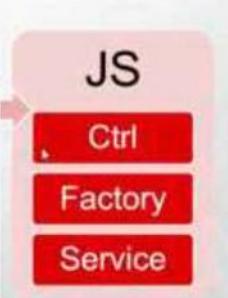
Doble binding (Two-way data binding)
entre el interface (vista) y el modelo

la vista se actualiza
automáticamente
cuando cambia el
modelo, y viceversa

Técnica frecuente en
Flex / ActionScript



Elementos: Modularización



Máxima modularización

- el código del controlador es el mínimo
- se distribuye en **módulos** (*module*)
- se transfieren funciones a los **servicios** (*service*)
- la creación de objetos se canaliza en **factorías** (*factory*)

Inyección de dependencias

- Modularidad → escalabilidad
- Acceso a servicios únicamente cuando son necesarios

Guía de estilo



John Papa

↓
Modularización
extrema

[GitHub /johnpapa /angular-styleguide](https://github.com/johnpapa/angular-styleguide)

Watch 1,096 Star 16,272 Fork 2,349

Code Issues 34 Pull requests 8 Pulse Graphs

Angular Style Guide: A starting point for Angular development teams to provide consistency through good practices. <http://johnpapa.net>

1,017 commits 3 branches 0 releases 131 contributors

Branch: master New pull request

New file Find file HTTPS https://github.com/johnpapa/ Download ZIP

johnpapa Merge pull request #634 from kel-sakal-bilyk/turkish

Latest commit 5859ad4 4 days ago

assets description changed from "Angular controller" to "Angular directive" 2 months ago

i18n Turkish translation 7 days ago

LICENSE Update license year to 2016 5 days ago

README.md Update license year to 2016 5 days ago

README.md

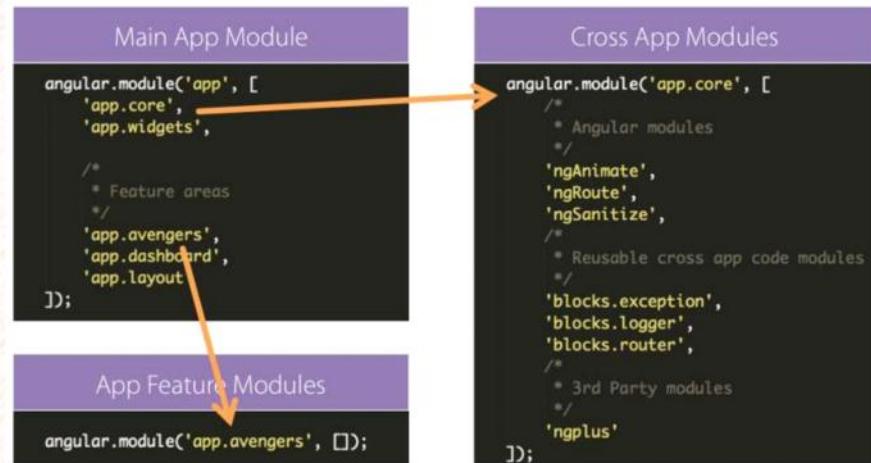
Angular Style Guide

Angular Team Endorsed

[https://github.com/johnpapa/
angular-styleguide](https://github.com/johnpapa/angular-styleguide)

Special thanks to Igor Minar, lead on the Angular team, for reviewing, contributing feedback, and entrusting me to shepherd this guide

Módulos según John Papa



Módulo APP [Style Y161]
Módulos de "características" [Style Y163]
Módulos reutilizables [Style Y164]

Nomenclatura
Scaffolding elegido

Otros elementos de estilo



John Papa

- Referencia a los módulos sin mapearlos nunca como variables
- *Closures* con el patrón de auto ejecución "Immediately Invoked Function Expression" (IIFE), con objeto de eliminar las variables del ámbito global.
- Empleo continuado de funciones con nombre en lugar de funciones anónimas
- Uso del formato "*controller as*" unido al empleo de una variable dentro del controlador, var `vm = this`, para evitar el uso de *this*
- Uso de `$inject` para definir los nombres de los elementos que se inyectan, de cara a evitar los problemas en el caos de la minimificación

Elementos de AngularJS 1.5



Componentes

- template + controller
- controllerAs
(por defecto \$ctrl)
- uso de clases
- constructores:
inyección de dependencias
- ciclo de vida del componente
(onInit)
- comunicación entre
componentes:
 - parent
 - binding

```
1 // ...
6 class SampleController {
7
8   /** ...
14   constructor($scope) {
15     'ngInject';
16     this.$scope = $scope;
17   }
18
19   /** ...
23   $onInit () {
24
25     this.name = "Sample"
26     this.value = parent.value
27
28   } // Fin del $onInit
29
30 } // Fin del controller SampleController
31
32
33 angular.module('moduleName')
34
35 // ...
36 */
41 .component("sample", {
42   require: {parent : '^appMain'},
43   templateUrl : 'components/sample.html',
44   // usa controller as por defecto
45   controller: SampleController,
46   //controllerAs: '$ctrl', valor por defecto
47   bindings: {}
48
49 }) //Fin del componente y del objeto que lo define
```



Guía de estilo 1.5

The screenshot shows a GitHub repository page for 'toddmotto / angular-styleguide'. At the top, there's a header with links for Personal, Open source, Business, Explore, Pricing, Blog, Support, This repository, Search, Sign in, and Sign up. Below the header is a profile picture of a man with short brown hair and a black t-shirt, smiling. The repository name 'toddmotto / angular-styleguide' is displayed above a list of statistics: 4 Issues, 0 Pull requests, 0 Projects, 181 commits, 3 branches, 0 releases, and 40 contributors. A button for 'New pull request' is visible. The commit history shows three recent commits:

Commit	Description	Date
laskovlymishka committed with toddmotto feat(ts): Initial typescript version of guide. (#127)	Adding Italian translation (#138)	Latest commit 7815d1e 8 days ago
i18n	feat(ts): Initial typescript version of guide. (#127)	11 days ago
typescript	Update README.md (#137)	8 days ago
README.md		21 days ago

The README.md file content is shown in a code block:

```
https://github.com/toddmotto/angular-styleguide
```

Angular 1.x styleguide (ES2015)

Architecture, file structure, components, one-way dataflow and best practices

Want an example structure as reference? Check out my component-based architecture 1.5 app.

Entorno de trabajo

sábado, 9 de septiembre de 2017 13:33

Navegadores.
Editores de código
Gestión de versiones. GIT
NodeJS y npm. Empaquetado

Entorno de trabajo

- Navegadores** → versiones "genéricas"
herramientas de desarrollo
versiones para desarrollo
- Git** → gestión del repositorio de software
- Visual Studio Code** → *plugins* adecuados a cada lenguaje / framework y otros de escritura ágil de código como *Emmet*
- Node.js:** → entorno JS en el lado servidor
permite crear servidores Web a medida
- npm** → gestor de **paquetes y dependencias**, utilizado por otros muchos elementos
- Web pack** → gestión de tareas, como la compilación de CSS, la transpilación de *TypeScript*, el *live reload* y el empaquetado de la aplicación final

Navegadores

sábado, 9 de septiembre de 2017 18:20

Herramienta de desarrollador en las últimas versiones de Chrome, en este caso la 60.0.3



Llévate Chrome a todas partes

Guarda página como... Ctrl+S
Añadir al escritorio...
Borrar datos de navegación... Ctrl+Mayús+Supr
Extensión...
Administrador de tareas Mayús+Esc
Herramientas para desarrolladores Ctrl+Mayús+I

Editor Cortar Copiar Pegar

Configuración Ayuda Ctrl+Mayús+Q

Salir

HTML (text/html; charset=UTF-8) 1 item hidden by filters

```
<html><head>
```

Herramienta de desarrollador en las últimas versiones de Firefox, en este caso la 55.0.3



Tu Firefox está al día.
Ahora ponte en marcha.

Envía Firefox a tu teléfono y da rienda suelta a tu Internet.

Escribe tu email

Nueva ventana Nueva ventana privada Guardar página
Imprimir Historial Pantalla completa
Buscar Opciones Complementos
Desarrollador Pestañas sincronizadas
Conectarse a Sync

Reglas Personalizar

elemento {
body { responsive-bundle.09005ea9425a.css:
position: static;
transition: transform .25s ease-in-out;
}
body { responsive-bundle.09005ea9425a.css:
overflow-x: hidden;
}

```
<!DOCTYPE html>
```

Herramienta de desarrollador en Edge, el navegador incorporado desde Windows 10



Sugerencias de Microsoft Edge

Ventana nueva Ventana InPrivate nueva

Zoom 100%

Transmitir contenido en un dispositivo

Buscar en la página

Imprimir

Añadir esta página a Inicio

Herramientas de desarrollo F12

Abrir con Internet Explorer

Enviar comentarios

Extensiones

Noticias y sugerencias

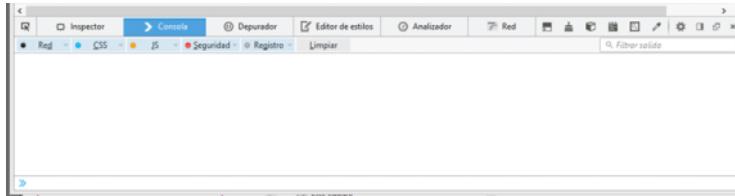
Configuración

```
<!DOCTYPE html>
```

Consola JavaScript

El ambiente en el que se ejecutan los scripts (navegador) proporciona un objeto console, que corresponde a la consola JS que podemos hacer visible en la parte inferior del navegador.

Los métodos console.log y console.dir son otra alternativa para presentar texto en pantalla desde un script. Algunos autores la prefieren a alert(), por considerarla menos intrusiva.



Versiones "especiales" para desarrolladores

chrome

DOWLOAD ▾ SET UP ▾ CHROMEBOOKS ▾ CHROMECAST ▾

Get on the bleeding edge of the web

Google Chrome Canary has the newest of the new Chrome features.
Be forewarned: it's designed for developers and early adopters, and can sometimes break down completely.

[Download Chrome Canary](#)

For Windows 10/8/1/7 64-bit
You can also download Chrome for Windows 32-bit, OSX, and Android.

<https://www.google.es/chrome/browser/canary.html>

Firefox Developer Edition

Herramientas modernas para la Web Abierta

Crea y depura experiencias web con poderosas herramientas de código abierto.

[Firefox Developer Edition](#)

Privacidad de Firefox

Firefox Developer Edition envía opiniones y sugerencias automáticamente a Mozilla. [Más información](#)

Modernizar
Crea interfaces de usuario rápidas, flexibles e interactivas con las herramientas integradas de React y Redux.

Personalizar
Haz tuyas nuestras herramientas de desarrollo gracias al código abierto y la personalización.

Optimizar
Crea sitios adaptables y compatibles que funcionan para todos, en todas partes.

<https://www.mozilla.org/es-ES/firefox/developer/>

Plugin en Chrome

Augury
offered by Augury.io

★★★★★ (50) | [Developer Tools](#) | 43,694 users

OVERVIEW REVIEWS SUPPORT RELATED G+1 22

Component Tree Router Tree
Properties Injector Graph
TodoList (View Source) (3a in Console)
Change Detection: Default
todoService: Object
Dependencies
TodoService
TodoInput (a-id = 0 13:0)
= TodoService
= FormatService
TodoList (a-id = 0 13:1)
= TodoService

Compatible with your device
Extends the Developer Tools, adding tools for debugging and profiling Angular 2.0 applications.
Augury is a Google Chrome Dev Tool extension for debugging Angular 2 applications. Treat this as a "developer preview":
- New in version 1.2.5
- Update kicthen sink example app to Angular 2.0.
- Resolve conflicts with Jira boards.
- Remove Angular core dependency from backend bundle.

Website Report Abuse

Additional Information
Version: 1.2.5
Updated: November 18, 2016
Size: 858KB
Languages: English

USERS OF THIS EXTENSION HAVE ALSO USED

<https://chrome.google.com/webstore/detail/augury/elgalmkoelokbchkhacckoklejnhcd>



Git es un SCV distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux. La licencia de Git es libre y hay distribuciones oficiales para los sistemas operativos:

- Mac OS X
- Windows
- Linux y
- Solaris.

La distribución de Git incluye herramientas de línea de comando y escritorio.

Además, hay disponibles herramientas proporcionadas por terceros que permiten una mayor integración con el escritorio o con entornos de desarrollo.



Comandos

git init - crear un repositorio local en un directorio.
git clone - crear un repositorio local haciendo una copia de otro (local o remoto).
git add - registra los ficheros del directorio de trabajo cuyos cambios se quieren.
git commit - confirma los cambios de los ficheros registrados.
git remote - permite interactuar con los remotos.

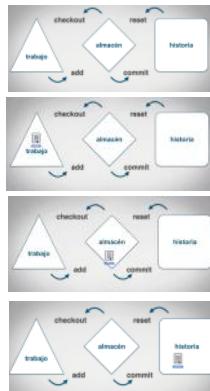
git branch - crear y borrar rama.
git checkout - permite cambiar de rama en el directorio de trabajo (Por defecto se trabaja en la rama denominada master).
git push - enviar los cambios a un repositorio remoto en la rama indicada.
git pull - obtener los cambios de un repositorio remoto y aplicarlos de una rama local.

Este comando es esencialmente un encadenamiento de dos comandos:
git fetch - obtiene los cambios de una rama remota.
git merge - fusiona si es posible estos cambios con una rama local.

Estados del archivo

Modificado Preparado Confirmado

Ciclo de operaciones



El repositorio creado tiene tres partes principales: el directorio de trabajo, el directorio de almacenamiento y la carpeta .git que ha sido creada por el comando git init. El directorio de trabajo es el área de preparación, que actúa como una zona intermedia, y el historial de cambios.

.Git permite el uso de ramas (branches).

El comando git pull es un ejemplo de la orientación a caja de herramientas que caracteriza el diseño de Git.

git pull = git fetch + git merge



Resultado



Comprobación



Apéndice 2 páginas 52

Git en la Nube. GitHub

domingo, 10 de septiembre de 2017 12:28

- **GitHub** →
- GitLab
- Bitbucket

The screenshot shows the GitHub sign-up interface. At the top, there's a navigation bar with links for Features, Business, Explore, Marketplace, and Pricing. On the right side of the header, there are buttons for 'Search GitHub', 'Sign in', and 'Sign up'. The main content area features a large heading 'Built for developers' followed by a descriptive paragraph about GitHub's mission. To the right, there's a form for creating a new account, including fields for 'Username', 'Email', 'Password', and a 'Sign up for GitHub' button. Below the button, a small text box contains the terms of service and privacy policy information.

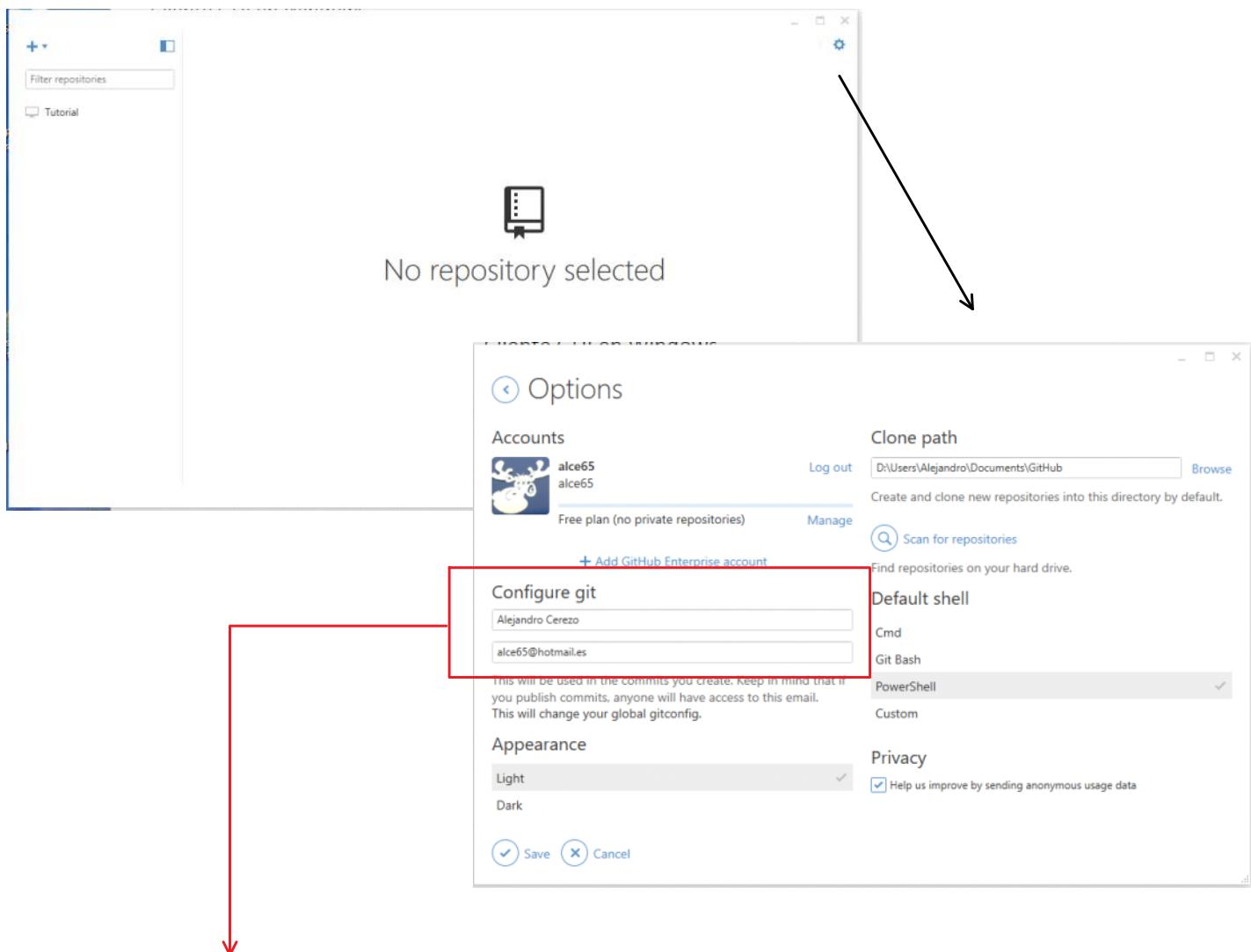
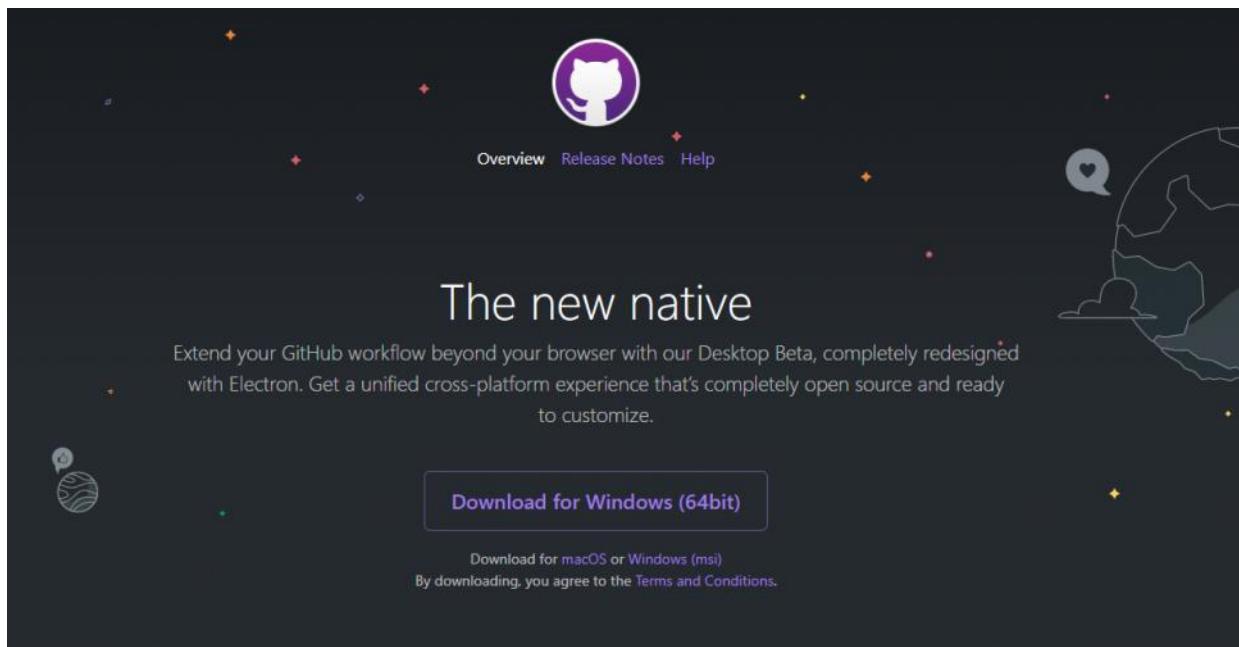
<https://github.com/alce65>

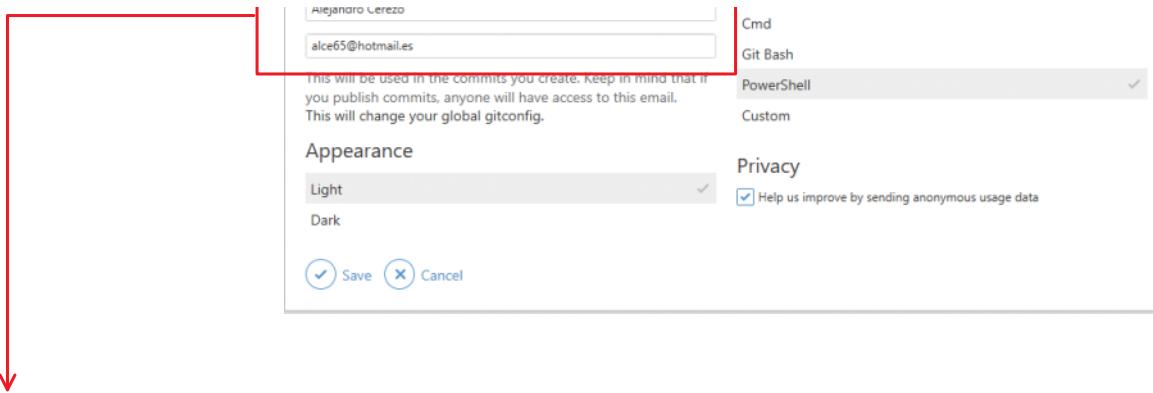
The screenshot shows the GitHub profile page for the user 'alce65'. The top navigation bar is identical to the sign-up page. The profile page includes a large profile picture of a reindeer, the user's name 'alce65', and a link to their GitHub profile. Below the profile picture, there are sections for 'Overview', 'Repositories 18', 'Stars 2', 'Followers 2', and 'Following 6'. A search bar at the top allows users to search for repositories. The main content area displays three repository cards: 'Curso_Angular2' (last updated 25 minutes ago), 'angularjs_2017' (last updated on 11 Aug), and 'CursoPelayo' (last updated on 16 Jun). Each repository card includes a thumbnail, the repository name, a brief description, and the last update date.

GUI para GitHub

domingo, 10 de septiembre de 2017 12:52

<https://desktop.github.com/>





Corresponde a los comandos de configuración de *git*

```
$ git config --global user.name "Pepe Perez"  
$ git config --global user.email pperez@example.com
```

Repositorio en GitHub

domingo, 10 de septiembre de 2017 12:25

Nuevo repositorio en GitHub

The screenshot shows the GitHub interface for creating a new repository. At the top, it says "Create a new repository" and "A repository contains all the files for your project, including the revision history." Below this, there are fields for "Owner" (set to "alce65") and "Repository name" (set to "Curso_Angular2"). A note says "Great repository names are short and memorable. Need inspiration? How about probable-spork." There is a "Description (optional)" field containing "Curso de Angular2 en Icono Training Consulting". Below these, there are two radio button options: "Public" (selected) and "Private". A note under "Public" says "Anyone can see this repository. You choose who can commit." A note under "Private" says "You choose who can see and commit to this repository." There is also a checked checkbox for "Initialize this repository with a README", with a note below it saying "This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository." At the bottom, there are buttons for "Add .gitignore: Node" and "Add a license: MIT License", followed by a "Create repository" button. A large red arrow points downwards from this section to the repository details page below.

This repository Search Pull requests Issues Marketplace Explore

alce65 / Curso_Angular2

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights Edit

Curso de Angular2 en Icono Training Consulting Add topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download ▾

alce65 initial commit

.gitignore Initial commit just now

LICENSE Initial commit just now

README.md Initial commit just now

README.md

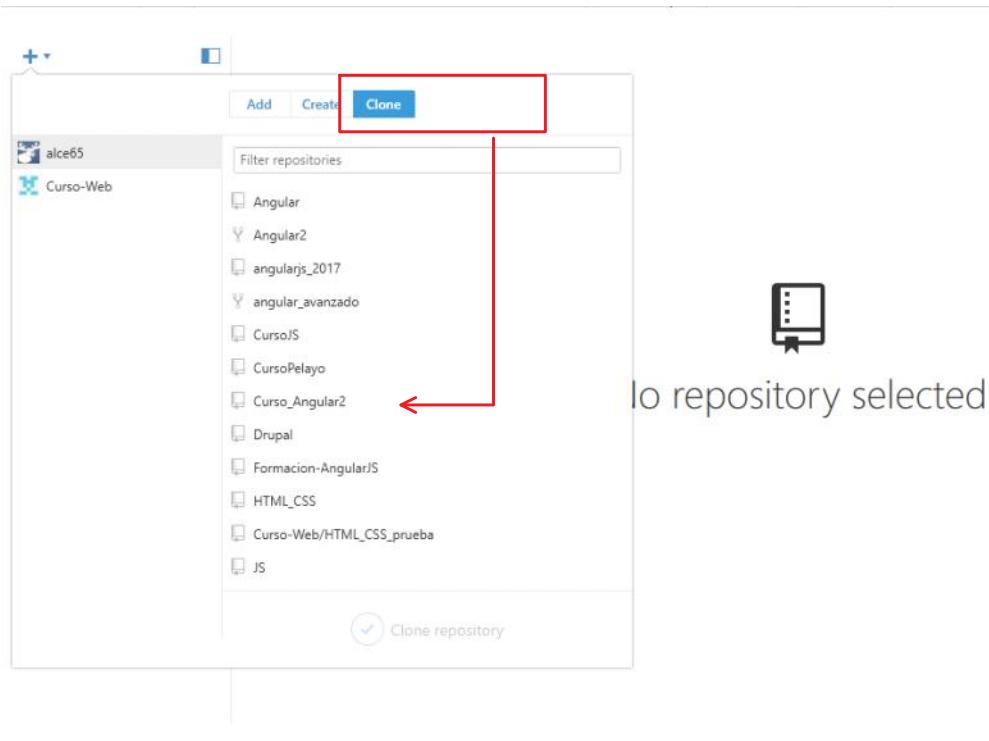
Curso_Angular2

Curso de Angular2 en Icono Training Consulting

Clonación local del repositorio

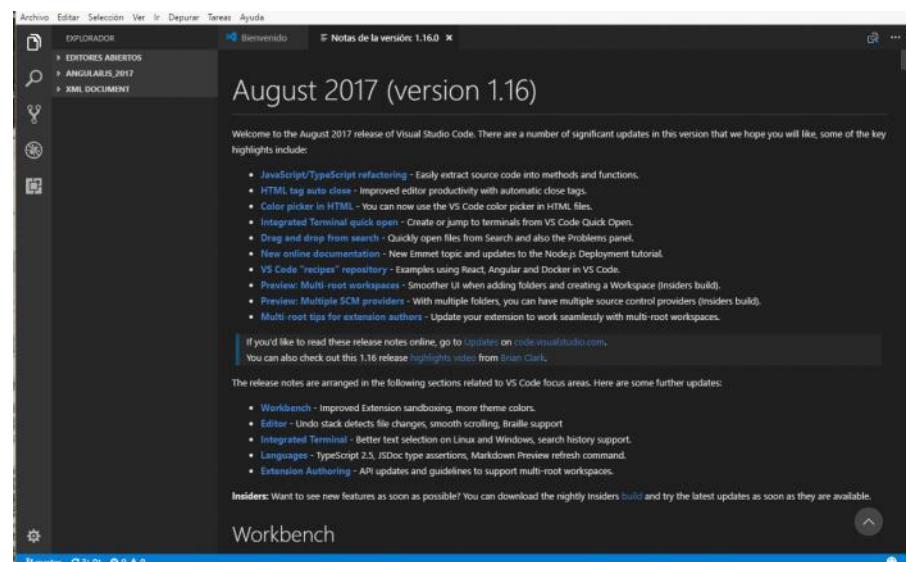
Clonar un repositorio

domingo, 10 de septiembre de 2017 17:22

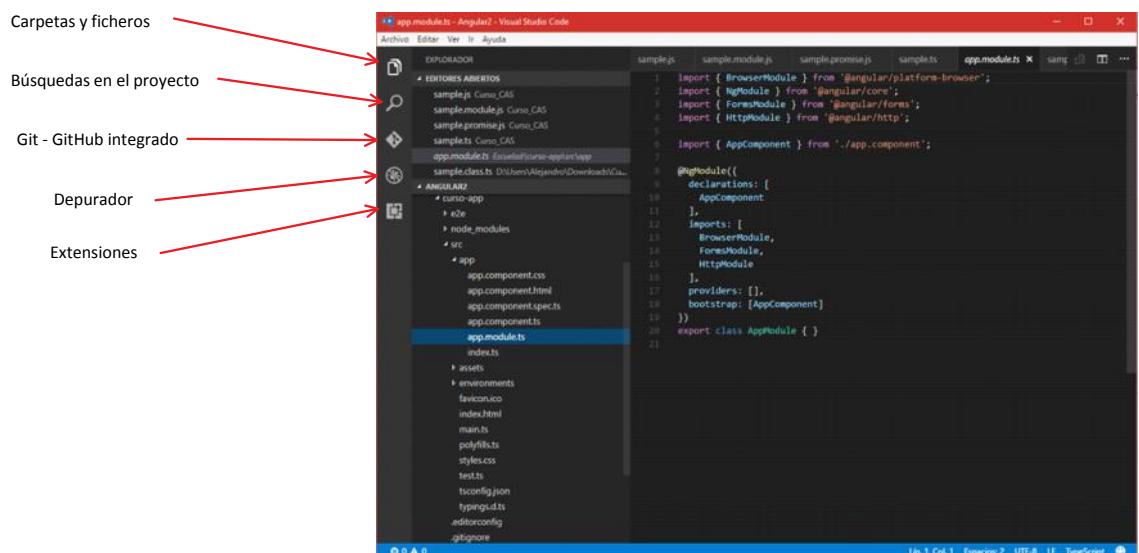


Editores de código

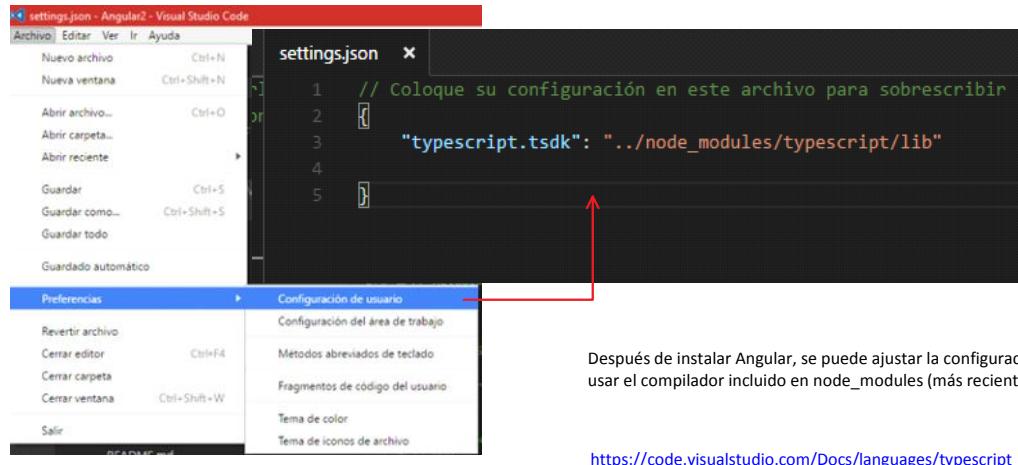
sábado, 9 de septiembre de 2017 18:21



Visual Studio Code



Configuración VSC



Después de instalar Angular, se puede ajustar la configuración para usar el compilador incluido en node_modules (más reciente)

<https://code.visualstudio.com/Docs/languages/typescript>

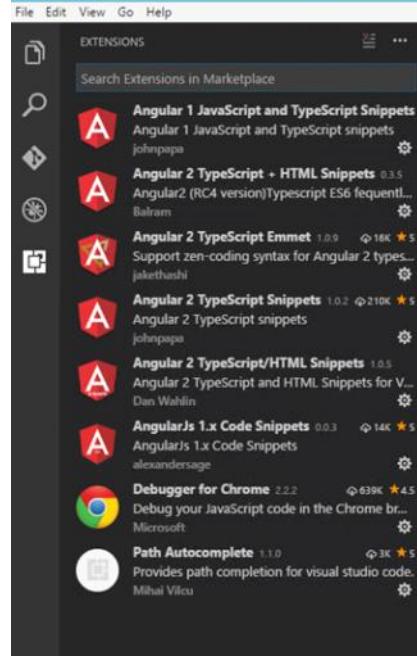
Extensiones de VSC

Angular 2 Typescript
Angular 2 Typescript Emmet
Angular 2 Typescript Snippets (John Papa)

TSLint
TypeScript HTML Snippets
TypeScript Emmet
TypeScript Snippets

Debugger para Chrome
Path Intellisense
(Path Autocomplete)

Auto Import
AutoRenameTag



Node y npm

sábado, 9 de septiembre de 2017 18:21

JS en el servidor (SSJS): Node.js

<https://nodejs.org/>



Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

Important security releases, please update now!

Descargar para Windows (x64)

v6.11.3 LTS

Recomendado para la mayoría

v8.4.0 Actual

Últimas características

Otras Descargas | Cambios | Documentación del API

Otras Descargas | Cambios | Documentación del API

Node.js® es un **entorno de ejecución para JavaScript** (una plataforma de software)

- Se emplea para construir aplicaciones de red escalables (especialmente servidores).
- Está construido con el motor de JavaScript V8 de Chrome
- Utiliza un modelo de operaciones que lo hace liviano y eficiente, gracias a
 - o **operaciones E/S sin bloqueo** y
 - o orientado a eventos, con un **bucle de eventos de una sola hebra**

Además, el **ecosistema de paquetes de Node.js, npm**, es el ecosistema más grande de librerías de código abierto en el mundo.

Su funcionalidad es especialmente adecuada en

- operaciones en tiempo real (e.g. chats)
- Bases de datos *NoSQL* / No relacionales



Node.js: ampliación de JS

Al *core* de JS no le acompañan las APIs habituales en cualquier lenguaje de programación

Node.js puede entenderse como la ampliación de JS para llegar a ser un lenguaje "completo", independiente de un entorno huésped

v8 (JavaScript)

- Una gramática que define la sintaxis del lenguaje
- Un intérprete/compilador que lo sabe interpretar y ejecutar
- Mecanismos para interactuar con el mundo exterior (llamadas al sistema)
- Librería estándar (consola, ficheros, red, etc...)
- Utilidades (intérprete interactivo, depurador, paquetes)

Node.js

También puede verse como un entorno huésped para JS en el servidor



Node.js: orígenes



Tiene su origen en un proyecto de **Ryan Dahl** y sus colaboradores en la empresa *Joyent*, que fue presentado en una conferencia en la *JSConf* de 2009.

<https://youtu.be/ztspvPYybIY>

Escrito en C/C++ y JS

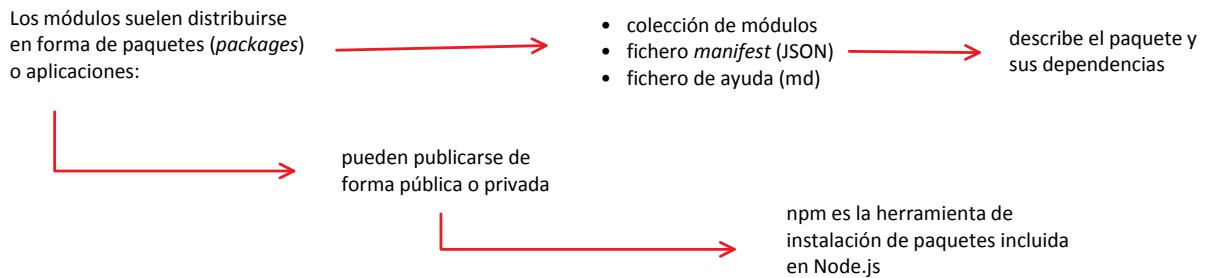
Objetivo del proyecto

Escribir aplicaciones muy eficientes en E/S con el lenguaje dinámico más rápido (v8) para soportar miles de conexiones simultáneas

Planteado sin complicaciones innecesarias

- Concurrencia sin paralelismo
- Lenguaje sencillo y muy extendido: JS
- API muy pequeña y muy consistente
- Apoyándose en Eventos y *Callbacks*

Paquetes: npm



Los proyectos Node.js creados en Visual Studio ya responden a la estructura de paquetes, para facilitar la creación de estos

▲ JS 04 Modulo Web Server

```
▷ └─ npm  
▷ └─ obj  
└─ app.js  
  └─ package.json  
  └─ README.md  
  └─ server.js
```

Instalación de Node.js & npm

sábado, 9 de septiembre de 2017 20:35

The screenshot shows the Node.js Setup Wizard. In the first window, the 'Welcome to the Node.js Setup Wizard' screen is displayed. In the second window, 'Installation Folder' is selected, with the path '(Program Files\nodejs)' chosen. A third window shows the 'Add to PATH' options: 'Node.js runtime', 'npm package manager', and 'Online documentation shortcuts'. An orange box highlights the 'node' and 'npm' entries under 'node'. Below these windows, a row of icons includes 'Node.js command prompt', 'Node.js documentation', 'Node.js website', 'Node.js', and 'Uninstall Node.js'. A red arrow points from the 'Node.js command prompt' icon to a terminal window titled 'Node.js command prompt'. The terminal displays the message: 'Your environment has been set up for using Node.js 8.1.4 (x64) and npm.' followed by the command prompt 'C:\Users\alce6>'.

A screenshot of the Windows Start Menu showing pinned icons for 'Node.js command prompt', 'Node.js documentation', 'Node.js website', 'Node.js', and 'Uninstall Node.js'.

A screenshot of the 'Símbolo del sistema' (System Command Prompt) window. It shows the command 'node -v' outputting 'v8.1.4' and the command 'npm -v' outputting '5.0.3'.

A screenshot of the 'Node.js command prompt - node' window. It shows a Read-Eval-Print-Loop (REPL) session where variables 'a' and 'b' are defined and multiplied, resulting in 36. A gray box at the bottom right indicates 'Salida ctrl-C o ctrl-D' (Output ctrl-C or ctrl-D).

Con el comando "node" entramos en la consola de Node, un entorno REPL (Read-Eval-Print-Loop) similar a la consola de JS en los navegadores

El mismo comando node <fichero.js>, permite la ejecución de un fichero

- Accesible desde cualquier CLI (cmd, PowerShell...)
- Incluido en el path
- Se puede comprobar consultando la versión de Node y de npm

Construcción de proyectos / empaquetado

herramientas para procesar los fuentes de la aplicación

- Reducción del tiempo de descarga
- Preprocesadores CSS
- Optimización del código, CSS, HTML
- Cumplimiento de estilos y
- Generación de JavaScript (transpilación)



<http://gruntjs.com/>



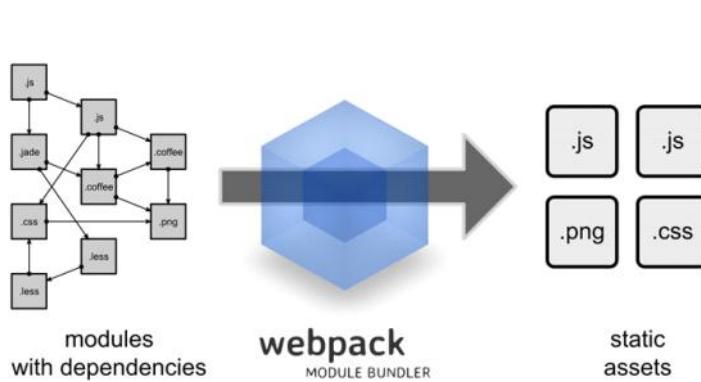
<http://gulpjs.com/>



<http://broccolijs.com/>



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



finalmente incluido en [Angular-cli](#)

Configuración de proxies

lunes, 7 de agosto de 2017 21:38

Configuración git

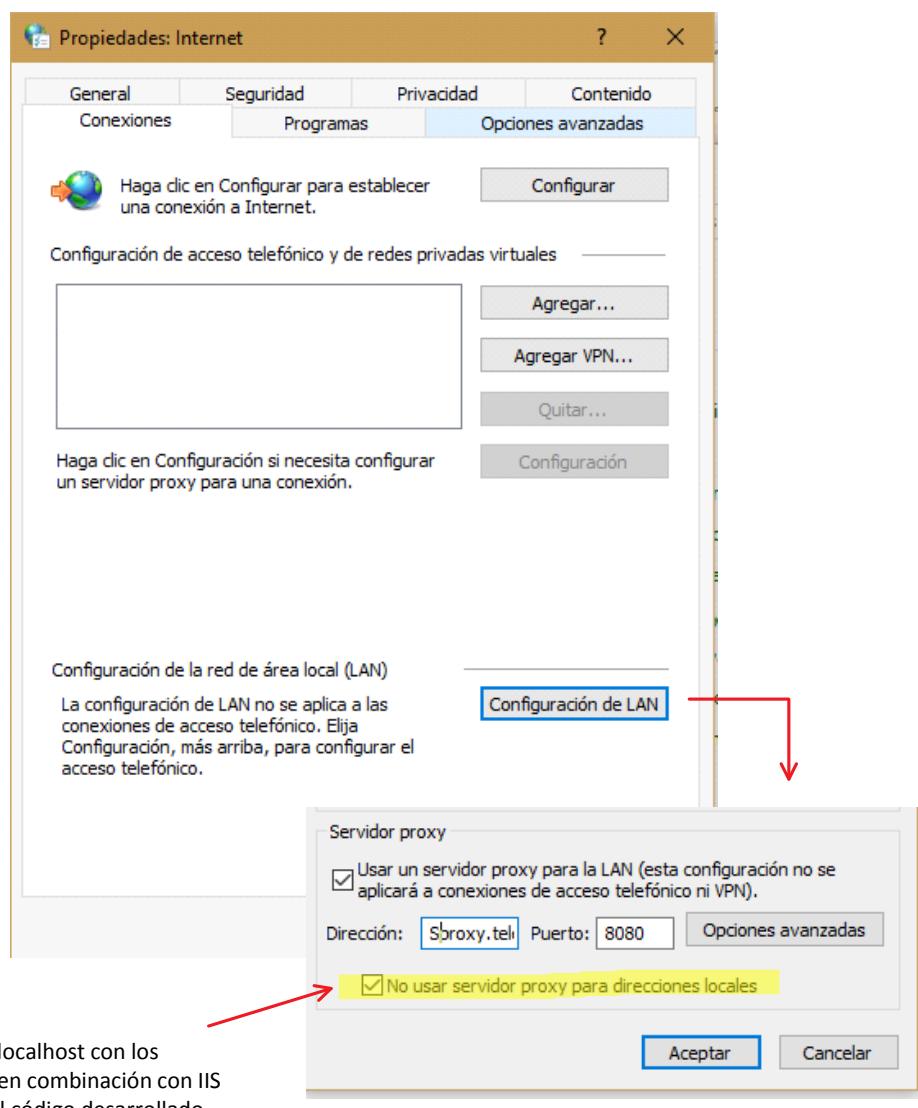
```
git config --global http.proxy http://user:passw@proxy.empresas.es:8080
```

Configuración npm

```
$>npm config set proxy http://user:passw@proxy.empresas.es:8080
```

```
$>npm config set https-proxy http://user:passw@proxy.empresas.es:8080
```

Propiedades de internet



Formas de creación de proyectos

No se suele crear un proyecto desde cero porque hay muchos ficheros y carpetas que son muy parecidos en todos los proyectos



Arquitectura o esqueleto (*scaffolding*)

Enfoques para conseguir el esqueleto inicial (*scaffolding*) de una web SPA (AngularJS / Angular)



- Generadores de plantillas ([Yeoman](#))
- Proyectos semilla ([seed](#)) disponibles en github
- En el caso de Angular
Herramienta oficial de gestión de proyectos : [angular-cli](#)

The screenshot shows the official Yeoman website. At the top right is a circular icon of a cartoon Yeoman character wearing a top hat and holding a sword. The main title "Yeoman" is centered in a large, bold, teal font. Below the title is a subtitle in Spanish: "Ecosistema de generadores de código y de proyectos de distintos lenguajes y plataforma, incluyendo uno específico para Angular.JS". A blue link "http://yeoman.io/" is provided. The main content area features a large illustration of the Yeoman character on the left, waving his hand. To the right, there's a scene with three people (two men and one woman) working on a large white rocket or engine component. The text "THE WEB'S SCAFFOLDING TOOL FOR MODERN WEBAPPS" is prominently displayed in the center. At the bottom, there's a call-to-action section with the text "Get started and then [find a generator](#) for your webapp. Generators are available for [Angular](#), [Backbone](#), [React](#), [Polymer](#) and over [1500+ other projects](#).", a command line interface (CLI) command "npm install -g yo", and a note "One-line install using [npm](#):".



Yeoman: instalación (1)

Dependencias previas

- Git
 - Node.js
 - Ruby
 - Compass
- 
- Pre-procesador
SaSS

Ejecutamos

npm -g install yo grunt-cli bower

A continuación se instalan los generadores requeridos, de los que existen para *Yeoman*; en este caso los de *Angular* y *Karma*

Generadores de código Angular 2 no oficiales basados en Yeoman

- <https://www.npmjs.com/package/generator-modern-web-dev>
- <https://www.npmjs.com/package/generator-angular2>
- <https://www.npmjs.com/package/generator-gulp-angular2>
- <https://github.com/joshuacaron/generator-angular2-gulp-webpack>
- <https://www.npmjs.com/package/slush-angular2>



Yeoman: instalación (2)

Creamos la carpeta del proyecto y en ella ejecutamos

yo angular <nombre_proyecto>

.../03c_Proyecto_Yo>yo angular 03c_Proyecto_Yo

El interface permite
seleccionar
fácilmente las
opciones presentadas

En un momento, la
instalación parece
quedarse detenida ;
hay que pulsar enter
aunque no se indica

```
Welcome to Yeoman,  
ladies and gentlemen!  
But of the box I include Bootstrap and some AngularJS recommended modules.  
Would you like to use Gulp (experimental) instead of Grunt? No  
Would you like to use Sass (with Compass)? No  
Would you like to include Bootstrap? Yes  
Which modules would you like to include? (Press <space> to select)  
(*) angular-animate.js  
( ) angular-aria.js  
(*) angular-cookies.js  
(*) angular-resource.js  
( ) angular-messages.js  
(*) angular-route.js  
(*) angular-sanitize.js  
(*) angular-touch.js
```



Yeoman: instalación (3)

```
Done, without errors.

Execution Time (2015-11-28 19:33:40 UTC)
loading tasks      534ms [██████████] 50%
loading grunt-wiredep 14ms [██] 15%
wiredep:app       472ms [██████████] 44%
wiredep:test       39ms [██] 4%
Total 1.1s
```

Una vez concluido ejecutamos **grunt**, para que realice todas las tareas definidas en Gruntfile.js

Finalmente **grunt serve** se encarga de levantar un servidor node.js en determinado puerto (e.g. 9000)

Proyecto con Yeoman

Nuevamente disponemos en el proyecto de app (donde trabajamos) y *dist* (mantenida por *grunt*) y actualizada con los cambios que hagamos gracias a *LiveReload*.

Vemos en el ejemplo el modo responsive tras haber modificado la vista `home.html`

03cProyectoYo Home About Contact

'Allo, 'Allo!

03cProyectoYo

Curso de Angular



Always a pleasure scaffolding your apps.

Splendid! ✓

HTML5 Boilerplate
HTML5 Boilerplate is a professional front-end template for building fast, robust, and adaptable web sites.

Angular
AngularJS is a toolset for building web apps or sites.

Karma
Spectacular Test Runner for Java

Angular
AngularJS is a toolset for building the framework most suited to your application development.

♥ from the Yeoman team

HTML5 Boilerplate
HTML5 Boilerplate is a professional front-end template for building fast, robust, and adaptable web apps or sites.

Angular

AngularJS is a toolset for building the framework most suited to your application development.

♥ from the Yeoman team



Yeoman: generadores

Yeoman dispone además de diversos generadores de código

<https://github.com/yeoman/generator-angular>

angular:controller
angular:directive
angular:filter
angular:route
angular:service
angular:provider
angular:factory
angular:value
angular:constant
angular:decorator
angular:view

Para ejecutarlos:

- detenemos el servidor web levantado por **grunt**
- ejecutamos yo y el generador que necesitamos

yo angular:view <nombre>

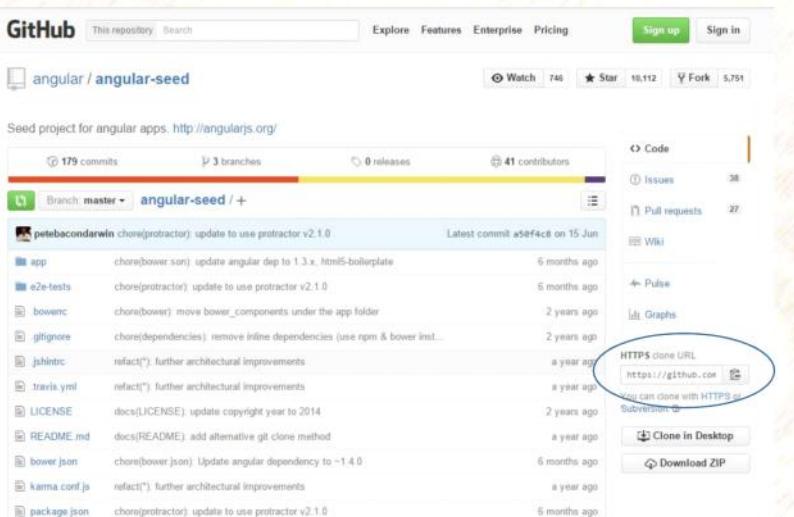
Crearía la correspondiente vista nueva en la carpeta adecuada

Angular seed 1.x



en Angular 1.x esqueleto de una aplicación desarrollada por el propio equipo de AngularJS, con el respaldo de Google, como punto de partida para proyectos

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





Angular seed 2

Proyectos semilla (seed) disponibles en github

- <http://mgechev.github.io/angular2-seed/>
- <https://github.com/ghpabs/angular2-seed-project>
- <https://github.com/cureon/angular2-sass-gulp-boilerplate>
- <https://angularclass.github.io/angular2-webpack-starter/>
- <https://github.com/LuxDie/angular2-seed-jade>
- <https://github.com/justindujardin/angular2-seed>

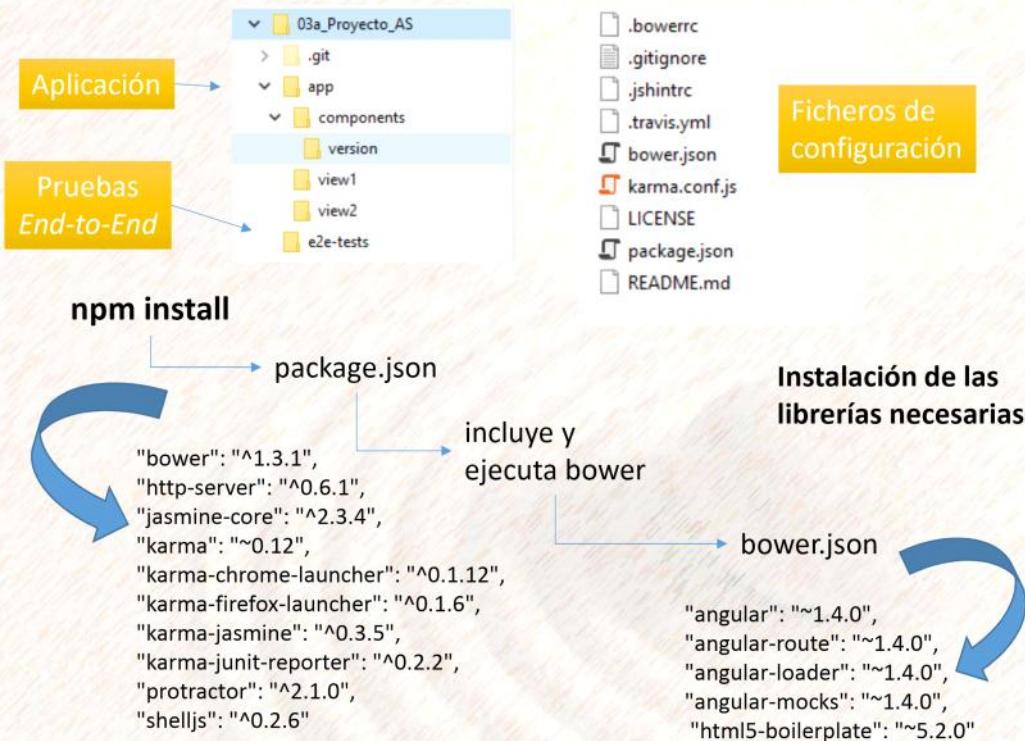
git clone origen destino

git clone https://github.com/angular/angular-seed.git 03a_Proyecto_AS

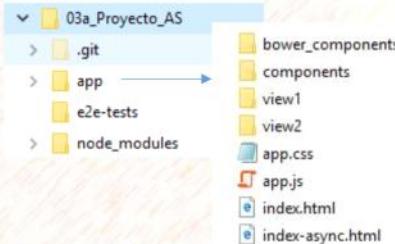
```
D:\Users\Alejandro\Mi nube\OneDrive\Desarrollo\Angular>git clone https://github.com/angular/angular-seed.git 03a_Proyecto_AS
Cloning into '03a_Proyecto_AS'...
remote: Counting objects: 2590, done.
Receiving objects: 100% (2590/2590), 12.21 MiB | 4.16 MiB/s, done.
Resolving deltas:  1% (14/1370)    0 (delta 0), pack-reused 2590
Resolving deltas: 100% (1370/1370), done.
Checking connectivity... done.
```

No incluye aún ninguna librería

Angular seed: Instalación



Proyecto con Angular seed



npm start

```
> http-server -a localhost -p 8000 -c-1
Starting up http-server, serving ./ on port: 8000
Hit CTRL-C to stop the server
```

En el navegador

<http://localhost:8000/app>

Podemos modificar el contenido

Accedemos a
index.html vía *localhost*

- usando IIS
- usando el servidor Node.js incluido y ya configurado

[[view1](#) | [view2](#)]

This is the partial for view 1.

Angular [[view1](#) | [view2](#)]

Este es el "parcial" para la vista 2.

Angular-cli

sábado, 9 de septiembre de 2017 18:51

Angular command line interface

Herramienta oficial de gestión de proyectos

<https://cli.angular.io>

Ofrece comandos para todo el **ciclo de desarrollo**:

- Generación del proyecto inicial
- Generación de partes posteriormente
- Modo desarrollo con compilado automático de *TypeScript* y actualización del navegador
- *Testing*
- Construcción del proyecto para distribución (*build*)



Herramientas incluidas en Angular-cli



Herramientas de testing

Instalación

Desde una ventana de terminal "como administrador"

`npm install -g @angular/cli`

instala globalmente la herramienta angular cli

`npm update -g @angular/cli`

actualiza la instalación global de angular cli

Destino de la instalación

"<user>\AppData\Roaming\npm\node_modules"

Resultado de la instalación

```
C:\WINDOWS\system32\cmd.exe
D:\>ng version
angular-cli: 1.0.0-beta.19-3
node: 6.9.1
os: win32 x64
```

```
C:\Users\alce6>ng version
Angular CLI: 1.4.1
@angular/cli: 1.4.1
node: 8.4.0
os: win32 x64
C:\Users\alce6>
```

Generación de un proyecto

Desde la carpeta donde queremos que resida nuestro proyecto

`ng new my-app`

crea una carpeta en la que descarga hasta 250MB. configura y organiza el conjunto de herramientas de una forma prefijada

```
cd my-app
```

```
ng serve
```

desde el directorio del proyecto, recién creado se levanta un servidor *Node* que mostrará la aplicación en localhost:4200

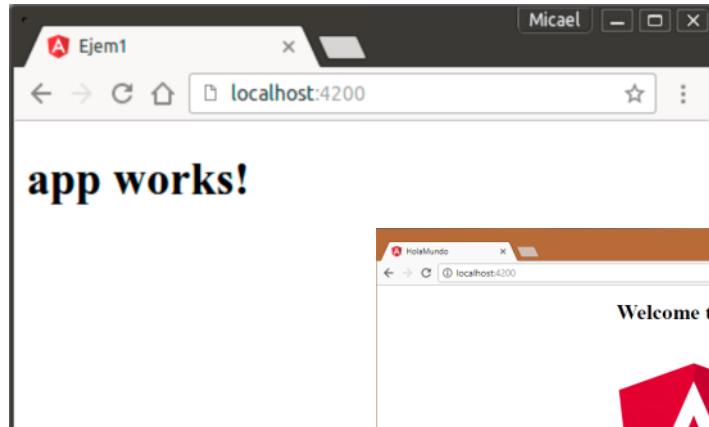
- *transpilará Typescript*
- recargara automáticamente al guardar un fichero fuente

Resultado: ejecución

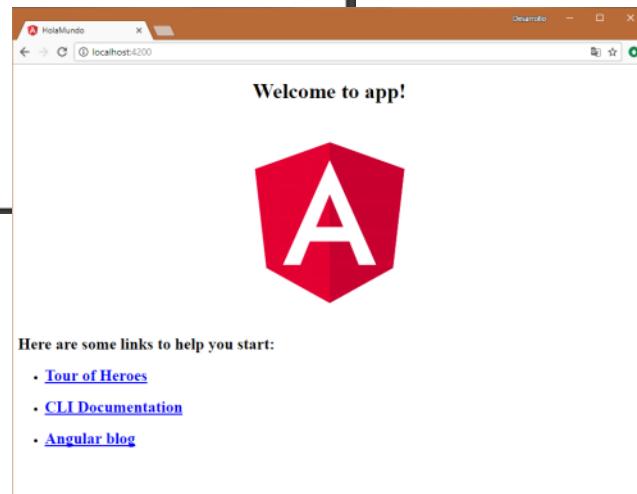
```
D:\Desarrollo\Angular2\Curso_CAS\curso-app>ng serve
** NG Live Development Server is running on http://localhost:4200. **
4387ms building modules
47ms sealing
0ms optimizing
0ms basic module optimization
18ms module optimization
4ms advanced module optimization
18ms basic chunk optimization
15ms chunk optimization
10ms forced chunk optimization
16ms module and chunk tree optimization
266ms module reviving
15ms module order optimization
16ms module id optimization
16ms chunk reviving
0ms chunk order optimization
31ms chunk id optimization
109ms hashing
16ms module assets processing
250ms chunk assets processing
15ms additional chunk assets processing
0ms recording
0ms additional asset processing
4069ms chunk asset optimization
2923ms asset optimization
78ms emitting
Hash: 541eb735658c9449ad60
Version: webpack 2.1.0-beta.25
Time: 5202ms
    Asset      Size  Chunks   Chunk Names
main.bundle.js  2.71 MB  0, 2  [emitted]  main
styles.bundle.js 10.2 kB  1, 2  [emitted]  styles
inline.js       5.53 kB  2  [emitted]  inline
main.map        2.82 MB  0, 2  [emitted]  main
styles.map       14.2 kB  1, 2  [emitted]  styles
inline.map       5.59 kB  2  [emitted]  inline
index.html     475 bytes  [emitted]
Child html-webpack-plugin for "index.html":
    Asset      Size  Chunks   Chunk Names
index.html     2.81 kB  0
webpack: bundle is now VALID.
```

Ventana de la consola en la que webpack

- levanta un servidor Web basado en *Node* en la máquina local y el puerto 4200
- con la aplicación empaquetada de forma temporal en modo adecuado para el desarrollo,
- capaz de actualizarse automáticamente ante los cambios en los ficheros fuente



Primera aplicación en Angular2 generada por angular-cli en sus versiones beta



Primera aplicación en Angular2 generada por angular-cli en sus versiones definitivas

Hola Mundo

sábado, 9 de septiembre de 2017 23:57

Instalación de VSC y sus extensiones

Instalación de *git*
configuración del proxy

```
$>git config --global http.proxy http://user:passw@proxy.empresa.es:8080
```

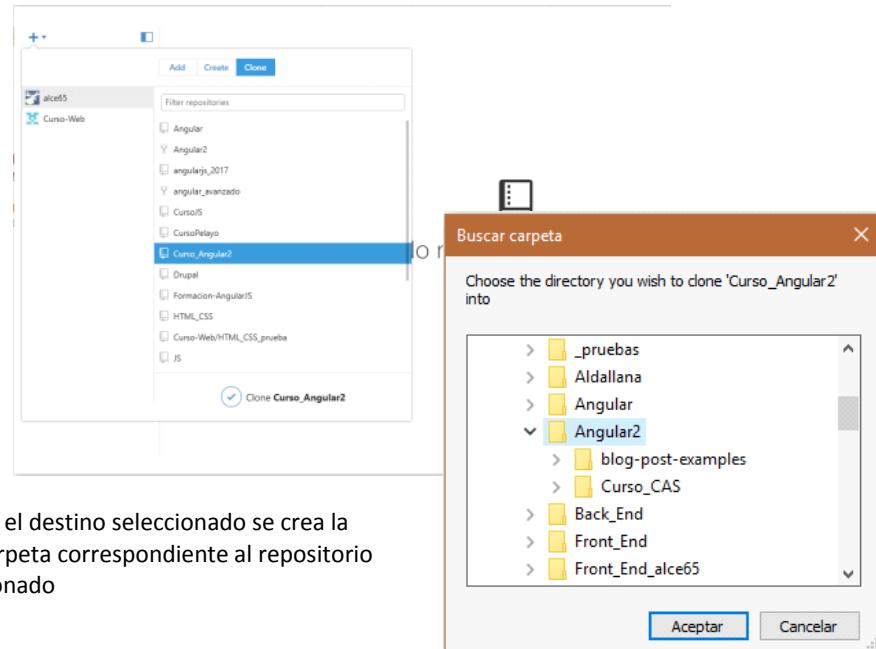
Instalación de *node/npm*
configuración del proxy

```
$>npm config set proxy http://user:passw@proxy.empresa.es:8080  
$>npm config set https-proxy http://user:passw@proxy.empresa.es:8080
```

Instalación de Angular-cli:

```
$>npm install -g @angular/cli
```

Creación de la carpeta para el curso,
como repositorio *git* vinculado a GitHub



Creación del proyecto: ng new hola-mundo

```
C:\Windows\system32\cmd.exe
D:\Desarrollo\Angular2\Curso_Angular2> ng new hola-mundo

C:\Windows\system32\cmd.exe
D:\Desarrollo\Angular2\Curso_Angular2> ng new hola-mundo
Unable to find "@angular/cli" in devDependencies.

Please take the following steps to avoid issues:
"npm install --save-dev @angular/cli@latest"

  create  hola-mundo/e2e/app.e2e-spec.ts (292 bytes)
  create  hola-mundo/e2e/app.po.ts (208 bytes)
  create  hola-mundo/e2e/tsconfig.e2e.json (235 bytes)
  create  hola-mundo/karma.conf.js (923 bytes)
  create  hola-mundo/package.json (1315 bytes)
  create  hola-mundo/protractor.conf.js (722 bytes)
  create  hola-mundo/README.md (1100 bytes)
  create  hola-mundo/tsconfig.json (363 bytes)
  create  hola-mundo/tslint.json (3040 bytes)
  create  hola-mundo/.angular-cli.json (1128 bytes)
  create  hola-mundo/.editorconfig (245 bytes)
  create  hola-mundo/.gitignore (516 bytes)
  create  hola-mundo/src/assets/.gitkeep (0 bytes)
  create  hola-mundo/src/environments/environment.prod.ts (51 bytes)
  create  hola-mundo/src/environments/environment.ts (387 bytes)
  create  hola-mundo/src/favicon.ico (5430 bytes)
  create  hola-mundo/src/index.html (296 bytes)
  create  hola-mundo/src/main.ts (370 bytes)
  create  hola-mundo/src/polyfills.ts (2480 bytes)
```

```
Administrator: C:\Windows\system32\cmd.exe
D:\Desarrollo\Angular2\Curso_Angular2> ng new hola-mundo
Unable to find "@angular/cli" in devDependencies.

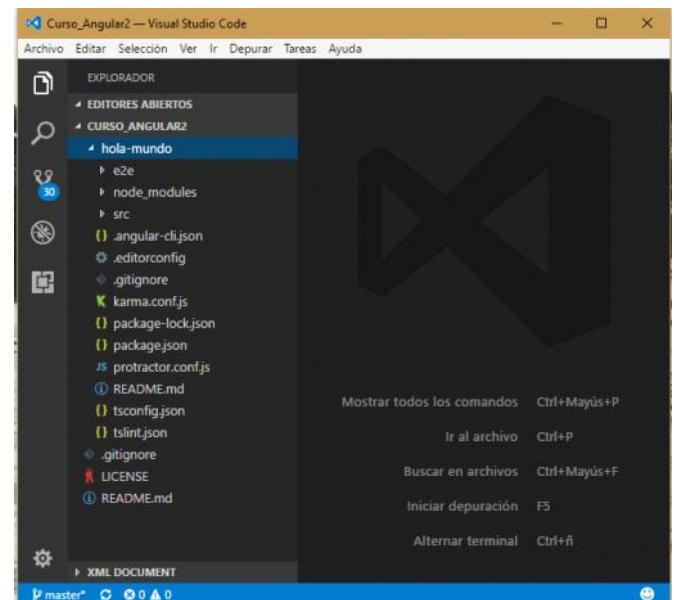
Please take the following steps to avoid issues:
"npm install --save-dev @angular/cli@latest"

  create  hola-mundo/e2e/app.e2e-spec.ts (292 bytes)
  create  hola-mundo/e2e/app.po.ts (208 bytes)
  create  hola-mundo/e2e/tsconfig.e2e.json (235 bytes)
  create  hola-mundo/karma.conf.js (923 bytes)
  create  hola-mundo/package.json (1315 bytes)
  create  hola-mundo/protractor.conf.js (722 bytes)
  create  hola-mundo/README.md (1100 bytes)
  create  hola-mundo/tsconfig.json (363 bytes)
  create  hola-mundo/tslint.json (3040 bytes)
  create  hola-mundo/.angular-cli.json (1128 bytes)
  create  hola-mundo/.editorconfig (245 bytes)
  create  hola-mundo/.gitignore (516 bytes)
  create  hola-mundo/src/assets/.gitkeep (0 bytes)
  create  hola-mundo/src/environments/environment.prod.ts (51 bytes)
  create  hola-mundo/src/environments/environment.ts (387 bytes)
  create  hola-mundo/src/favicon.ico (5430 bytes)
  create  hola-mundo/src/index.html (296 bytes)
  create  hola-mundo/src/main.ts (370 bytes)
  create  hola-mundo/src/polyfills.ts (2480 bytes)
  create  hola-mundo/src/styles.css (80 bytes)
  create  hola-mundo/src/test.ts (1085 bytes)
  create  hola-mundo/src/tsconfig.app.json (211 bytes)
  create  hola-mundo/src/tsconfig.spec.json (304 bytes)
  create  hola-mundo/src/typings.d.ts (104 bytes)
  create  hola-mundo/src/app/app.module.ts (314 bytes)
  create  hola-mundo/src/app/app.component.html (1075 bytes)
  create  hola-mundo/src/app/app.component.spec.ts (986 bytes)
  create  hola-mundo/src/app/app.component.ts (207 bytes)
  create  hola-mundo/src/app/app.component.css (0 bytes)

Installing packages for tooling via npm.
Installed packages for tooling via npm.
Directory is already under version control. Skipping initialization of git.
Project 'hola-mundo' successfully created.

D:\Desarrollo\Angular2\Curso_Angular2>
```

Accedemos al proyecto desde VSC



Resultado

domingo, 10 de septiembre de 2017 18:51

The screenshot shows the final result of the Angular 2 'Hello World' application. At the top, there is a Visual Studio Code interface with a code editor showing the file 'app.component.html'. The code contains the following HTML:

```
only if placeholder and can be replaced.-->
<h1>Welcome to app!
Tour of Heroes</a><br/>
<a href="https://github.com/angular/angular-cli/wiki">CLI Documentation</a><br/>
<a href="https://blog.angular.io/">Angular blog</a></p>
```

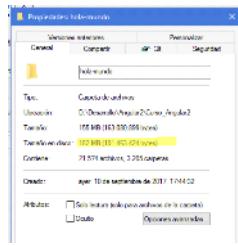
Below the code editor is a terminal window showing the command 'ng serve' being run in a PowerShell session. The output of the command is visible.

At the bottom, a browser window displays the application. The title bar says 'HolaMundo'. The page content includes the text 'Welcome to app!', the Angular logo, and a section titled 'Here are some links to help you start:' with three links: 'Tour of Heroes', 'CLI Documentation', and 'Angular blog'.

```
PS D:\Desarrollo\Angular2\Curso_Angular2> cd .\hola-mundo\>
PS D:\Desarrollo\Angular2\Curso_Angular2\hola-mundo> ng serve
```

```
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2017-09-10T16:55:54.342Z
Hash: d48ae1a31f62b6eb16bc
Time: 8899ms
chunk {inline} inline.bundle.js, inline.bundle.js.map (inline) 5.83 kB [entry] [rendered]
chunk {main} main.bundle.js, main.bundle.js.map (main) 8.64 kB [vendor] [initial] [rendered]
chunk {polyfills} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 209 kB [inline] [initial] [rendered]
chunk {styles} styles.bundle.js, styles.bundle.js.map (styles) 11.3 kB [inline] [initial] [rendered]
chunk {vendor} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.28 MB [initial] [rendered]
webpack: Compiled successfully.
```

Resultado: estructura



ficheros de infraestructura

- Infraestructura para Typescript
- Infraestructura de pruebas
- Infraestructura de despliegue

- .editconfig
- .gitignore: git

- package.json: librerías (dependencias)
- package_lock.json
- angular-cli.json: angular-cli
- angular-cli-build.js: angular-cli
- tsconfig.json: typescript
- tsconfig.json: tipos
- karma.conf.js
- protractor.conf.js

Ficheros: de configuración del editor y de git

package.json

```
{
  "name": "curso-app",
  "version": "0.0.0",
  "license": "MIT",
  "angular-cli": {},
  "scripts": {
    "start": "ng serve",
    "lint": "tslint \"src/**/*.ts\"",
    "test": "ng test",
    "pree2e": "webdriver-manager update",
    "e2e": "protractor"
  },
  "private": true,
  "dependencies": {
    "@angular/common": "~2.1.0",
    "@angular/compiler": "~2.1.0",
    "@angular/core": "~2.1.0",
    "@angular/forms": "~2.1.0",
    "@angular/http": "~2.1.0",
    "@angular/platform-browser": "~2.1.0",
    "@angular/platform-browser-dynamic": "~2.1.0",
    "@angular/router": "~2.1.0",
    "core-js": "~2.4.1",
    "rxjs": "5.0.0-beta.12",
    "ts-helpers": "~1.1.1",
    "zone.js": "~0.6.23"
  },
  "devDependencies": {
    "@types/jasmine": "~2.2.30",
    "@types/node": "~6.0.42",
    "angular-cli": "1.0.0-beta.19-3",
    "codeylyer": "1.0.0-beta.1",
    "jasmine-core": "2.4.1",
    "jasmine-reporter": "2.5.0",
    "karma": "1.2.0",
    "karma-chrome-launcher": "~2.0.0",
    "karma-cli": "~1.0.1",
    "karma-jasmine": "~1.0.2",
    "karma-jasmine-html-reporter": "0.2.1",
    "protractor": "4.0.9",
    "ts-node": "1.2.1",
    "tslint": "3.13.0",
    "typescript": "~2.0.3",
    "webdriver-manager": "10.2.5"
  }
}
```

Scripts

Dependencias

Dependencias de desarrollo

- **E2e:** Testing end to end
- **dist:** Recursos que hay que publicar en el servidor web
- **node_modules:** Librerías y herramientas descargadas
- **src:** Fuentes de la aplicación

Angular 2.0

- **config:** configuración de environments y tests
- **E2e:** Testing end to end
- **dist:** Recursos que hay que publicar en el servidor web
- **node_modules:** Librerías y herramientas descargadas
- **public:**
- **src:** Fuentes de la aplicación
- **tmp:** temporal
- **typings:** tipos predefinidos para información al editor de código

config

- config
- dist
- e2e
- node_modules
- public
- src
- tmp
- typings
- .clang-format
- .editorconfig
- .gitignore
- angular-cli.json
- angular-cli-build.js
- package.json
- tslint.json
- typings.json

- karma.conf.js: tests con karma
- protractor.conf.js: tests con protractor

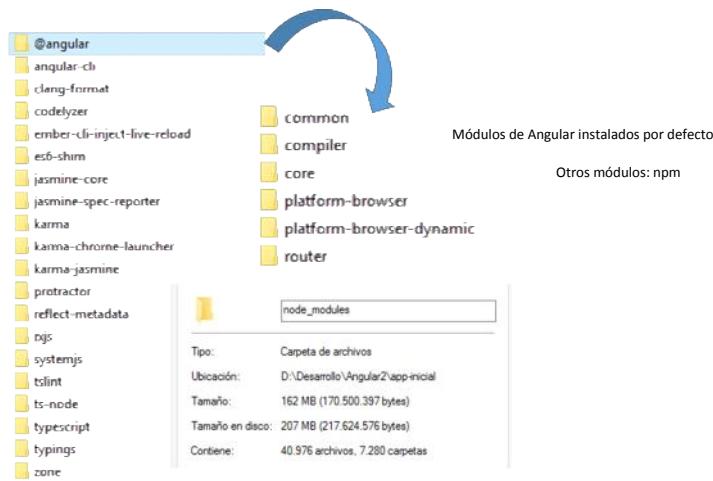
- package.json: librerías (dependencias)
- .editconfig
- .gitignore: git

- package.json: librerías (dependencias)
- angular-cli.json: angular-cli
- angular-cli-build.js: angular-cli
- tsconfig.json: typescript
- tslint.json: tipos
- typings.json: tipos

Node_Modules

domingo, 10 de septiembre de 2017 11:45

Resultado: node_modules



- No se incluye en la copia de los proyectos distribuida en repositorios git / GitHub

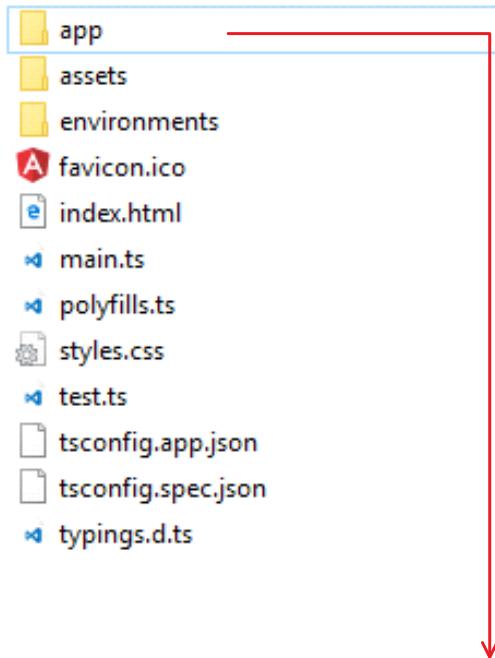
- Puede volver a generarse en cualquier momento, de acuerdo con lo indicado en package.json: npm install

- Puede reubicarse en niveles superiores para compartirlo en diversos proyectos

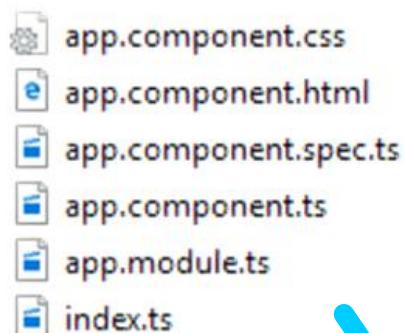
- npm busca la carpeta node_modules en la carpeta actual o en los niveles superiores
- Puede reubicarse en niveles superiores para compartirlo en diversos proyectos
- Puede ser necesario ajustar la configuración del editor de código, para indicarle donde se ubica el compilador de typescript

Link Shell Extension 14,6 MB 15/07/2017

ng serve --preserve-symlinks



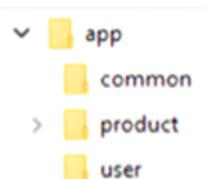
- **index.html:** Página principal.
Se editarará para incluir CSS globales en la web
- favicon.ico: Icono de la aplicación
- **main.ts:** Fichero principal de la aplicación.
No es necesario modificarlo
- **tsconfig.app.json**
tsconfig.spec.json: Configuración del compilador TS
- style.css
- polyfills.ts
- test.ts
- typings.d.ts



carpeta que contiene los ficheros fuente principales de la aplicación.

Distribución por características

se agrupan los elementos correspondientes a sus distintas características, e.g. las opciones del menú principal



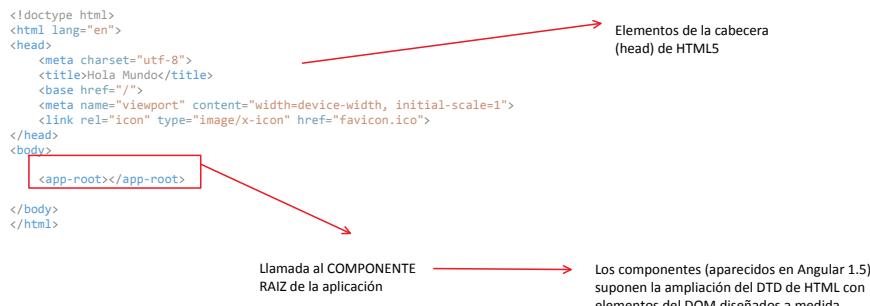
Ficheros principales

domingo, 10 de septiembre de 2017 18:24

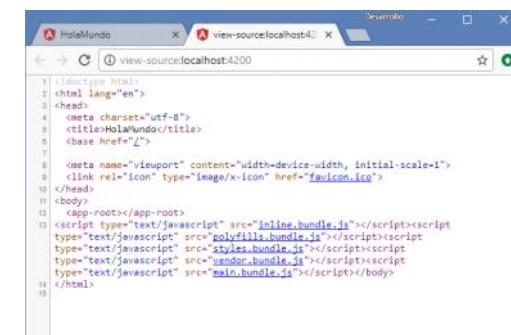
Automáticamente traducido a JS e injectado en index.html por parte de Webpack



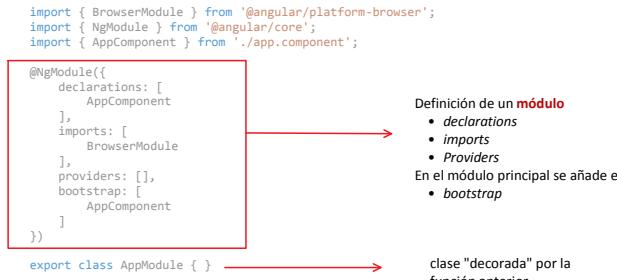
.\src\index.html



Se echan en falta llamadas a ficheros externos (JS, CSS)
La función de Webpack es "empaquetar" toda esa información en bundles e injectar las correspondientes llamadas a ellos



.\src\app\app.module.ts



clase "decorada" por la función anterior

.\src\app\app.component.ts



Personalizar y publicar

domingo, 10 de septiembre de 2017 19:00

Teniendo activo `ng serve`, modificamos

`.\src\app\app.component.html`

```
<div style="text-align:center">
  <h1>
    Bienvenidos al curso {{curso}}!
  </h1>
  
</div>
```

Creamos la variable en el componente

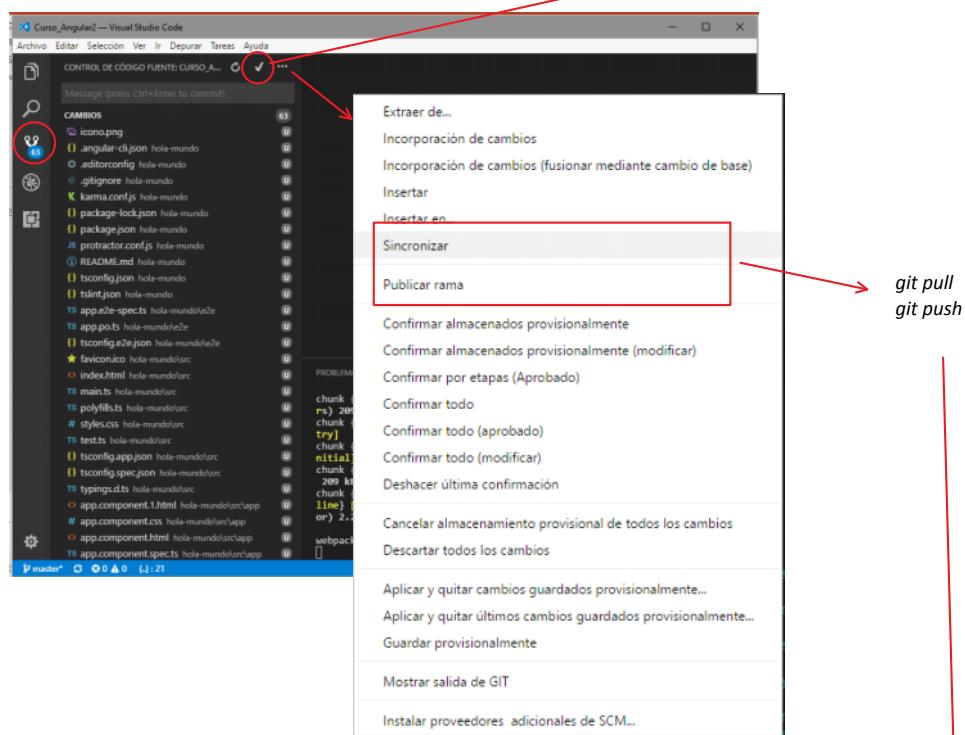
Incorporamos el fichero

Automáticamente se recompila y se muestran los resultados



Desde el propio VSC podemos incorporar los proyectos al repositorio de [GitHub](#)

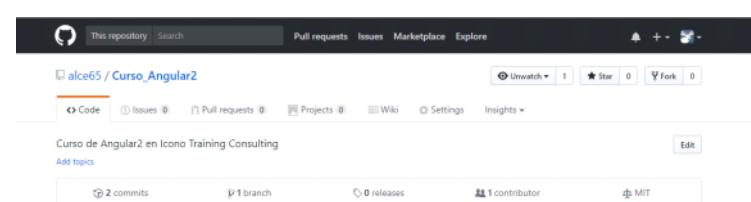
`git commit`



IMPORTANTE

`.gitignore`

```
# Dependency directories
node_modules/
jspm_packages/
```



dependency directories
node_modules/
jspm_packages/

The screenshot shows a GitHub repository page for 'Proyectos Hola Mundo'. At the top, there are navigation links for Code, Issues, Pull requests, Projects, Wiki, Settings, and Insights. Below the header, it says 'Curso de Angular2 en Icono Training Consulting'. The repository has 2 commits, 1 branch, 0 releases, 1 contributor, and is licensed under MIT. A 'New pull request' button is visible. The repository contains files: .gitignore, LICENSE, README.md, and icono.png. The README.md file is expanded, showing its content: 'Curso Angular2'. The page footer indicates it's from 'Icono Training Consulting'.

Angular cli: despliegue

domingo, 10 de septiembre de 2017 10:40

Angular-cli incluye un comando para preparar el despliegue de la aplicación

Cuando queremos publicar la aplicación en **producción** tenemos que generar los archivos optimizados y publicarlos en un servidor web

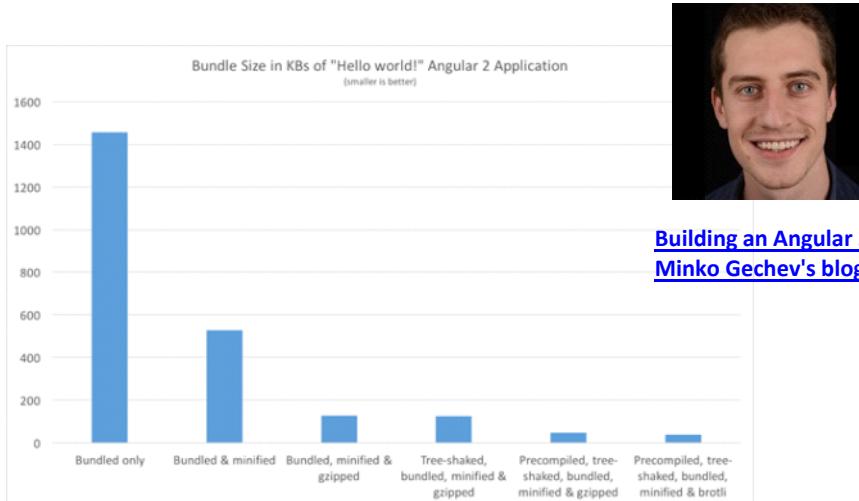
`ng build`

se generan en la carpeta *dist* los ficheros conocidos como *bundle*

Inicialmente no optimizaba el resultado (main.bundle.js ocupa 2,5 megas),

Sucesivas versiones de angular-cli han ido ajustando la configuración de *webpack* hasta llegar a un *bundle* de 50Kb

Mejoras del bundle



[Building an Angular 2 Application for Production – Minko Gechev's blog](#)

<http://blog.rangle.io/optimize-your-angular2-application-with-tree-shaking/>

Generamos la aplicación Hola-mundo para comparar los *bundles* con los que se generan temporalmente en desarrollo

Se optimiza con la opción *ng build -t production*

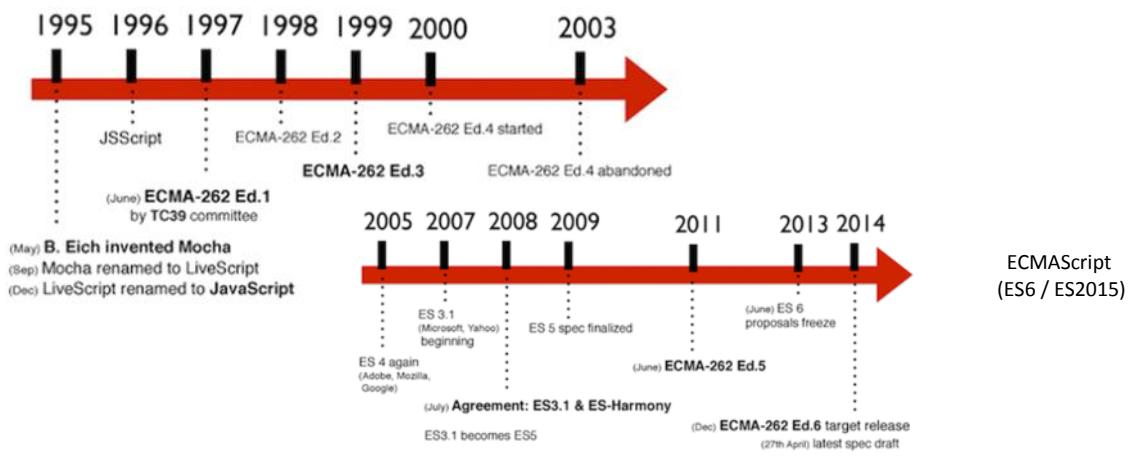
ECMAScript 6 (ES6 / ES2015)

sábado, 9 de septiembre de 2017 13:33

Brendan Eich
Netscape



European Computer Manufacturers' Association



- Constates (`const`). Variables con ámbito (`let`)
- Función Arrow. This "semántico"
- *Template Strings*: interpolación de variables
- Valores por defecto
- Clases (`class`)
- Módulos (`export / import`)
- Promesas (`promise`)
- *Destructuring* ...

Más información

JS

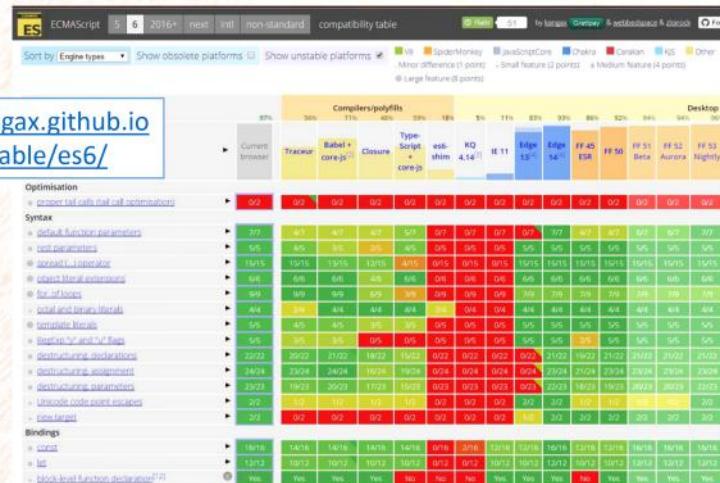
ECMAScript 6 — New Features: Overview & Comparison

<http://es6-features.org/>

<http://kangax.github.io/compat-table/es6/>



TRAINING



Nuevos elementos de código

domingo, 30 de julio de 2017 22:19

- **Constates (const).** Variables con ámbito (let)
- **Función Arrow.** This "semántico"
- **Template Strings:** interpolación de variables
- Valores por defecto

El uso de var sigue siendo identico a versiones anteriores, usandose en este caso para declarar un array

const y let

Salida por consola utilizando "template strings" en los que se conservan los saltos de línea

```
// Ejemplo de código en ES6
var data = [{precio: 12}, {precio: 34}, {precio: 19}];
data.forEach( elem => {
  if (true) {
    const iva = 1.16
    let precioFinal = elem.precio * iva
    console.log(`Oferta:
      El precio final es ${precioFinal}`);
  }
})
// console.log (iva)
```

la función callback del método forEach, propio de ESS se define con el nuevo formato "Arrow function" con elem como único argumento

línea que daría error por hacer referencia a una variable en un ámbito en el que no existe

Clases

sábado, 29 de julio de 2017 15:30

NO EXISTEN
Propiedades definidas
fuera de los métodos

Azúcar sintáctico.
En JS NO EXISTEN CLASES
Sólo hay PROTOTYPES

La nueva forma de escribir en ES6 hace más sencillo el uso de los prototipos al asimilarlos a la forma habitual de trabajar con clases

Módulos

sábado, 29 de julio de 2017 15:30

Creación de un módulo en el que se exporta una función, escrita en el nuevo formato "arrow function"

```
//File: lib/sample.js
definición de → module "sample" {
    un módulo
función →     export hello = (nombre) => {
    exportada      return "Hola " + nombre;
}
}
```

Uso del módulo anteriormente creado

```
importación de → //File: app.js
una función → import { hello } from "sample";
objeto en el que se usa la → var app = {
función importada   saludo : () => {
                    hello("Carlos");
                }
}
app.saludo()
objeto exportado → export app;
```

NO DISPONIBLE EN
LOS NAVEGADORES →



<https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias>

La sentencia `import` se usa para importar funciones que han sido exportadas desde un módulo externo, otro script, etc.

Nota: Esta característica aún no es implementada en ningún navegador por el momento. Esto es implementado en muchos transpiladores, tales como [Traceur Compiler](#) y [ES6 Module Transpiler](#).

La declaración `export` es usada para exportar funciones, objetos o tipos de dato primitivos a partir de un archivo (o módulo).

Note: Esta característica no ha sido implementada de forma nativa todavía. Está implementada en algunos transpiladores, como [Traceur Compiler](#), [Babel](#) o [Rollup](#).

Una promesa representa el resultado eventual de una operación.
Se utiliza para especificar que se hará cuando esa eventual operación de un resultado de éxito o fracaso.

Promesas

JS

Un objeto promesa representa un valor que todavía no está disponible pero que lo estará en algún momento en el futuro

Permiten escribir código asíncrono de forma más similar a como se escribe el código síncrono:

- La función asíncrona retorna inmediatamente y
- ese retorno se trata como un proxy cuyo valor se obtendrá en el futuro

El API de las promesas en Angular corresponde al servicio **\$q**

la biblioteca Q desarrollada por **Kris Kowal**

<https://github.com/kriskowal/q>



Promesas: \$q

JS

```
function getPromise()
```

```
    var deferred=$q.defer();
```

crea una promesa

```
    deferred.resolve()  
    deferred.reject()
```

resuelve la promesa en un sentido
u otro al cabo del tiempo

```
    return deferred.promise
```

devuelve la promesa

```
var promise = getPromise();
```

```
promise.then(successCallback,failureCallback,notifyCallback);
```

```
promise.catch(errorCallback)
```

```
promise.finally(callback)
```

promise.catch(errorCallback)

promise.finally(callback)



Promesas: ES6

JS

Implementación: new Promise

el objeto promesa recibe como parámetros dos funciones:

- La función "resolve": se ejecuta cuando queremos finalizar la promesa con éxito.
- La función "reject": se ejecuta cuando queremos finalizar una promesa informando de un caso de fracaso.

```
function hacerAlgoPromesa() {  
    return new Promise( function(resolve, reject) {  
        console.log('hacer algo que ocupa un tiempo...');  
        setTimeout(resolve, 1000);  
    })  
}
```



Promesas: ES6

JS

Utilización

a la función que retorna el objeto promesa se le encadenan dos:

- .then : la función que se ejecutará cuando la promesa haya finalizado con éxito.
- .catch : la función que se ejecutara cuando la promesa haya finalizado informando de un caso de fracaso.

```
hacerAlgoPromesa()
.then( function() { console.log('la promesa terminó.');
})
.catch( function() { console.log('la promesa fracasó.'));
```



Ejemplo de promesas en ES6

```
function msgAfterTimeout (msg, who, timeout) {
    return new Promise((resolve, reject) => {
        setTimeout(
            () => resolve(` ${msg} Hello ${who} !`),
            timeout)
    })
}

msgAfterTimeout("", "Foo", 100)
.then((msg) =>
    msgAfterTimeout(msg, "Bar", 200))
.then((msg) => {
    console.log(`done after 300ms:${msg}`))
```

Función que crea y devuelve un **objeto promesa**

En este caso, la promesa siempre se resuelve correctamente, creando un mensaje de saludo a un usuario

Utilización de las promesas, encadenando las llamadas a ellas

```
PS D:\Desarrollo\Angular2\Curso_Angular2\tecnologias\ES6> node .\sample.promise.js
done after 300ms: Hello Foo! Hello Bar!
PS D:\Desarrollo\Angular2\Curso_Angular2\tecnologias\ES6>
```

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promise

TypeScript

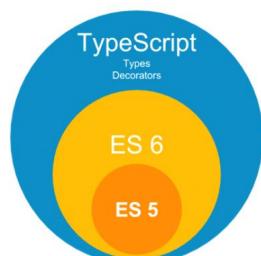
sábado, 9 de septiembre de 2017 13:34



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

Open source

Super Set de JavaScript ES6
(=> es JavaScript)



- Orientado a Objetos con clases
- Añade **tipos estáticos**
 - Inferencia de tipos
(no hay que declararlos en muchos sitios)
 - Tipos opcionales (si no quieres, no los usas)
- Anotaciones
- Es *transpilado*: se genera código JavaScript ES5 (compatible con los navegadores web actuales)

Documentación on-line

The screenshot shows the GitBook interface for the 'TypeScript Deep Dive' book. At the top, there's a navigation bar with links for Pricing, Explore, About, and Blog. On the right, there are buttons for Sign In and Sign Up. Below the navigation, the page title 'TypeScript Deep Dive' is displayed, along with a subtitle: 'I've been looking at the issues that turn up commonly when people start using TypeScript. This is based on the lessons from StackOverflow / DefinitelyTyped and general engagement with the TypeScript community. You can follow for updates and don't forget to star on Github.' There are also buttons for Download PDF and Read.

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

Basarat Ali Syed



Clases

sábado, 29 de julio de 2017 15:33

```
clase // ejemplo de clase en TypeScript
      export class Empleado {
propiedades   private nombre: string;
                private salario: number;
constructor     constructor(nombre: string, salario: number) {
                    this.nombre = nombre;
                    this.salario = salario;
}
métodos        getNombre() {
                    return this.nombre;
}
                    toString() {
                    return "Nombre:" + this.nombre +
", Salario:" + this.salario;
}
}
```

JS

Ejemplo de módulo más ES6

definición de un módulo

```
1 //File: lib/sample.js
2 module "sample" {
3
4   export hello = [ nombre ) => {
5     return nombre;
6   }
7 }
8
```

función exportada

importación de
una función

```
1 //File: app.js
2 import { hello } from "sample";
3 var app = {
4   foo: [ () => {
5     hello("Carlos");
6   }
7 }
8 export app;
9
```

uso de la función
importada

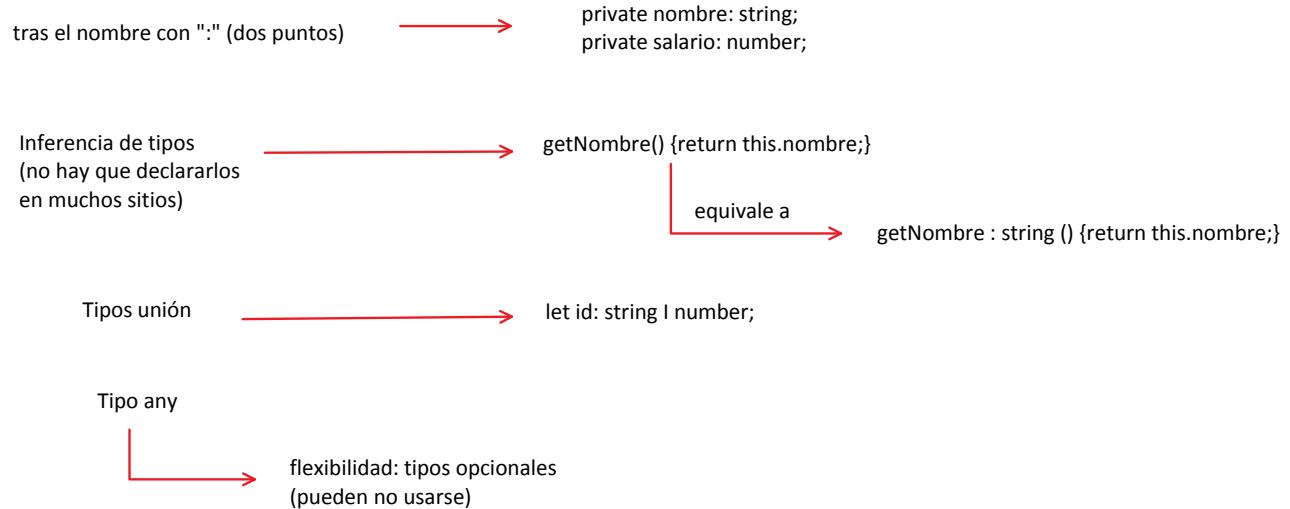
objeto
exportado



Tipos

domingo, 10 de septiembre de 2017 22:40

Tipos estáticos en tiempo de desarrollo;
evidentemente desaparecen tras la compilación (*traspilación*) a JS



Módulos (y elementos de ES6)

domingo, 10 de septiembre de 2017 22:49

empleado.ts
fichero en el que se crea y
exporta una clase

```
export class Empleado {  
    nombre : string;  
    salario: number;  
  
    constructor (pNombre, pSalario)  
    {  
        this.nombre = pNombre;  
        this.salario = pSalario;  
    }  
}
```

sample.ts
fichero en el que se importa y utiliza
la clase anterior

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

Importación desde otro módulo (fichero).
Se sobreentiende la extensión .ts

Tipo Array de objetos de la clase importada

código ES6 dentro de TypeScript

Anotaciones o decoradores

domingo, 10 de septiembre de 2017 23:04

Importación de una clase desde otro módulo (fichero)

decorador de la clase a la que acompaña

exportación de la clase "decorada"

```
import {Component} from 'angular2/core';

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

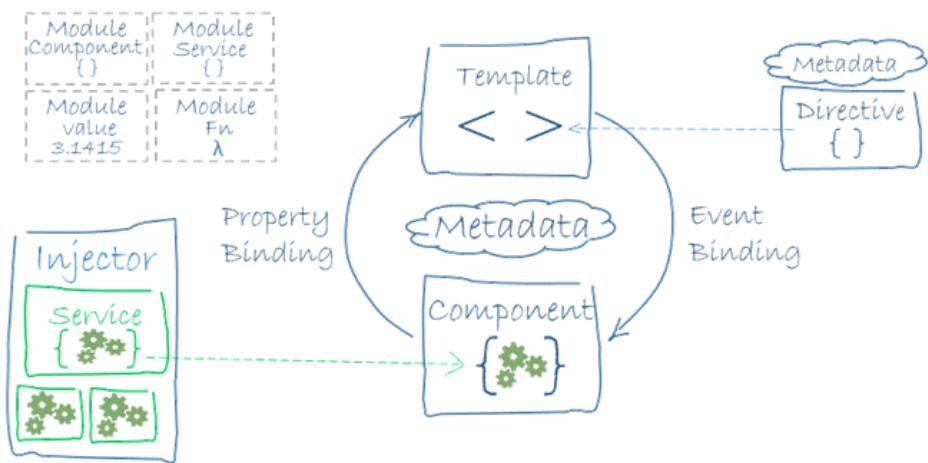
El decorador es un objeto JSON que da valor a una serie de METADATOS (propiedades) que esperan ser definidas y que pasarán a formar parte de la clase decorada

Otras características

domingo, 10 de septiembre de 2017 23:02

- Constructores
- Getter / Setter con sintaxis de atributo
- Type guards Instanceof / typeof
- Objetos literales “tipados” por interfaces
- Compatibilidad de tipos estructural
- Sobrecarga de métodos “especial”

Arquitectura Angular2



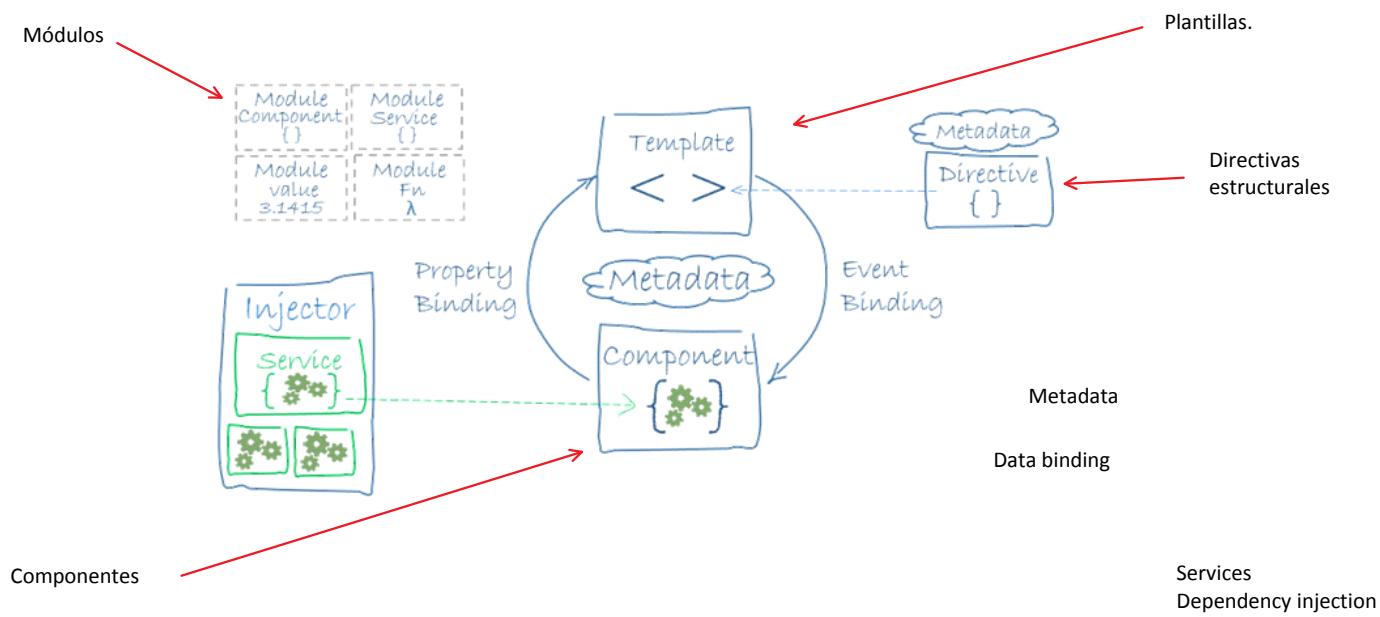
Elementos principales

Modules
Components
Templates
Metadata
Data binding
Directives
Services
Dependency injection

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

Módulos. Componentes. Vistas

lunes, 11 de septiembre de 2017 22:17



Módulos

Lunes, 11 de septiembre de 2017 22:23

Agrupación de componentes y otros elementos que son declarados en la decoración de una determinada clase

Toda app tiene

al menos un módulo que define los componentes de la app : AppModule

includiendo al menos el componente principal
AppComponent -> selector: app-root

Estructura de un módulo

- importación
- decoración
- clase exportada (vacía)

En cualquiera de las distribuciones, hay que diferenciar 2 procesos

- distribuir el código en módulos
- distribuir el código en ficheros

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

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [
    AppComponent
  ]
})
export class AppModule { }
```

Importación

Lunes, 11 de septiembre de 2017 22:32

Todo los ficheros de *typescript/JS* que tengan que utilizarse dentro del módulo tienen que ser importados



Desaparecen los <script> en el HTML durante el desarrollo

No se indica la extensión:

- en tiempo de desarrollo será TS
- en ejecución será JS, una vez *transpilado* y se reagrupará en el bundle.js

El comando *import* siempre implica dos elementos

```
import{ nombre de la clase }from 'donde esta la clase';
```

Si se omite el nombre de la clase {*} -> se importaran todas las que existan en el sitio indicado
(No es recomendable, por cómo funciona el *tree shaking*).

Dos posibles "orígenes" a la hora de importar elementos

por **nombre simbólico**, desde *node_modules*, los elementos de Angular y otras librerías

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

por **ruta**, desde un *path*, relativo a nuestra "base", los elementos de la aplicación
(esa "base" se define en los metadatos de index.html)

```
import { AppComponent }from './app.component';
```

Un módulo también puede ser importado desde otro, dando lugar a un árbol de módulos

Importación: index.ts



En TypeScript se pueden definir ficheros index.ts que exportan todos los ficheros de una carpeta.

- simplifica la importación desde otros ficheros
- desacopla los tipos del fichero en el que se declaran

index.ts

```
export * from './app.component';
export * from './app.module';
```

Fichero que lista todos los ficheros TS de una carpeta

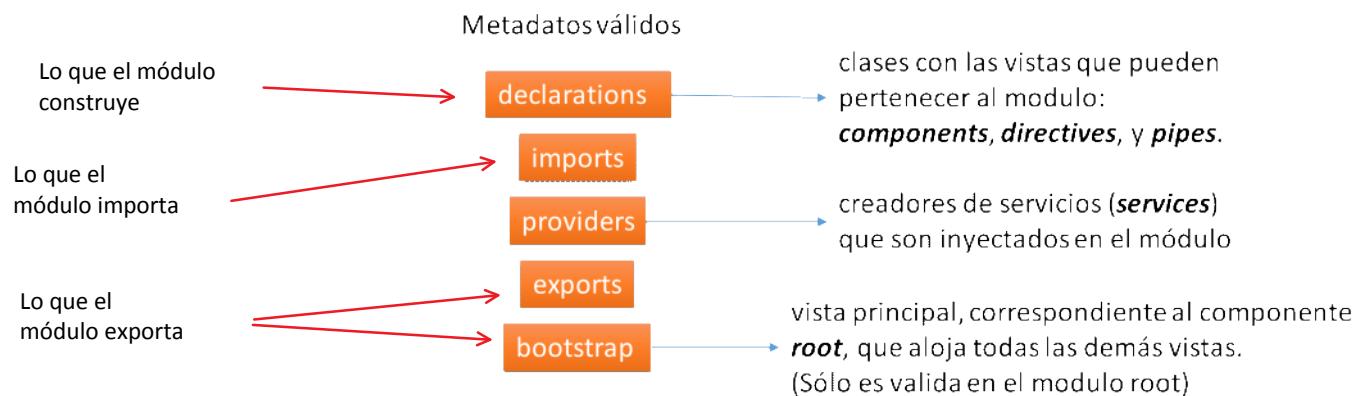
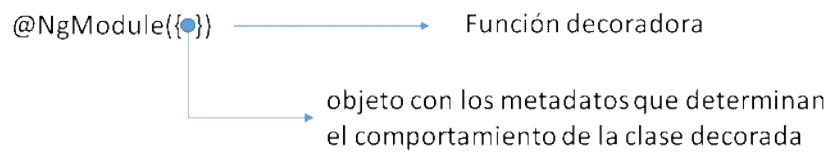
src/main.ts

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

Al importar ese fichero se puede referenciar cualquier clase de la carpeta (similar a paquetes Java)

Decoración / Anotación

Lunes, 11 de septiembre de 2017 22:49



Componentes

lunes, 11 de septiembre de 2017 22:52

Un componente es una nueva **etiqueta HTML**
con una **vista** y una **lógica** definidas por el desarrollador

vista → plantilla (template) en HTML con elementos
especiales propios de Angular

lógica → clase TypeScript (ES6) vinculada a la vista

Estructura de
un componente →

- importación
- decoración
 - selector
 - template / templateUrl
- clase exportada

Importamos al menos la
clase *Component* de Angular

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

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

Dicha clase puede usarse
en forma de decorador

Este decorador concreto admite
determinados metadatos

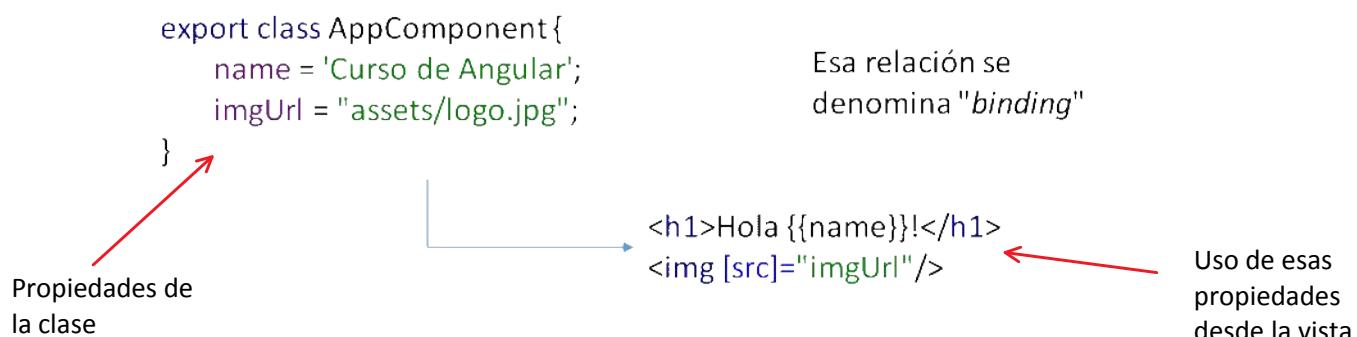
Nombre mediante el que podrá ser importada
para que forme parte de un módulo
(suele coincidir con el nombre del fichero)

Vistas HTML

lunes, 11 de septiembre de 2017 22:54

La **vista** del componente (**HTML**) se genera en función de dos elementos

- su estado, definido por el valor de los **atributos de la clase** en un determinado momento
- la plantilla o *template* que tiene asociado el componente, donde además de HTML puede haber referencia a dichos atributos



Recursos de la aplicación

lunes, 11 de septiembre de 2017 23:19

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



Nuevos componentes

lunes, 11 de septiembre de 2017 23:21

Utilizamos angular-cli para generar la base de un nuevo componente

```
ng g(enerate) c(omponent) "nombre"
```

Se habrá añadido directamente en el módulo

- como import
- como declaration
- Dispondremos de una carpeta con la estructura de archivos.

Inicialmente crearemos un componente "dumpy" :

stateless, incluso sin lógica ninguna.

Lo incorporaremos al componente principal

Cuando utilizamos angular-cli para generar la base de un nuevo componente,

la clase se crea con una estructura de partida, que es la recomendada en todos los componentes

```
export class <nombre> implements OnInit {  
    constructor() {} : inyección de dependencias  
    ngOnInit() {} : inicialización de valores
```

```
}
```

```
export class FeatureComponent implements OnInit {  
    constructor() {}  
  
    ngOnInit() {}  
}
```

Ejemplos

Junes, 11 de septiembre de 2017 23:22

app-root

Hola Mundo 2 componentes

app-pie

Creamos una nueva aplicación

ng new hola-componentes

Modificamos el componente principal como en casos anteriores

Creamos un nuevo componente

ng g c pie

En el módulo principal se añade

```
import { PieComponent } from './pie/pie.component';
@NgModule({
  declarations: [
    AppComponent,
    PieComponent
  ],
  ...
})
```

Cambiamos la lógica (*controller*) de nuestro componente

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-pie',
  templateUrl: './pie.component.html',
  styleUrls: ['./pie.component.css']
})
export class PieComponent implements OnInit {
  public formador: string
  public empresa: string
  public fecha: string

  constructor() {}

  ngOnInit() {
    this.formador = "Alejandro Cerezo Lasne"
    this.empresa = "Icono Training Consulting"
    this.fecha = "2017"
  }
}
```

Cambiamos el contenido de la vista de nuestro componente

```
<footer>
  <p>{{formador}} - {{fecha}}</p>
  <p>{{empresa}}</p>
</footer>
```

Cambiamos el CSS específico de nuestro componente

```
footer {
  position: fixed;
  bottom: 0;
  width: 100%;
  border-top: 1px papayawhip solid
}
p {
  text-align: center;
  font-size: 1.3em;
  color: papayawhip;
}
```

Consumimos el componente <app-pie> desde la vista del componente principal

```
<header style="text-align:center">
  <h1>
    Bienvenidos al curso {{curso}}!
  </h1>
  
</header>
<app-pie></app-pie>
```



Alejandro Cerezo Lasne - 2017
Icono Training Consulting

Plantillas (*Templates*)



Las plantillas (*templates*) permiten definir la vista en función de la información del componente

- Desarrollo declarativo. Expresiones
- Directivas estructurales
- Visualización condicional
- Repetición de elementos
- Estilos
- Formularios

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

Desarrollo declarativo



desarrollo declarativo → **expansión** de las características del **HTML**, añadiéndole **funcionalidades** sin necesidad de escribir código JavaScript

Se puede ver como una forma de **agregan valor semántico** al HTML.

Directivas estructurales **atributos *ng-** específicos de angular que se pueden asignar a etiquetas HTML.

Expresiones lenguaje de **plantillas** mediante **{{}}**

En todas ellas es posible añadir el prefijo **data-** para que las directivas no supongan problema de validación

Directivas y compilación



atributos específicos de angularJS que se pueden asignar a cualquier etiqueta HTML.

un elemento del DOM queda "marcado" para que AngularJS le asigne un determinado **comportamiento**, que puede suponer incluso una transformación de ese elemento del DOM o alguno de sus hijos

Si modifican la estructura del DOM se denominan **directivas estructurales**.
Su nombre comienza por * (*ngIf, *ngFor,*ngSwitch)

Para que este proceso tenga lugar, Angular incorpora un **compilador de HTML (HTML Compiler)** cuya función es

- recorrer el documento y localizar las directiva
- ejecutar los comportamientos asociados a esas directivas.

El atributo o etiqueta se denomina **ng-nombre**; el método asociado **ngNombre**

Templates: Lógica básica



Combinando directivas, eventos y expresiones tenemos:

- Respuesta a eventos
`<button (click)="setName('Pepe')">`
- Entrada de datos (data binding)
`<input type="text" [ngModel]="name">`
- Datos enlazados (two-way data binding)
`<input type="text" [(ngModel)]="name">`
`{name}`

Lo comparamos con los mismos procesos en JS

Directiva ngModel



relaciona elementos del DOM con modelos de datos, informando al compilador HTML de AngularJS de que se está declarando una variable del modelo.

- ✓ Se utiliza en los controles de formulario, como INPUT, SELECT, TEXTAREA o controles personalizados.
- ✓ enlaza (*binding*) una propiedad de la clase que define al componente con el correspondiente control de formulario de la vista
- ✓ este "enlace" puede funcionar sólo en una dirección o en las dos,
- ✓ según se invoque como propiedad [] o como evento ()

"banana in
a box"

Sirve de base al **doble binding**
[(ngModel)] = "sName"

Gestión de eventos



- () indicando su nombre dentro de los paréntesis, permite definir el **manejador** del evento cualquier estándar de un determinado elemento del DOM.

- puede ser una expresión asociada al evento

(click) = "setContador(2)"

- una llamada a una función definida en la clase que constituye el componente

(click) = "++nCount"

Eventos del sistema (1)

Evento	Descripción	Elementos para los que está definido
blur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
change	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
click	Pinchar y soltar el ratón	Todos los elementos
dblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
focus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
keydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
keypress	Pulsar una tecla	Elementos de formulario y <body>
keyup	Soltar una tecla pulsada	Elementos de formulario y <body>
load	La página se ha cargado completamente	<body>

Eventos del sistema (2)

Evento	Descripción	Elementos para los que está definido
mousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
mousemove	Mover el ratón	Todos los elementos
mouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
mouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
mouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
reset	Inicializar el formulario (borrar todos sus datos)	<form>
resize	Se ha modificado el tamaño de la ventana del navegador	<body>
select	Seleccionar un texto	<input>, <textarea>
submit	Enviar el formulario	<form>
unload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

Nuevos eventos en HTML5



Window

- onafterprint
- onbeforeprint
- onbeforeunload
- onerror
- onhaschange
- onmessage
- onoffline
- ononline
- onpagehide
- onpageshow
- onpopstate
- onredo
- onresize
- onstorage
- onundo

Form

- oncontextmenu
- onformchange
- onforminput
- oninput
- oninvalid

Mouse

- ondrag
- ondragend
- ondragenter
- ondragleave
- ondragover
- ondragstart
- ondrop
- onmousewheel
- onscroll

Media Events

- oncanplay
- oncanplaythrough
- ondurationchange
- onemptied
- onended
- onerror
- onloadeddata
- onloadedmetadata
- onloadstart
- onpause
- onplay
- onplaying
- onprogress

- onratechange
- onreadystatechange
- onseeked
- onseeking
- onstalled
- onsuspend
- ontimeupdate
- onvolumechange
- onwaiting

Doble binding



- expresiones
- directivas ngModel

```
<form name="formulario">
    <label for="fNombre" id="lblNombre">Indica tu nombre</label>
    <input type="text" id="fNombre" name="fNombre"
        [(ngModel)]="nombre">
</form>
<p>Hola {{nombre}}.</p>
```

binding

información desde el
controller a la vista

two-way binding

información desde la
vista al *controller*.



Formulario: Doble binding

Comprobamos el doble *binding* en un sencillo formulario que en la parte inferior reproduce el input en el momento en que se escribe

Formulario de saludo

Indica tu nombre

Hola **Pepe**.
Bienvenid@ al curso de **Desarrollo Web Front-End**

Data binding y respuesta a eventos

El doble *binding* se ve aún más claro si añadimos un sencillo botón, que también nos anticipa como se gestiona un evento en angularJS con la directiva (**click**).

Podemos reproducir el ejemplo sin AngularJS para comprobar la cantidad de código que se hace innecesario al usar el *framework*

Expresiones



- Lógica limitada: no se pueden incluir en ellas condicionales (excepto el operador ternario), bucles o excepciones
- Se les puede dar formato mediante filtros

Referencias a modelos

`{{Dato}}`

Operaciones
aritméticas básicas

`{{Dato + 4}}`

Empleo de los métodos de
los objetos envolventes de
JS (e.g. String)

`{{"Beginning AngularJS".toUpperCase()}}`

`{{"ABCDEFG".indexOf('D')}}`

Uso del operador ternario

`{{Dato == 1 ? "Red" : "Blue"}}`



Ejemplos de expresiones

Ejemplos de Expresiones

Operaciones aritméticas básicas

$6 + 4 = 10$

El resultado se obtiene con la expresión: `{(6 + 4)}`

Empleo de los métodos de los objetos envolventes de JS (e.g. String)

BEGINNING ANGULARJS

Resultado de la expresión: `{"Beginning AngularJS".toUpperCase()}`

3

Resultado de la expresión: `{"ABCDEFG".indexOf('D'))}`

Uso del operador ternario

Red

Resultado de la expresión: `{(1==1 ? "Red" : "Blue")}`

Alejandro L. Cerezo - Madrid 2015



Iteraciones

*ngFor

Directiva estructural que genera el recorrido a una colección (un *array* o un objeto del modelo) indicándole la variable local a su ámbito donde se almacenará el elemento actual de cada iteración.

en la clase aElementos = [.....]

Existe un *array* en el modelo

```
<tag_html *ngFor="(let) elemento of aElementos | filtros | ordenación">  
  ... {{elemento}} ...  
</tag_html>
```

Similitud con el bucle **for ... in** de JS

Más adelante veremos las opciones de filtrado y ordenación

<https://angular.io/docs/ts/latest/api/common/index/NgFor-directive.html>

Ámbito de las iteraciones



Técnicamente, el código HTML iterado tendrá un ámbito (*scope*) local

- en él se pueden definir propiedades específicas de este ámbito, que se inicializan mediante let
- además existen una serie de propiedades ya predefinidas que toman valor dinámicamente conforme se realizan las sucesivas iteraciones y que pueden ser recogidas en propiedades del controlador

La más importante es **index** que devuelve en cada momento el índice en el recorrido del elemento actual sobre el que se está iterando

```
*ngFor="let elemento of aElementos; let i = index">
```

Scope de las iteraciones



El conjunto de las variables que almacena automáticamente el ámbito (*scope*) de una iteración es el siguiente

index	Number	Iterator offset of the repeated element (0..length-1)
first	Boolean	True, if the repeated element is first in the iterator
middle	Boolean	True, if the repeated element is between first and last in the iterator
last	Boolean	True, if the repeated element is last in the iterator
even	Boolean	True, if the iterator position \$index is even (otherwise, false)
odd	Boolean	True, if the iterator position \$index is odd (otherwise, false)

Pensamientos: ejemplo de iteraciones

Conforme añadimos ideas, creamos un *array* que mostramos, iterando sobre el, en la parte interior

Junto con la iteración, vemos de nuevo como funciona el doble *binding*.

Pensamientos

Indica tu nombre

Comparte una idea

Hola Pepe

Pensamientos que has tenido

- Tu pensamiento numero 1 fue: "Una idea"
- Tu pensamiento numero 2 fue: "Segunda idea"
- Tu pensamiento numero 3 fue: "No se me ocurre nada"
- Tu pensamiento numero 4 fue: "Ya termino"

Alejandro L. Cerezo - Madrid 2015



Condiciones: *ngIf*

Se puede controlar si un elemento aparece o no en la página dependiendo del valor de un atributo de la clase usando la directiva `ngIf`

- dependiendo del valor del atributo booleano `visible`

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

- dependiendo de una **expresión**

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

Muestrario: mostrar y ocultar (1)

Uso de la directiva

*ngIf

- asociándola al evento clic en un botón
- asociándola al paso del ratón sobre una imagen

Separamos el *footer* del fichero principal y lo incorporamos con otro componente

Muestrario

Ratones inalambricos disponibles en los siguientes colores



Ocultar Colores Disponibles

Rojo

Verde

Azul

Combinando ngFor / ngIf



No se pueden incluir dos directivas estructurales (de tipo *) en el mismo elemento

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

La solución más simple es anidar elementos HTML (divs), cada uno con su directiva

También es posible usar la versión de las directivas sin el azúcar sintáctico (*), en su versión extendida con el elemento *template* (que no aparece en el DOM)

```
<template ngFor let-elem [ngForOf]="elems">  
    <li *ngIf="elem.check">{{elem.desc}}</li>  
</template>
```

<https://github.com/angular/angular/issues/4792>

Estilos iniciales



Existen varias formas de definir inicialmente un CSS en Angular 2

- **Globalmente** en el **fichero css** asociado al index.html
- Local al componente:
 - En la propiedad styles de @Component
 - En el **fichero css** definido en styleUrls de @Component
 - En el template

Gestión de estilos



Directamente

- Asociar un estilo concreto de un elemento a un atributo

Mediante clases (Buena práctica)

- 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 objeto (mapa de string a boolean)



Style (1)

[style.<nombre>]

permite asignar a un elemento del DOM una propiedades de estilo almacenadas como un string en el modelo de la aplicación

```
<p [style.color] ="estiloColor">
```

Se pueden indicar fácilmente las unidades

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

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



Style(2)

[ngStyle] permite asignar a un elemento del DOM una o varias propiedades de estilo almacenadas como un objeto en el modelo de la aplicación

```
<p [ngStyle] ="estiloColor">
```

En el *controller*:

```
estilos = {"bachgrouun-color" : "green",
           "color": "silver"}
```

Como en cualquier otro contexto CSS, no es una buena práctica la aplicación directa de estilos, siendo más recomendable la aplicación de las clases adecuadas a cada caso

class (1)



[class] permite asignar a un elemento del DOM una clase mediante expresiones que hacen referencia al modelo. De esa forma al modificar el modelo se aplican clases diferentes y se modifica el aspecto del elemento

La propiedad del modelo puede tener varios formatos:

- una **cadena de caracteres** con uno o más nombres de clases

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

class controller : → nombreClase =..."

El cambio del valor de la variable se refleja en la aplicación de diferentes clases dinámicamente

class (2)



[class.<nombre>] = "boolean"

Es posible activar una clase concreta con un atributo boolean y se puede hacer con diversas clases

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

class controller : → redActive ="true"
 yellowActive ="false"



class (3)

[ngClass] permite asignar a un elemento del DOM una clase mediante expresiones más complejas

[ngClass] = "array"

Es posible aplicar diversas clases cuyos nombres se almacenan en un array

```
<h1      [ngClass]="['class1', 'class2', 'class3']"  
Title!</h1>
```

[ngClass] = "objeto"

- las claves permiten especificar nombres de clases y
- sus valores corresponden a expresiones que deben cumplirse para que éstas se apliquen.

```
<p [ngClass]="{positivo: total>=0, negativo: total<0}">
```



Acumulador con clases

Ejemplos de clases
que se aplican
dinámicamente

- en respuesta a una selección por el usuario
- en función del valor de un dato del modelo

Acumulador con clases

Acumulador

Elije en tamaño de letra del titular

Control de operación:

Incremento

Totales:

En el acumulador llevamos **10**

Alejandro L. Cerezo
CLE Formación
Madrid - 2015



Directivas y referencias

[src] indica la *url* del fichero que actúa como fuente para una etiqueta **img**, sustituyendo el habitual atributo *src*

```
<img [src]="fileName" >
```

En lugar de ``

La directiva cumple una función similar a *ng-bind* cuando se emplean expresiones, evitando que se muestre la expresión o un ícono antes de que haya sido cargada la imagen.

[href] cumple una función similar respecto a los hiperenlaces (etiqueta `<a>`), evitando que se pueda pinchar en uno de ellos con una expresión de AngularJS antes de que ésta se haya resuelto.

Slider de imágenes

Para comprobar el uso de la directiva `ngSrc`, creamos un slider capaz de mostrar una sucesión de imágenes.

Slider - Imágenes

Imágenes de Luis Royo



[Avanzar](#)

Alejandro L. Cerezo
CLE Formación

Madrid - 2015

Directivas y formularios



input
textarea
select

→ **[(ngModel)]**: indica la propiedad del modelo.
a la que se asocia el elemento

```
<label for="nif">NIF:</label>
<input id="nif" type="text" id = "nif" name="nif"
[(ngModel)]="oSeguro.nif" />
```

```
<label for="casado">Casado:</label>
<input id="casado" type="checkbox" id="casado" name="casado"
[(ngModel)]="oSeguro.casado" >
```

Se suele decir:

→ “Si el valor de una directiva ng-model no incluye un punto es que está mal.”

RadioButtons y Checkboxes



ngModel: como en cualquier control de formulario, indica la propiedad del modelo asociada al elemento

Radio-buttons

→ cada conjunto de ellos comparte el mismo valor de ngModel

Checkboxes

→ Si la propiedad tiene tipo, debe ser boolean

```
<input type="checkbox" [(ngModel)]="angular"/>
<label>Angular</label>
<input type="checkbox" [(ngModel)]="javascript"/>
<label>JavaScipt</label>
```

class controller :

angular:boolean
javascript:boolean

Checkboxes y arrays



*ngFor: permite utilizar un array de objetos asociado a un conjunto de controles

```
<span *ngFor="let item of items">
    <input type="checkbox"
        [(ngModel)]="item.selected"/> {{item.value}}
</span>
```

class controller :

```
items: Array<Object> = [
    {value:'Item1', selected:false},
    {value:'Item2',selected:false}
]
```

Checkboxes en Angular 1.x



Checkboxes

ngTrueValue: permite asignar un valor personalizado al elemento cuando el campo *checkbox* está marcado.

ngFalseValue: es lo mismo que ngTrueValue, pero en este caso con el valor asignado cuando el campo no está marcado.

ngChange: sirve para indicar operaciones a realizar cuando se produce un evento de cambio en el elemento. Se dispara cuando cambia el estado del campo, marcado a no marcado y viceversa. El valor puede ser una expresión o una llamada a una función del *scope*.

Radio-Buttons



ngModel: se asocia a una misma propiedad del controller en todos los RB de un grupo.

su valor vera el correspondiente al atributo value del RB seleccionado

```
<input type="radio" name="gender"
[(ngModel)]="genero" value="Male"><label>Male</label>
<input type="radio" name="gender"
[(ngModel)]="genero" value="Female"><label>Female</label>
```

class controller : → genero: string

Formularios: select / options



En HTML, cada "option" puede tener 2 componentes

```
<option value="valor opcional">Etiqueta</option>
```

Angular únicamente añade la directiva **ngModel** para recoger el valor (si existe) o la etiqueta de la opción seleccionada

```
<select name="country" ng-model="user.country">
  <option value="">Please select an option</option>
  <option value="US">United States</option>
  <option value="GB">United Kingdom</option>
  <option value="AU">Australia</option>
</select>
```

Angular añade una opción en blanco a no ser que exista una con valor ""

Select / options desde el modelo



***ngFor
ngValue**

permite **crear automáticamente** un select/options a partir de un conjunto de datos, procesando un array de objetos (altems)

```
<select id="select" [(ngModel)]="resultado">  
<option *ngFor="let elem of aElementos" [ngValue]="elem">  
{{elem.nombre}} </option>
```

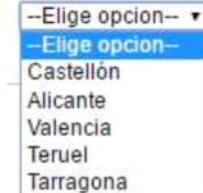
ngValue es el responsable de recoger los valores de los elementos (en este caso objetos completos) y de que el seleccionado se almacene en la variable asociada a ngModel

corresponde al ngOptions de Angular 1.x



Ejemplo de select/options

```
provincias=[  
    {idProvincia:2, nombre:"Castellón"},  
    {idProvincia:3, nombre:"Alicante"},  
    {idProvincia:1, nombre:"Valencia"},  
    {idProvincia:7, nombre:"Teruel"},  
    {idProvincia:5, nombre:"Tarragona"}  
];  
provinciaSeleccionada=null
```



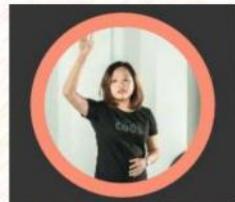
```
<select [(ngModel)] = "provinciaSeleccionada"  
        <option *ngFor = "let provincia of provincias"  
               [ngValue] = "provincia"> {{provincia.nombre}}  
        </option>  
</select>
```

Formularios: más información



[https://scotch.io/tutorials/
how-to-deal-with-different-
form-controls-in-angular-2](https://scotch.io/tutorials/how-to-deal-with-different-form-controls-in-angular-2)

Jecelyn Yeen



Angular 2 Form Controls

Code Demo

angular2 angularJS javascript typescript

How to Deal with Different Form Controls in Angular 2

How to Deal with Different Form Controls in Angular 2
(with latest forms module)



jecelyn Yeen

jul 18, 2016

Tutorials

Comments

0 🔥



Formulario: Selección de opciones

Ejemplo de formulario con 2 de los mecanismos habituales de selección de opciones: *check box* y *select / options*

Formulario

Selección de opciones

- Imprimir resultado
- Tono claro

Provincia —Elige opción-- ▾

Resultado

- Opción print seleccionada: false
- Opción claro seleccionada: oscuro
- Provincia elegida:

Alejandro L. Cerezo - Madrid 2015

Lista de tareas (1): formulario

- Añadimos un formulario que permita recoger la descripción de una tarea y que incluya un botón añadir tarea. Mediante ngSubmit vinculamos el botón a una función manejadora.
- A continuación de cada item añadimos un botón que nos permita eliminarlo

Tareas

<input type="text" value="Describe una tarea"/>	<input type="button" value="Añadir"/>
Preparar curso de Angular	<input type="button" value="X"/>
Preparar curso de Web Components	<input type="button" value="X"/>

♥ CLE Formación - Curso de Angular



Lista de compras

A checklist

There are no items yet.

Enter the description... Add

Ejemplo de "formulario" para gestionar una lista de tareas

A checklist

<input checked="" type="checkbox"/> Leche	<input type="button" value="Delete"/>
<input checked="" type="checkbox"/> Pan	<input type="button" value="Delete"/>
<input type="checkbox"/> Galletas	<input type="button" value="Delete"/>

Vino Add

Métodos de arrays:
array.push()
array.indexOf()
array.splice()



Formularios: validación

Directivas

- **ng-required**: valor booleano: cuando es true marca un campo como obligatorio.
- **ng-maxlength**: indica el número máximo de caracteres permitidos en un campo.
- **ng-minlength**: : indica el número mínimo de caracteres permitidos en un campo.
- **ng-pattern**: Valida un campo frente a una expresión regular (regex).

form la propia etiqueta HTML es además una directiva Angular
Instancia automáticamente un controlador *FormController*,
que registra todos los controles del formulario y sus estados.
Si el formulario tiene nombre (atributo *name*)
la instancia se registra en *\$scope* con ese nombre



Validación: propiedades

La instanciación automática del controlador FormController, y su registro en \$scope con el nombre del formulario expone una serie de **propiedades** tanto de este como de sus controles

- \$pristine** : true si el usuario aún no ha interaccionado con el formulario o control
- \$dirty** : es el opuesto al anterior

Estas propiedades permiten no mostrar mensajes de validación hasta que el usuario ha comenzado a llenar el formulario

- \$valid** : true si se cumplen las condiciones impuestas por la directivas
- \$invalid** : es el opuesto al anterior

Estas propiedades permiten determinar la validez de cualquier control para hacer visibles o no los correspondientes mensajes, e.g utilizando la directiva ngShow.



Validación: ejemplo

```
<form name="myform" novalidate> ← Anulamos la validación  
estándar HTML5
```

```
<input type="text" id="firstname" name="firstname"  
ng-model="user.firstname" ng-required="true" ng-minlength="2">
```

```
<span class="error-message"  
ng-show="myform.firstname.$dirty && myform.firstname.$error.required">  
El nombre es obligatorio</span>
```

Para cada directiva de validación existe una propiedad \$error que tomara un valor true o false según las circunstancias

Formularios: validación

Realizamos un formulario con:

- los campos Nombre y Apellido obligatorios y de un mínimo de 2 caracteres
- el campo Teléfono de exactamente 9 caracteres exclusivamente numéricos:
ng-pattern = "/^\\d{9}\$/"

Formulario

Datos personales

Nombre	<input type="text" value="P"/>
El nombre debe tener un mínimo de 2 caracteres	
Apellidos	<input type="text"/>
Teléfono	<input type="text"/>

Resultado

- Nombre:
- Apellido:
- Teléfono:

Alejandro L. Cerezo - Madrid 2015

Comunicación entre componentes

miércoles, 13 de septiembre de 2017 19:22

Formas de comunicación

Comunicación entre un componente padre (contenedor) y un componente hijo (incluido en el anterior)

- 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)

Los inyectables (servicios) son objetos *singleton* y por tanto compartidos entre los distintas clases que los instancian

<https://angular.io/docs/ts/latest/cookbook/component-communication.html>

Configuración de propiedades

miércoles, 13 de septiembre de 2017 19:54

(Padre → Hijo)

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

vista
(padre) → <hijo [title]='appTitle'></hijo>

El valor de title en el hijo corresponderá a la propiedad appTitle en el padre

class controller:
(hijo)

@Input()
private title: string;

vista
(hijo)

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

Envío de eventos

miércoles, 13 de septiembre de 2017 19:56

(Hijo → Padre)

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

El padre se suscribe al evento : le asigna una función manejadora.
La variable \$event apunta al evento generado

vista
(padre)  `<header (evento)='hiddenTitle($event)'></header>`

class controller :
(hijo)

`@Output()
evento= new EventEmitter<boolean>();`

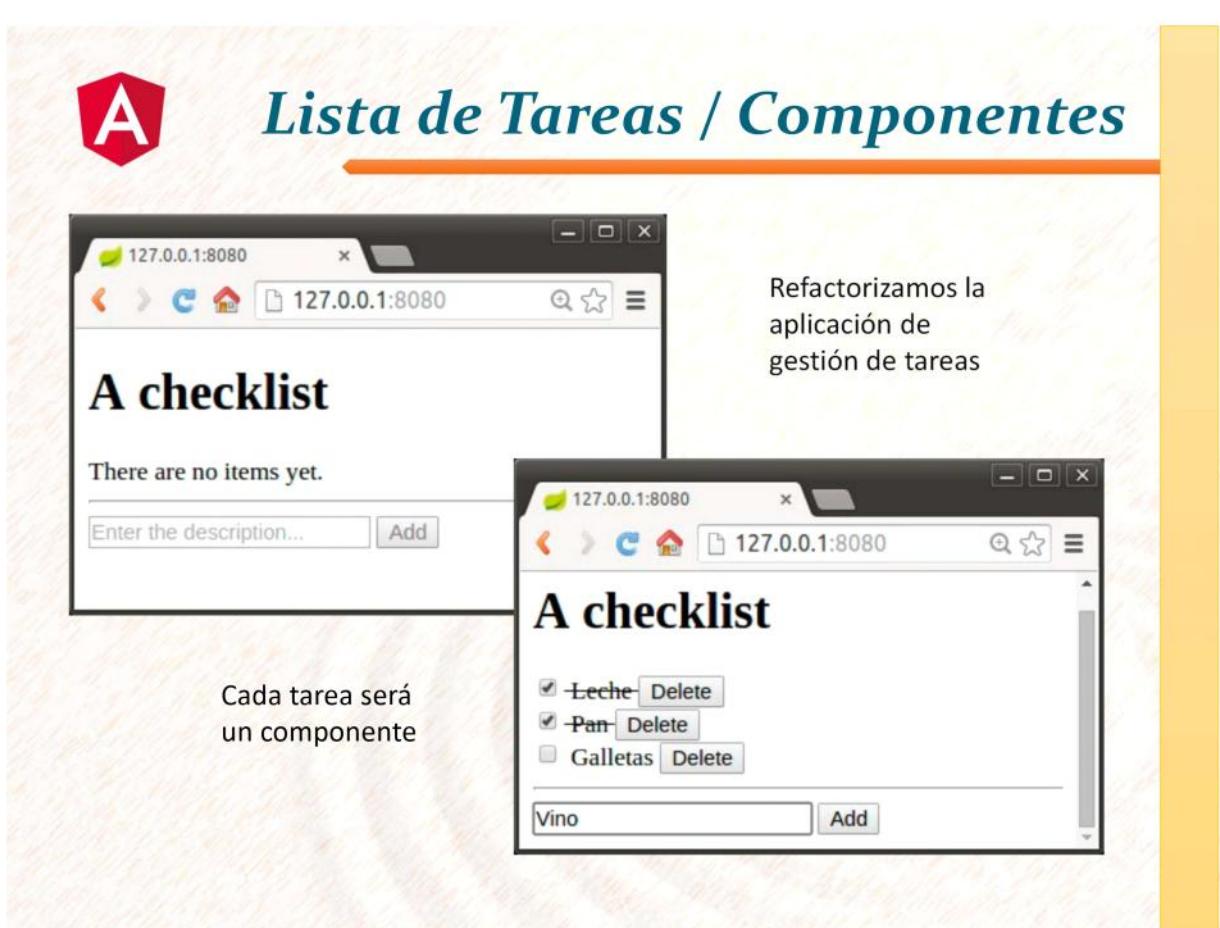
`(en alguna circunstancia)
this.evento.emit(parámetros)`

Ejemplo de comunicación entre componentes



- Cuando la lógica y/o el *template* sean suficientemente complejos
- Cuando los componentes hijos puedan reutilizarse en varios contextos

Los ejemplos del curso (y otros similares) suelen ser demasiado sencillos para que compense la creación de componentes hijos



Ciclo de vida de los componentes

miércoles, 13 de septiembre de 2017 22:20

I'm Todd, a Developer Advocate @Telerik. Founder of @UltimateAngular Creator of the ngMigrate. JavaScript, Angular, React, conference speaker. Developer Expert at Google.

ULTIMATE ANGULAR

Master Angular 1.x and Angular 2 with me online

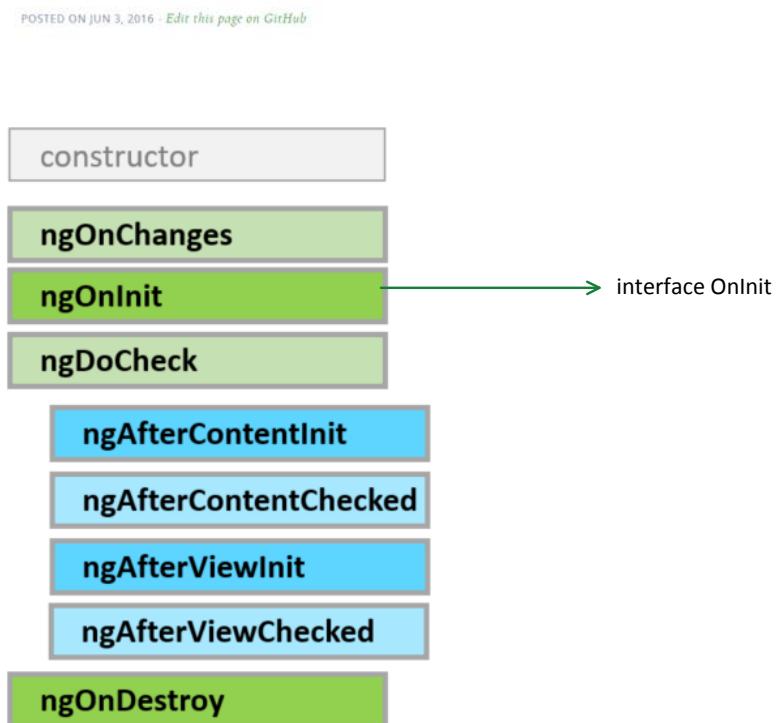
Limited Angular 2 preorders now available.
Master the latest Angular 1.5 components, or preorder the most in-depth Angular 2 courses.

See the courses >

Implementado en Angular 1.5

Lifecycle hooks in Angular 1.5

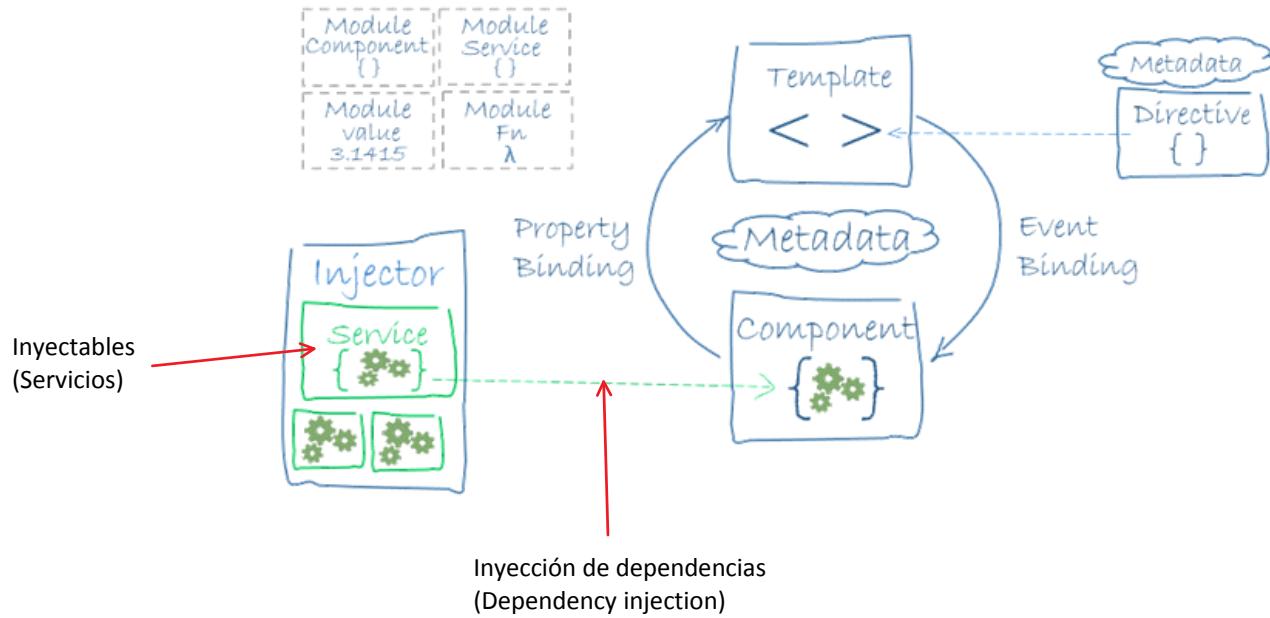
<https://toddmotto.com/angular-1-5-lifecycle-hooks>



<https://angular.io/docs/ts/latest/guide/lifecycle-hooks.html>

Inyectables (Servicios)

miércoles, 13 de septiembre de 2017 22:25



Servicios

Elementos de la aplicación que no se encargan del interfaz de usuario

- Son clave para modularizar la aplicación en elementos que tengan una única responsabilidad
 - Componente: Interfaz de usuario
 - Servicios (e.g. Peticiones http)
- Permiten la buena práctica de NO acoplar en el componente la lógica de negocio, e.g. las peticiones http
- Permiten reducir la complejidad de los componentes y facilitar que estos sean ampliados / modificados
- Facilitan la implementar tests unitarios al reducir el número de responsabilidades que tiene el componente

Servicios: características técnicas

- En principio se instancian según el patrón *singleton*: permiten compartir información entre componentes
- Los servicios mantienen el estado de la aplicación y los componentes ofrecen el interfaz de usuario
- Están anotados como injectable para que puedan ser inyectados en los componentes que necesitan utilizarlos
- Angular 2 ofrece muchos servicios predefinidos como la clase http utilizada para el acceso asincrónico (AJAX) a las APIs REST
- Lo habitual es que en cada proyecto de aplicación se implementen los servicios propios necesarios

Inyección de dependencias

miércoles, 13 de septiembre de 2017 22:38

- La técnica que permite solicitar objetos al *framework* se denomina inyección de dependencias
- Las dependencias que un módulo necesita son inyectadas por el sistema
- Técnica muy popular en el desarrollo de *back-end* en *frameworks* como Spring o Java EE
- En Angular 2 se realiza mediante el constructor de la clase controladora del componente

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