

Ξ AngularJS. Fundamentos

sábado, 16 de septiembre de 2017 19:36

- Fundamentos de AngularJS
 - *Frameworks en JS*
 - Presentación de AngularJS

- Versiones de AngularJS. AngularJS 1.5. Angular 2.x - 4.x
- Líneas maestras. MVC SPA
- Elementos de AngularJS
 - Vista y controlador. Doble *binding*
 - Módulos y estilos
- Evolución: Angular 1.5. Componentes

- Tecnologías implicadas
 - Entorno de trabajo
 - ECMAScript 6 (ES6)

- Navegadores y Servidor Web
- Editores de código
- Gestión del proyecto
 - Instalaciones: Node, npm
 - Versiones : Git, GitHub
 - Despliegue

- Funciones Arrow
- Clases
- Promises

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/>



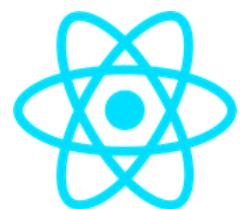
Ember.js

<http://emberjs.com/>



AngularJS

<https://angularjs.org>



React.js

<http://emberjs.com/>

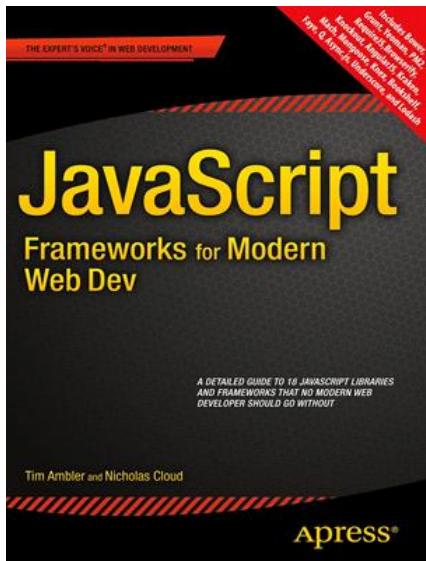
La elección de uno de ellos depende de los objetivos de cada proyecto



<http://todomvc.com/>



Frameworks interrelacionados



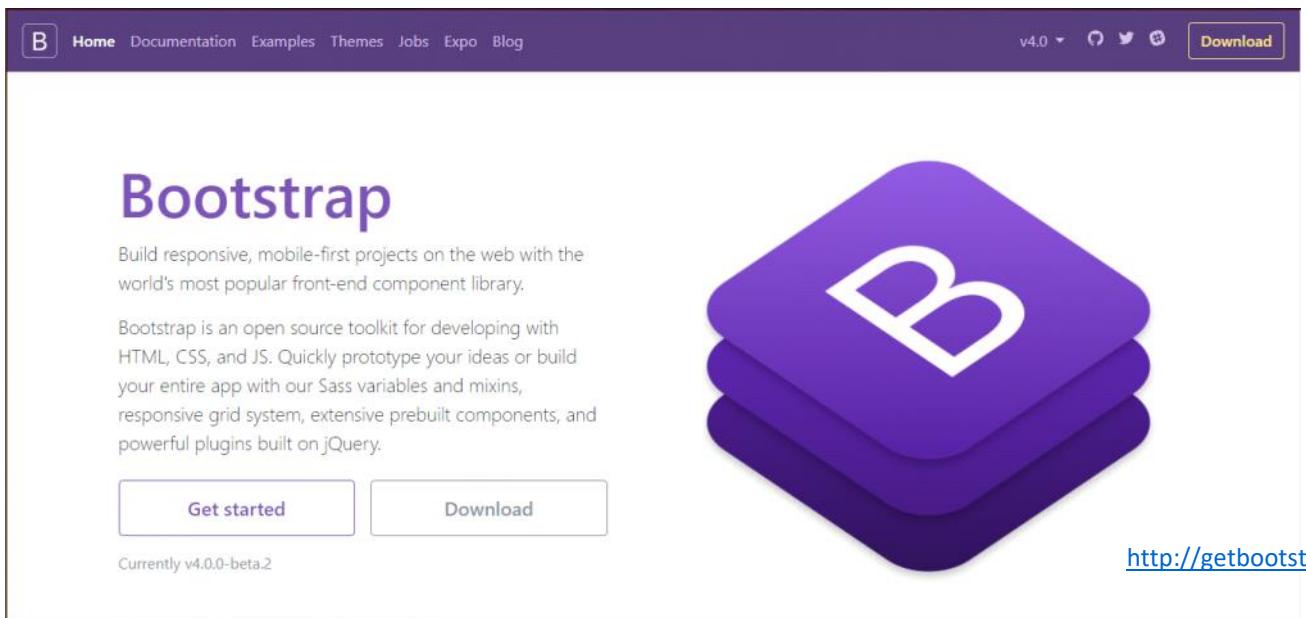
JavaScript Frameworks for Modern Web Dev
Tim Ambler & Nicholas Cloud
Apress, 2015

Contents at a Glance

About the Authors.....	xix
About the Technical Reviewer.....	xxi
Acknowledgments	xxii
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



AngularJS



The screenshot shows the official Bootstrap website. At the top is a purple header bar with the Bootstrap logo (a stylized 'B') and navigation links: Home, Documentation, Examples, Themes, Jobs, Expo, and Blog. To the right of the header are links for v4.0, social media icons (Facebook, Twitter, GitHub), and a 'Download' button. The main content area has a white background. It features a large 'Bootstrap' title in a purple font, followed by a subtitle: 'Build responsive, mobile-first projects on the web with the world's most popular front-end component library.' Below this is a paragraph describing Bootstrap as an open source toolkit for developing with HTML, CSS, and JS. It includes a 'Get started' button and a 'Download' button. At the bottom left is the text 'Currently v4.0.0-beta.2'. On the right side of the main content is a large, stylized purple icon of the letter 'B'.

<http://getbootstrap.com>

Definición

- framework* → modo de trabajar + conjunto de herramientas
- front-end / user interface (UI)* → permiten agilizar el desarrollo en la parte cliente.
- completo* →
 - Sistema de rejilla: esqueletos y estructuras HTML
 - Estilos predefinidos: CSS
 - Componentes: conjuntos de HTML y CSS para una determinada funcionalidad
 - Capa de comportamiento o interactividad: JS (JQuery)

Ventajas

- Aplicación desde la fase de diseño
- Compatibilidad con navegadores (desde IE8) –unificando su comportamiento
- Soporte avanzado para RWD (*Responsive Web Design*)
- Componentes de interfaz de usuario listos para usar
- Componentes y *plugins* útiles y habituales, basados en JQuery
- Agilidad durante el desarrollo

Responsividad

Bootstrap basa su diseño en el concepto de que los dispositivos móviles son lo más importante (*mobile first*)

Es importante incluir un valor adecuado para el *viewport*

Bootstrap 3

```
<meta name="viewport" content="width=device-width, initial-scale=1,  
maximum-scale=1, user-scalable=no">
```

Bootstrap 4

```
<meta name="viewport" content="width=device-width, initial-scale=1,  
shrink-to-fit=no">
```

AngularJS

sábado, 9 de septiembre de 2017 16:43

The screenshot shows the official AngularJS website. At the top, there's a navigation bar with links for Home, Learn, Develop, Discuss, and a search bar. The main header features the AngularJS logo (a red 'A' inside a hexagon) and the text "ANGULARJS by Google". Below the header, a tagline reads "HTML enhanced for web apps!". There are two prominent buttons: "Download AngularJS 1" (version 1.6.0-rc.2 / 1.5.9 / 1.2.32) and "Try the new Angular 2". Below these are links to "View on GitHub" and "Design Docs & Notes". Social media links for Facebook, Twitter, and GitHub are also present. A button at the bottom says "Learn Angular in your browser for free!".

<https://angularjs.org/>

The screenshot shows the official Angular website. It features a large red 'A' logo at the top left. The background is a blue gradient with a city skyline silhouette. Text on the page includes "One framework. Mobile & desktop." and a "GET STARTED" button. At the bottom, there's a banner for "Google Developer Day Beijing & Shanghai 12/2016" with a "REGISTER NOW" link.

<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*)

The screenshot shows the MEAN.io website. The header has a navigation menu with links for Home, Documentation, Packages, Release Notes, Support, Blog, and Contact. The main title is "The Friendly & Fun Javascript Fullstack for your next web application". Below the title, it says "MEAN is an opinionated fullstack javascript framework - which simplifies and accelerates web application development." A section titled "Get MEAN by running..." shows command-line instructions: "\$ sudo npm install -g mean-cli" and "\$ mean init yourNewApp". At the bottom, there's footer information: "LATEST RELEASE: v0.5.8", "LATEST COMMIT: Nov 23, 2015", and "FORKS: 2396".

Ejemplos de uso

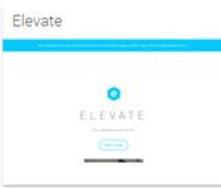
made with  N G U L A R

<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))
 - esta 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



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)

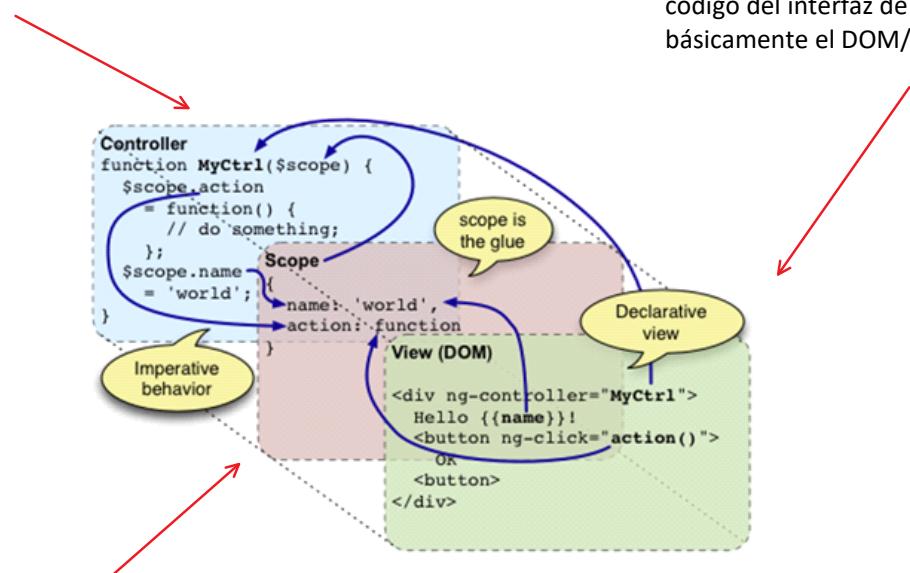
Más tarde fue re implementado
en Smalltalk-80



Model - View - Whatever

Controladores: Se encargarán de la lógica de la aplicación, incluyendo junto al "controller", "Factorías" y "Servicios" para mover datos contra servidores o memoria local en HTML5.

Vistas: Será la representación de los datos o la información, es decir, el código del interfaz de usuario, básicamente el DOM/HTML/CSS .



Modelo o Modelo de la vista,
según se emplee la variante del
patrón MVC o MVVC, 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.

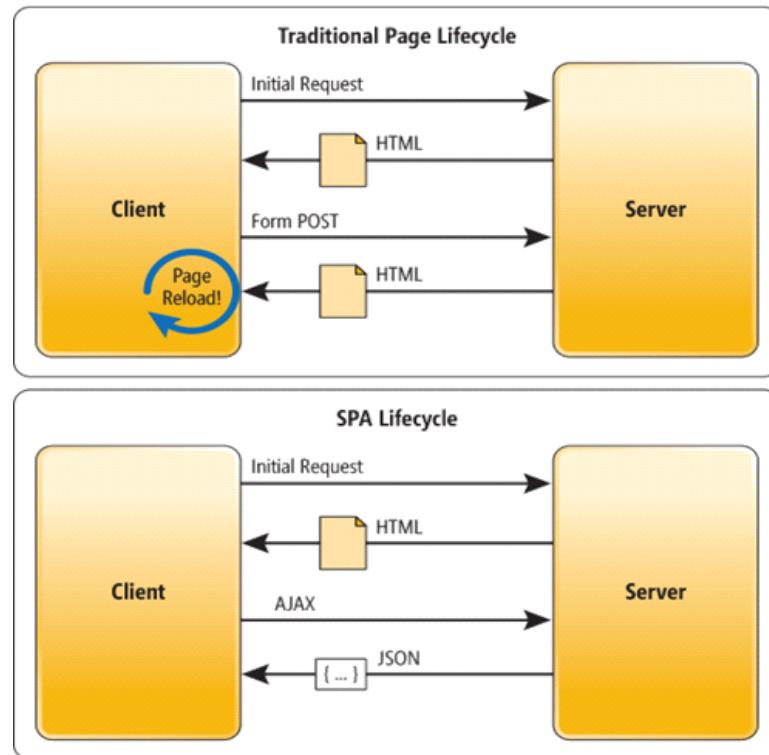
Ampliación: MVW en Angular

- MVC – Model-View-Controller
- MVVM – Model-View-View Model

SPA: *Single-Page Applications*

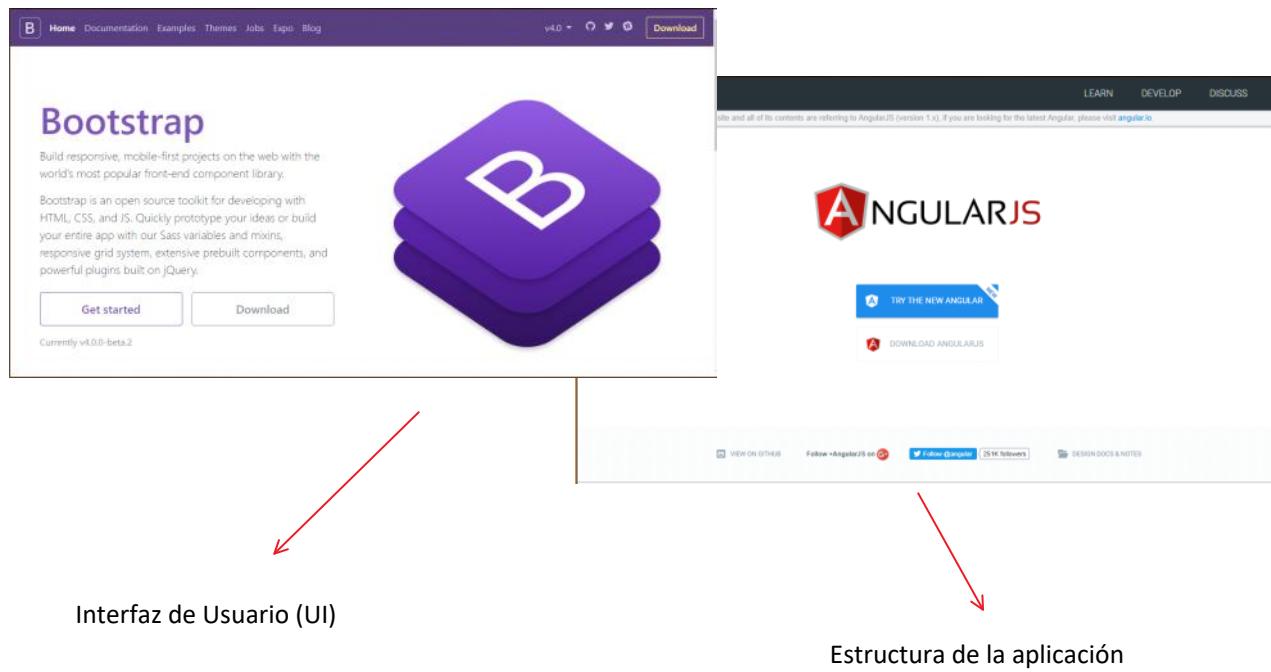
sábado, 9 de septiembre de 2017 17:13

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



Presentación

lunes, 6 de noviembre de 2017 20:40



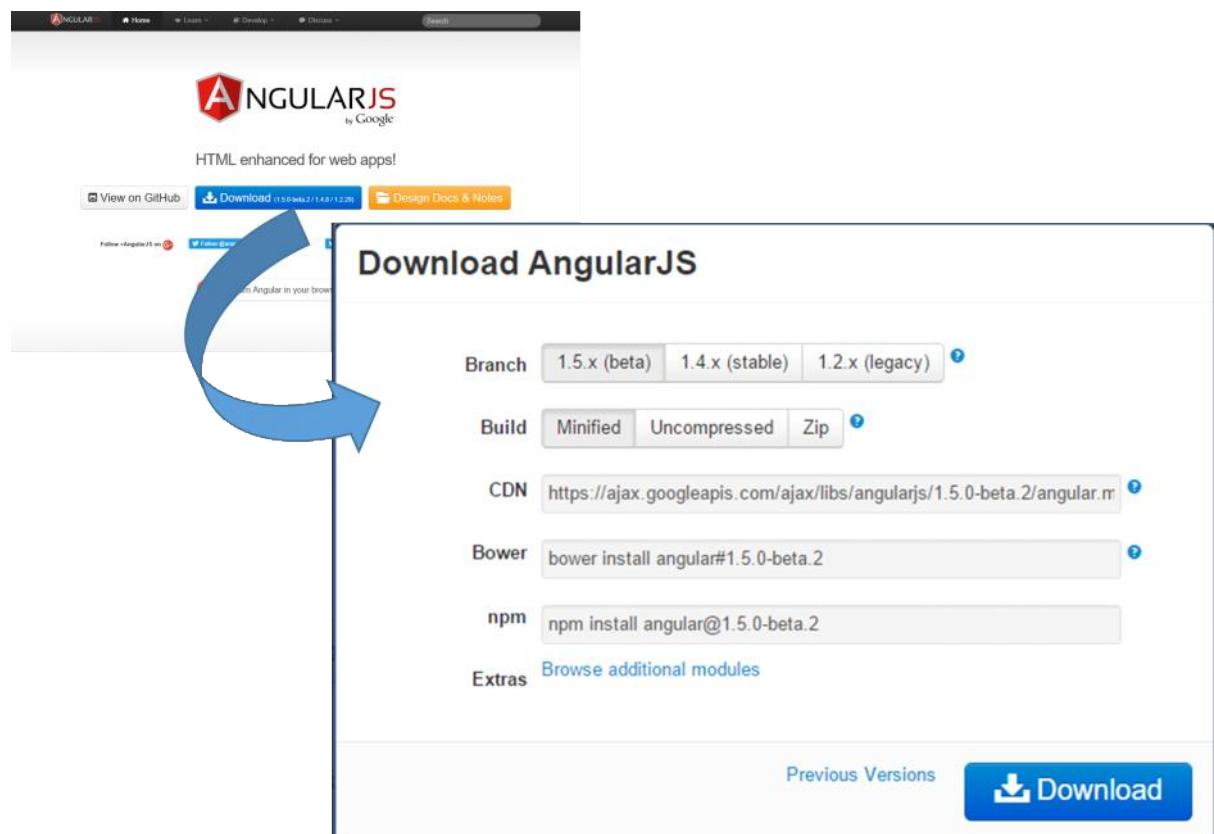
Instalación básica

domingo, 17 de septiembre de 2017 0:08

Opciones →

Descarga
Vinculación a una CDN

Angular vía CDN



Módulos de Angular

Download AngularJS

<https://code.angularjs.org/1.4.8/>

Index of /1.4.8/

File	Last Modified	Size
...	20-Nov-2013 00:13	-
angular	20-Nov-2013 00:13	3856072
angular-animate.js	20-Nov-2013 00:13	141532
angular-animate.min.js	20-Nov-2013 00:13	25900
angular-animate.min.js.map	20-Nov-2013 00:13	69733
angular-animate.tpl.js	20-Nov-2013 00:13	140915
angular-animate.tpl.js.map	20-Nov-2013 00:13	3774
angular-animate.min.js.map	20-Nov-2013 00:13	8437
angular-cookies.js	20-Nov-2013 00:13	9720
angular-cookies.min.js	20-Nov-2013 00:13	1444
angular-cookies.min.js.map	20-Nov-2013 00:13	3094
angular-loader.js	20-Nov-2013 00:13	242
angular-loader.min.js	20-Nov-2013 00:13	1488
angular-loader.min.js.map	20-Nov-2013 00:13	3774
angular-message-format.js	20-Nov-2013 00:13	77644
angular-message-format.min.js	20-Nov-2013 00:13	15080
angular-message-format.min.js.map	20-Nov-2013 00:13	28130
angular-messages.js	20-Nov-2013 00:13	29712
angular-messages.min.js	20-Nov-2013 00:13	2748
angular-messages.min.js.map	20-Nov-2013 00:13	7419
angular-resource.js	20-Nov-2013 00:13	62432
angular-resource.min.js	20-Nov-2013 00:13	27487
angular-resource.min.js.map	20-Nov-2013 00:13	3774
angular-route.js	20-Nov-2013 00:13	6071
angular-route.min.js	20-Nov-2013 00:13	35677
angular-route.min.js.map	20-Nov-2013 00:13	11132
angular-sanitize.js	20-Nov-2013 00:13	24048
angular-sanitize.min.js	20-Nov-2013 00:13	6627
angular-sanitize.min.js.map	20-Nov-2013 00:13	10981
angular-touch.js	20-Nov-2013 00:13	14337
angular-touch.min.js	20-Nov-2013 00:13	23179
angular-touch.min.js.map	20-Nov-2013 00:13	971
bootstrap.css	20-Nov-2013 00:13	3058
bootstrap.js	20-Nov-2013 00:13	1879720
bootstrap.min.css	20-Nov-2013 00:13	14457
bootstrap.min.js	20-Nov-2013 00:13	356648
bootstrap.min.js.map	20-Nov-2013 00:13	9180
bootstrap-theme.css	20-Nov-2013 00:13	203
bootstrap-theme.css.map	20-Nov-2013 00:13	1019
version.txt	20-Nov-2013 00:13	5

Branch: 1.5.x (beta) 1.4.x (stable) 1.2.x (legacy)

Build: Minified Uncompressed Zip

CDN: https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js

Bower: bower install angular#1.4.8

NPM: npm install angular@1.4.8

Extras: Browse additional modules

Previous Versions

[Download](#)

- angular-animate
- angular-aria
- angular-cookies
- angular-loader
- angular-message / message-format
- angular-resource
- angular-route
- angular-sanitize
- angular-touch

Bootstrap vía CDN

B Home Documentation Examples Themes Jobs Expo Blog v4.0 [Download](#)

Search... Download source

Getting started

- Introduction
- Download
- Contents
- Browsers & devices
- JavaScript
- Theming
- Build tools
- Webpack
- Accessibility
- Layout
- Content
- Components
- Utilities

Bootstrap CDN

Skip the download with the Bootstrap CDN to deliver cached version of Bootstrap's compiled CSS and JS to your project.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css" integrity="sha384-PsH8zvJ+P=Copy"
```

If you're using our compiled JavaScript, don't forget to include CDN versions of jQuery and Popper.js before it.

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UE=Copy"
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js" integrity="=Copy"
```

Compiled CSS and JS
Source files
Bootstrap CDN
Package managers
npm
RubyGems
Composer
NuGet

Bootstrap CDN

La red de distribución de contenidos *MaxCDN* proporciona soporte a Bootstrap, permitiendo enlazar directamente con los ficheros CSS y JavaScript.

<!--Latest compiled and minified CSS -->

```
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/.../css/bootstrap.min.css">
<!--
Latest compiled and minified JavaScript --><script
src="https://maxcdn.bootstrapcdn.com/bootstrap/.../js/bootstrap.min.js"></script>
```

Editores de código

**Editores de texto
específicos para
código**

Sublime Text



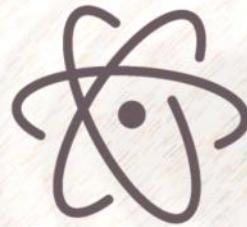
<http://www.sublimetext.com/>

Brackets



<http://brackets.io/>

Atom



<https://atom.io/>

Visual Studio Code

The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links to 'Visual Studio Code', 'Docs', 'Updates', 'Blog', 'Extensions', and 'FAQ'. A green 'Download' button is also present. Below the navigation, a message says 'Version 1.7 is now available! Read about the new features and fixes in October.' The main content area features a large blue banner with the text 'Code editing. Redefined.' and 'Free. Open source. Runs everywhere.' It includes a 'Download for Windows' button and a note about availability on other platforms. A 'By using VS Code, you agree to its license and privacy statement.' link is also visible. To the right, a window titled 'www.ts - node-express-ts - Visual Studio Code' is shown, displaying code for a Node.js application. The bottom right corner features the Microsoft logo.

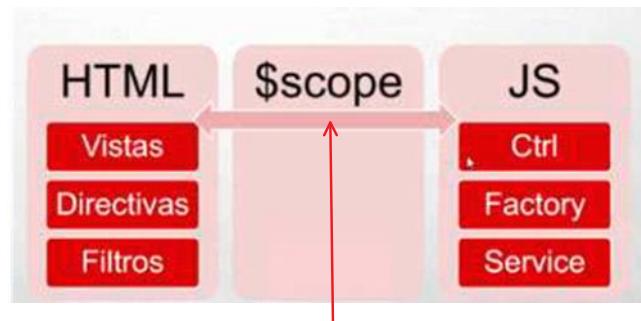
<https://code.visualstudio.com/>

Elementos de Angular JS

sábado, 9 de septiembre de 2017 17:35

En términos prácticos, se distingue

- HTML = programación declarativa
- JS = programación imperativa



El *binding* entre ambos es responsabilidad del *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 la vista, por lo que actuaría como un *view model*

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

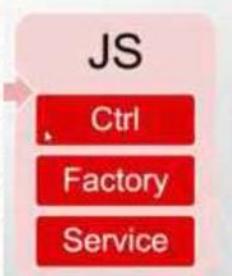
Expresiones {{}}

- 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)



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)

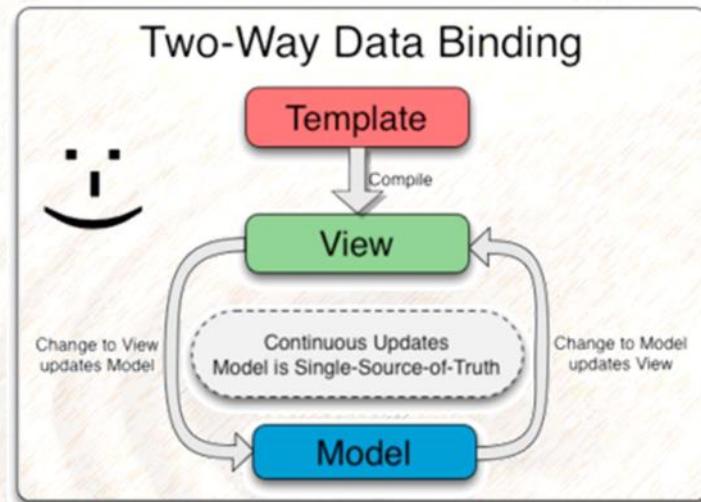


Elemento: Doble binding

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





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 atributos **ng-** específicos de angularJS que se pueden asignar a cualquier etiqueta 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 básicas

- **ng-app** inicializa la aplicación;
 define el elemento en el que se **auto ejecuta Angular**;
 - ✓ lo más común es ponerlo al principio del documento, en la etiqueta HTML o BODY
 - ✓ también se puede colocar en un área más restringida dentro del documento en otra de las etiquetas de tu página (e.g. SECTION, DIV)

- **ng-init** tareas de **inicialización** de la aplicación.
 Permite definir el modelo, creando propiedades en el **scope**, directamente en el HTML (anti-patrón)
 - ✓ El único caso apropiado donde se debería de usar ngInit es en enlace de propiedades especiales de **ngRepeat**.

Hola Mundo

Ejemplo de código AngularJS **puramente declarativo** con una funcionalidad muy básica:
la vista incorpora implícitamente el modelo y el controlador,

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Hola Mundo</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="../lib/angular/angular.js"></script>
</head>
<body ng-app ng-init="name='Pepe';apellido='Perez'">
    <form>
        <p>
            <label for="name">Nombre</label>
            <input id="name" type="text" ng-model="name">
        </p>
        <p>
            <label for="apellido">Apellido</label>
            <input id="apellido" type="text" ng-model="apellido">
        </p>
        <p>
            <button ng-click="name=''; apellido=''">Borrar</button>
        </p>
    </form>
    <p>Hola {{name.toUpperCase()}} {{apellido.toUpperCase()}}</p>
</body>
</html>
```

script con el código de AngularJS

Carga de AngularJS

Inicialización valores en \$scope para la definición del modelo

Vinculación de elementos del DOM a propiedades del modelo

Cambios en el modelo en respuesta a un evento (clic)

Acceso desde el DOM a propiedades del modelo

DOBLE BINDING

Hola Mundo MVC

Ejemplo de código AngularJS con la misma funcionalidad básica repartida entre elementos **declarativos** (la vista HTML) y elementos **imperativos** (el modelo y el controlador definidos en el fichero JS)

Vista HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Hola Mundo</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="../lib/angular.js"></script>
    <script src="/app.js"></script>
</head>
<body ng-app="appMain" ng-controller="AppController">
    <form>
        <p>
            <label for="name">Nombre</label>
            <input id="name" type="text" ng-model="user.name">
        </p>
        <p>
            <label for="apellido">Apellido</label>
            <input id="apellido" type="text" ng-model="user.apellido">
        </p>
        <p>
            <button ng-click="borrar()">Borrar</button>
        </p>
    </form>
    <p>Hola {{user.name.toUpperCase()}} {{user.apellido.toUpperCase()}}</p>
</body>
</html>
```

script con el código de AngularJS

script con el código específico de la aplicación

Carga de AngularJS definiendo módulo y controlador principal

Vinculación de elementos del DOM a propiedades del modelo

Función del controlador manejadora de un evento (clic)

Acceso desde el DOM a propiedades del modelo

- **ng-app** : define el módulo de Angular que se instanciará como inicio de la aplicación
- **ng-controller** : define cual será la función constructora encargada de instanciar el \$scope
- **ng-model** : define aquellos elementos dinámicos (entradas de datos) asociados a propiedades del modelo, i.e. de \$scope
- **ng-click** : permite declarar un método del \$scope que actuará como un determinado manejador de evento
- las expresiones **{}{}** permiten hacer uso de los valores del modelo como parte de la salida de datos en cualquier punto de la vista

Controlador y modelo (JS)

```
Instanciación del módulo principal → angular.module('appMain', [])
Propiedades del modelo declaradas en el scope ↗
Métodos del controlador declaradas en el scope ↗
                .controller('AppController', ['$scope', function ($scope)
{
    $scope.user = {
        name : 'Pepe',
        apellido : 'Perez'
    }
    $scope.borrar = function () {
        $scope.user.name='';
        $scope.user.apellido=''
    }
}])
```

controlador principal 2 argumentos:

- su nombre
- array de inyección de dependencias
- nombre de los argumentos
- función anónima con dichos argumentos

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 atributos **ng-** específicos de angularJS que se pueden asignar a cualquier etiqueta 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 básicas



- **ng-app** inicializa la aplicación;
define el elemento en el que se **auto ejecuta Angular**;
 - ✓ lo más común es ponerlo al principio del documento, en la etiqueta HTML o BODY
 - ✓ también se puede colocar en un área más restringida dentro del documento en otra de las etiquetas de tu página (e.g. SECTION, DIV)

- **ng-init** tareas de **inicialización** de la aplicación.
Permite definir el modelo, creando propiedades en el **scope**, directamente en el HTML (anti-patrón)
 - ✓ El único caso apropiado donde se debería de usar ngInit es en enlace de propiedades especiales de **ngRepeat**.



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](#) This repository Search Explore Features Enterprise Pricing Sign up Sign in

johnpapa / angular-styleguide Watch 1,096 Star 15,272 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/bk/turkish	Latest commit 58599ad4 4 days ago
assets description changed from "Angular controller" to "Angular directive"	2 months ago
!1Bn 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.

- Uso del formato "*controller as*" unido al empleo de una variable dentro del controlador, var vm = this, para evitar el uso de *this*
 - 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 de \$inject para definir los nombres de los elementos que se inyectan, de cara a evitar los problemas en el caos de la minimificación
 - Extrema modularización

Arquitectura de un *controller* tal como lo proporciona el *snippet* de la extensión de **John Papa** para VSC

Closures con el patrón de auto ejecución IIFE

```
(function() {
    'use strict';
    angular
        .module('Module')
        .controller('ControllerController', ControllerController);

    ControllerController.$inject = ['dependency1'];
    function ControllerController(dependency1) {
        var vm = this;
        activate();
        ///////////////
        function activate() { }

    }
});
```

Referencia a los módulos sin mapearlos

Uso de \$inject

Función con nombre

variable dentro del controlador que referencia a *this*, como complemento al uso del formato "controller as" en la vista (HTML)

Hola Mundo en Angular 1.2

sábado, 16 de septiembre de 2017 20:31

Hola Mundo "as"

Modo "*controller as*" que define el acceso al *scope* mediante una variable asignada en la vista, tal como recomienda la guía de estilo "oficial" creada por **John Papa**.

Vista HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Hola Mundo</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="../lib/angular.js"></script>
    <script src="./app.js"></script>
</head>
<body ng-app="appMain" ng-controller="AppController as vm">
    <form>
        <p>
            <label for="name">Nombre</label>
            <input id="name" type="text" ng-model="vm.user.name">
        </p>
        <p>
            <label for="apellido">Apellido</label>
            <input id="apellido" type="text" ng-model="vm.user.apellido">
        </p>
        <p>
            <button ng-click="vm.borrar()">Borrar</button>
        </p>
    </form>
    <p>Hola {{vm.user.name.toUpperCase()}} {{vm.user.apellido.toUpperCase()}}</p>
</body>
</html>
```

Acceso al *scope* mediante una variable definida en la vista

Todos los accesos al *scope* (modelo y funciones del controlador) se realizan mediante la variable definida

Controlador y modelo (JS)

```
Propiedades del modelo → angular.module('appMain', [])
                           .controller('AppController', function () {
                               this.user = {
                                   name : 'Pepe',
                                   apellido : 'Perez'
                               }
                               this.borrar = function () {
                                   this.user.name='';
                                   this.user.apellido='';
                               }
                           })
Métodos del controlador →
```

Los elementos del *scope* se declaran directamente como propiedades de la función constructora que constituye el *controller*

Siguiendo las recomendaciones de la guía de estilo de **John Papa**, se pueden utilizar funciones con nombre en lugar de anónimas.

```
angular.module('appMain', [])
.controller('AppController', AppController)
function AppController () {
    this.user = {
        name : 'Pepe',
        apellido : 'Perez'
    }
    this.borrar = function () {
        this.user.name='';
        this.user.apellido='';
    }
}
```

Controller declarado como una función con nombre

La propiedad del *hoisting* (alzamiento) permite declarar la función después de que haya sido utilizada.

Siguiendo estrictamente dicha guía el código quedaría como sigue

```
(function() {
    'use strict';
    angular
        .module('appMain', [])
        .controller('AppController', AppController);

    ControllerController.$inject = ['dependency1'];
    function AppController(dependency1) {
```

```
var vm = this;  
vm.user = {  
    name : 'Pepe',  
    apellido : 'Perez'  
}  
vm.borrar = function () {  
    vm.user.name='';  
    vm.user.apellido='';  
}  
}();
```

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 // ...
2 class SampleController {
3
4   /** ...
5    * @ngInject;
6    */
7   constructor($scope) {
8     'ngInject';
9     this.$scope = $scope;
10   }
11
12   /** ...
13    * @ngInject;
14    */
15   $onInit () {
16     this.name = "Sample"
17     this.value = parent.value
18   }
19   // Fin del $onInit
20
21 } // Fin del controller SampleController
22
23 angular.module('moduleName')
24
25 /**
26 */
27 .component("sample", {
28   require: {parent: '^appMain'},
29   templateUrl: 'components/sample.html',
30   // usa controller as por defecto
31   controller: SampleController,
32   //controllerAs: '$ctrl', valor por defecto
33   bindings: {}
34
35 })
36 //Fin del componente y del objeto que lo define
37
38
39
40
41
42
43
44
45
46
47
48
49 })
```

Guía de estilo 1.5



Personal Open source Business Explore Pricing Blog Support This repository Search Sign in Sign up

toddmotto / angular-styleguide

Issues 4 Pull requests 0 Projects 0 Pulse Graphs

Styleguide for teams <https://ultimateangular.com>

181 commits 3 branches 0 releases 40 contributors

laskoviy mishka committed with toddmotto feat(ts): Initial typescript version of guide. (#127) 8 days ago

i18n Adding Italian translation (#138) 11 days ago

typescript feat(ts): Initial typescript version of guide. (#127) 8 days ago

README.md Update README.md (#137) 21 days ago

README.md

<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.

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:

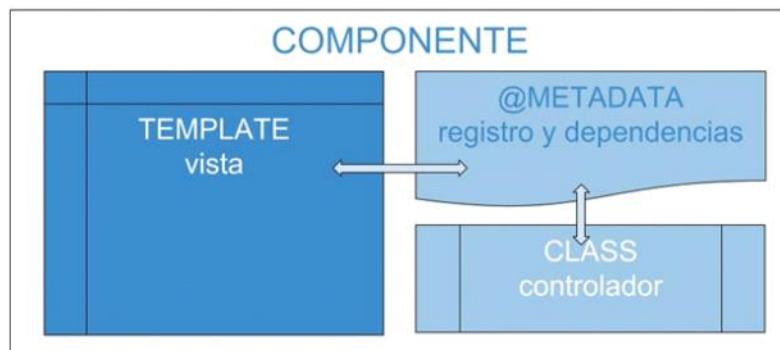
- **Custom elements:** Nos permite definir nuevos elementos HTML.
- **Templates:** Sistema de plantillas nativas en el navegador.
- **Shadow DOM:** DOM scope.
- **HTML Imports:** Carga de documentos HTML.

Continuando con Web Components

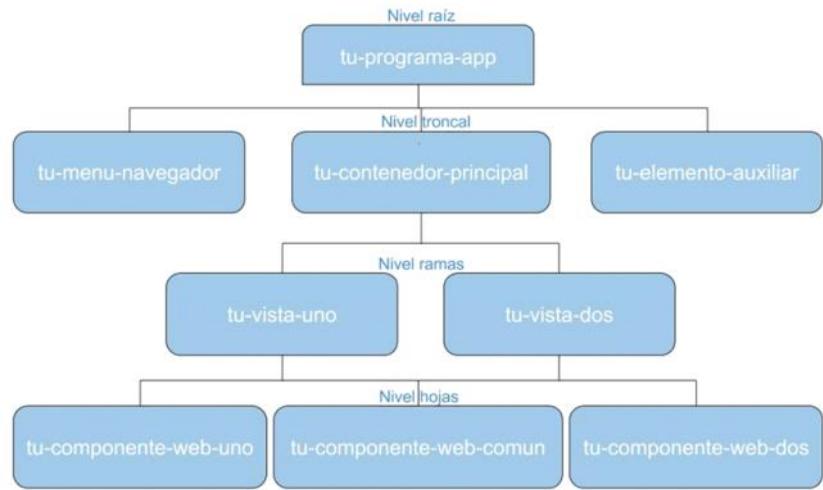


Componentes en Angular 2

- **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



Últimos Hola Mundos

sábado, 16 de septiembre de 2017 21:25

Hola Mundo ES6

Vista HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Hola Mundo</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="../lib/angular.js"></script>
    <script src=".appModule.js"></script>
    <script src=".appController.js"></script>
</head>
<body ng-app="appMain" ng-controller="AppController as $ctrl">
    <form>
        <p>
            <label for="name">Nombre</label>
            <input id="name" type="text" ng-model="$ctrl.user.name">
        </p>
        <p>
            <label for="apellido">Apellido</label>
            <input id="apellido" type="text" ng-model="$ctrl.user.apellido">
        </p>
        <p>
            <button ng-click="$ctrl.borrar()">Borrar</button>
        </p>
    </form>
    <p>Hola {{ctrl.user.name.toUpperCase()}} {{ctrl.user.apellido.toUpperCase()}}</p>
</body>
</html>
```

Controlador y modelo (JS)

```
class AppController {
    constructor () {} ← Sólo se usa para inyección de dependencias
    $onInit () {
        this.user = {
            name : 'Pepe',
            apellido : 'Perez'
        }
    } ← Método ligado automáticamente al momento inicial del ciclo de vida de la clase
    borrar () {
        this.user.name = '';
        this.user.apellido = ''; ← Método manejador de un evento click
    }
}
angular.module('appMain')
.controller('AppController', AppController)
```

Hola Mundo ES6 por componentes

Vista HTML

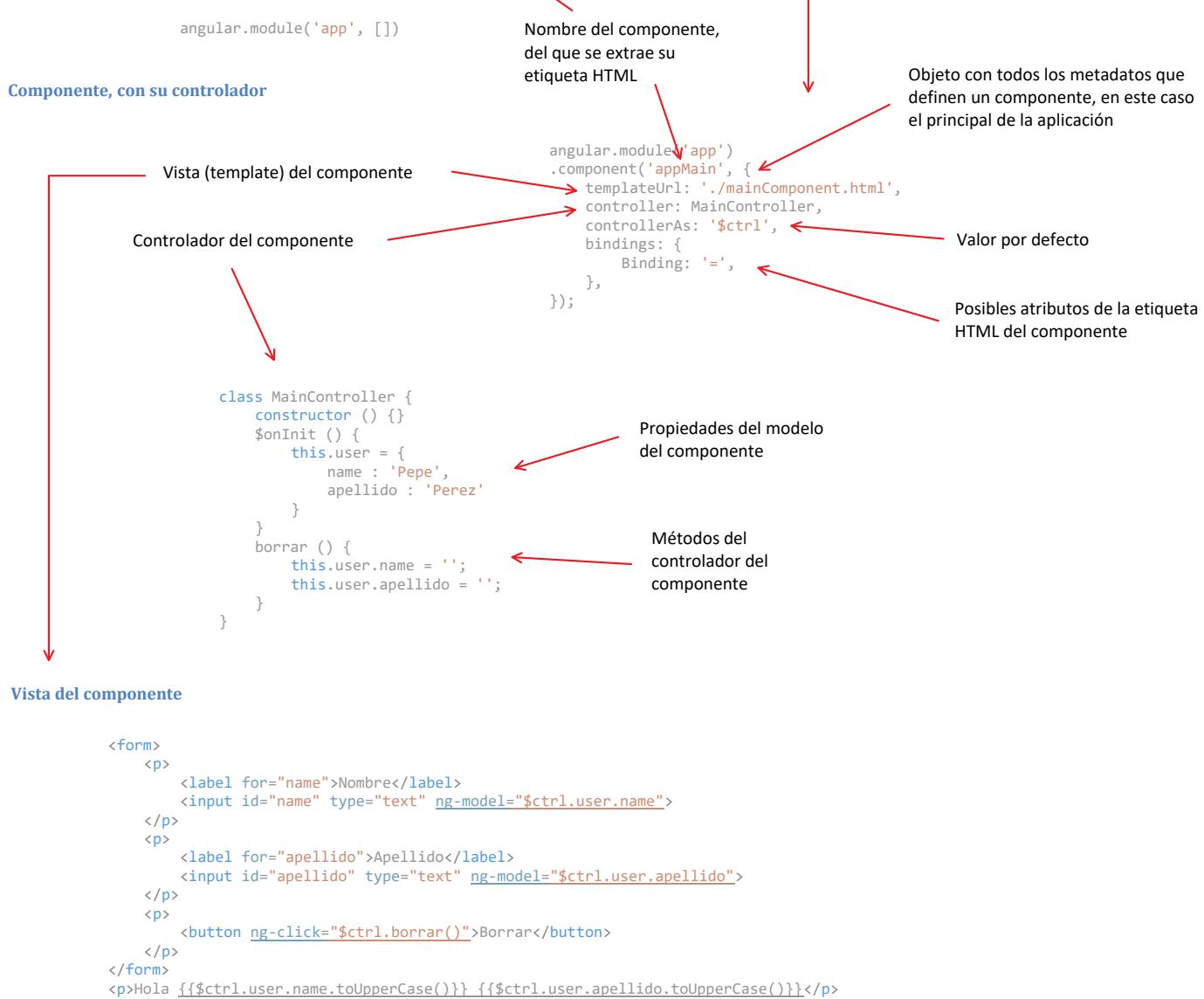
```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Hola Mundo</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <script src="../lib/angular.js"></script>
        <script src=".appModule.js"></script>
        <script src=".mainComponent.js"></script>
    </head>
    <body ng-app="app">
        <app-main></app-main>
    </body>
</html>
```

Módulo principal

```
angular.module('app', [])
```

Nombre del componente, del que se extrae su etiqueta HTML

Objeto con todos los metadatos que



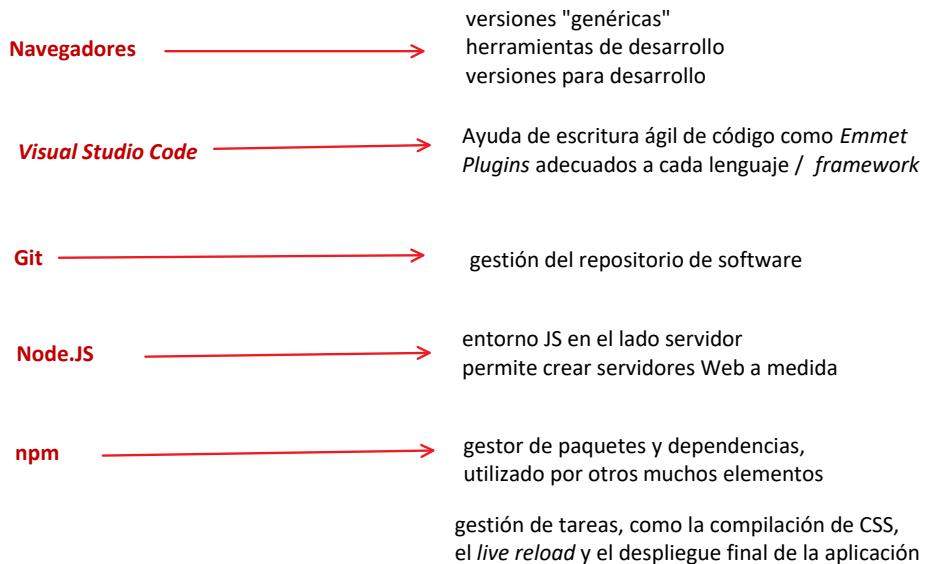
Tecnologías y Entorno de trabajo

sábado, 9 de septiembre de 2017 13:33

- Entorno de trabajo
 - Navegadores y servidor Web
 - Editores de código
 - Gestión de versiones. GIT
 - NodeJS y npm. Despliegue
- Tecnologías
 - ES6

- Editor de código
- Servidor Web
- Gestión del proyecto
 - Instalación
 - Versiones ([GIT](#))
 - Despliegue
- Entorno de pruebas
 - Tests unitarios
 - Test E2E

Entorno de trabajo



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

Inicia sesión con tu cuenta de Google para acceder a los marcadores, las contraseñas, el historial y otros ajustes desde todos tus dispositivos.

Guarda página como... Ctrl+S
Añade al escritorio...
Borrar datos de navegación... Ctrl+Mayús+S
Extensiones
Administrador de tareas Mayús+E
Herramientas para desarrolladores Ctrl+Mayús+I

Editor Cortar Copiar Pegar
Configuración Ayuda Salir

HTML CSS JS Rendimiento Memoria Red Almacenamiento Buscar

Elements Console Sources Network Performance Memory

DIRECTIVE: HTML

```
<!DOCTYPE html>
<html lang="es" dir="ltr">
  <head>...
```

html body

Console: What's New

top Filter Default levels

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



Navegador web Mozilla Firefox | Página de inicio de Mozilla | +

Tu Firefox está al día. Ahora ponte en marcha.

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

Escribe tu email

Inspector Console Depurador Editor de estilos Rendimiento Memoria Red Almacenamiento Buscar en HTML

```
<!DOCTYPE html>
<html lang="es" windows-x86_js-win7up-x64 loaded="" dir="ltr" data-latest-firefox="55.0.3" data-esr-versions="52.3.0" data-optimize-project-id="140882122" data-gtm-container-id="GTM-M2R8V" data-gtm-page-id=""/><!--firefox/whatnew--> data-stub-attribution-rate="1.0" lang="es-ES" data-gtm-page-id=""/>
```

html.windows-x86_js-win7up-x64.loaded https://firefox-mobile-downloads.mozilla.net/

Cortar Copiar Pegar Nueva ventana Nueva ventana privada Guardar página Imprimir Historial Pantalla completa Buscar Opciones Complementos Desarrollador Personalizar Desarrollador sincronizadas Conectarse a Sync

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



Sugerencias de Microsoft Edge

Tu progreso 3% completado

Sigue siendo productivo

Recibe notificaciones de tus sitios web Transmite tu contenido

Las opciones "Inspeccionar elemento" y "Ver origen" ahora se mostrarán en el menú contextual.

F12 Explorador DOM Consola 1 Depurador Red 1 Rendimiento Memoria Emulación Experimentos

HTML.js body-as-es

Ventana nueva Ventana InPrivate nueva

Zoom 100%

Transmitir contenido en un dispositivo

Buscar en la página

Imprimir

Anclar esta página a Inicio

Herramientas de desarrollo F12

Abrir con Internet Explorer

Enviar comentarios

Extensiones

Novedades y sugerencias

Configuración

background-color: #fff;

body { margin: 0; overflow-y: scroll; }

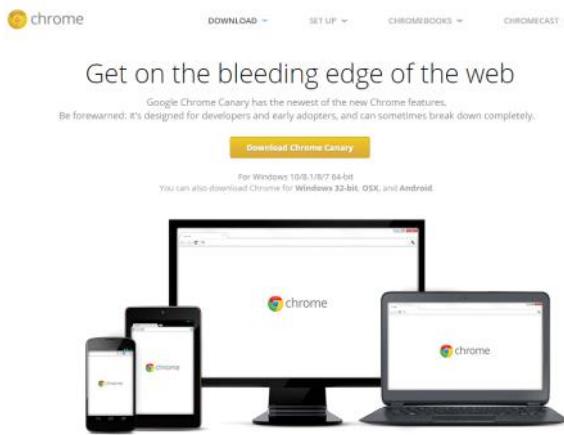
Consola JavaScript

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



The landing page for Google Chrome Canary. It features the Chrome logo and navigation links for 'DOWNLOAD', 'SET UP', 'CHROMEBOOKS', and 'CHROMECAST'. A prominent yellow button says 'Download Chrome Canary'. Below it, text reads: 'Get on the bleeding edge of the web' and '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.' It also mentions that you can download Chrome for Windows 32-bit, 64-bit, Mac, or Android. An image shows a desktop monitor, a laptop, and two mobile devices (a smartphone and a tablet) all displaying the Chrome logo.

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



The landing page for Firefox Developer Edition. It features the Firefox logo and the text 'Herramientas modernas para la Web Abierta'. Below it, it says 'Crea y depura experiencias web con poderosas herramientas de código abierto.' A blue button says 'Firefox Developer Edition'. To the right, there are three sections: 'Modernizar' (Create interfaces de usuario rápidas, flexibles e interactivas con las herramientas integradas de React y Redux), 'Personalizar' (Haz tipos innumerables de sugerencias automáticamente a Mozilla), and 'Optimizar' (Crea sitios adaptables y compatibles que funcionan para todos, en todos partes).

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

Plugin para Chrome

A screenshot of the AngularJS Batarang extension page on the Chrome Web Store. The page title is "AngularJS Batarang" with a subtitle "offered by AngularJS". It shows a 4-star rating with 1148 reviews, 293,492 users, and tabs for "OVERVIEW", "REVIEWS", and "RELATED". A large image in the center shows the extension's interface integrated into the Chrome DevTools, displaying AngularJS code and data structures. A call-to-action button "+ ADD TO CHROME" is at the top right. To the right of the main content, there's a sidebar with developer information: "Compatible with your device", "Extends the Developer Tools, adding tools for debugging and profiling AngularJS applications.", links to "Website" and "Report Abuse", and details about the extension: Version 0.10.1, Updated October 1, 2015, Size 390KIB, and Language English. A black bat silhouette is positioned to the right of the sidebar.

Batarang: ayuda en la depuración mediante Chrome de proyectos de AngularJS

Servidor Web

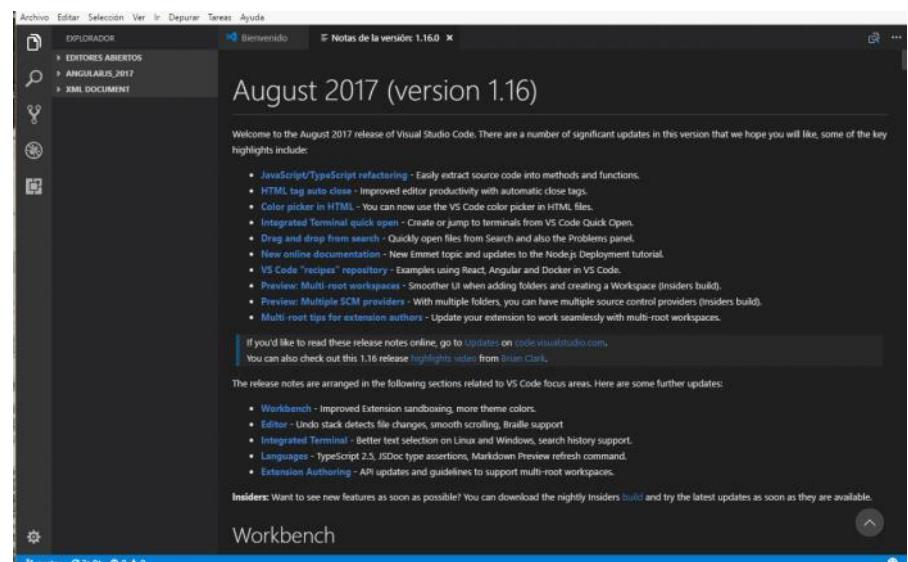
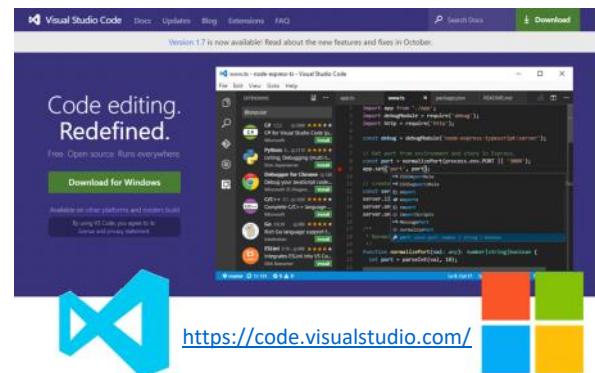
domingo, 17 de septiembre de 2017 9:44

Editores de código

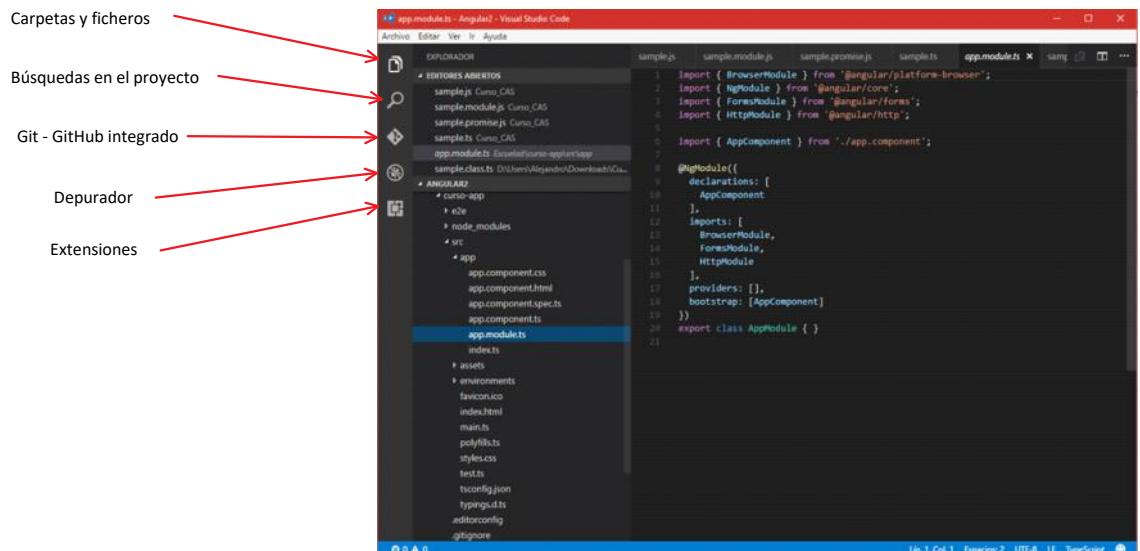
sábado, 9 de septiembre de 2017 18:21



Visual Studio Code

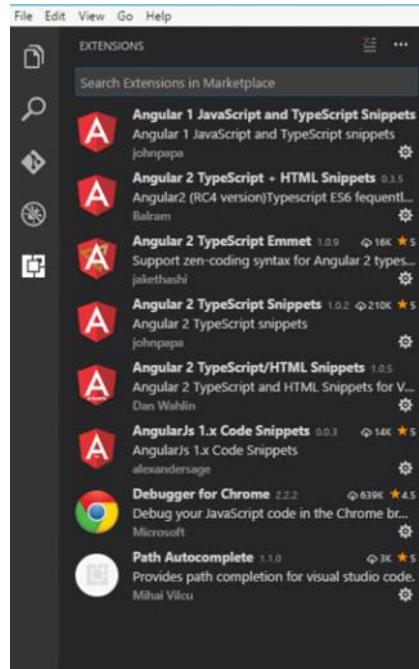


Visual Studio Code



Extensiones de VSC

- Angular 1 and TypeScript/
HTML VS Code Snippets
(Dan Wahlin)
- **Angular 1 JavaScript and Typescript
Snippets *****
(John Papa)**
- **Angular Language Service
angular2-inline
(Nate Wallace)**
- Debugger para Chrome *****
(Microsoft)
- npm ***
(egamma)
- **Path Intellisense ***
(Christian Kohler)**



Emmet

domingo, 14 de mayo de 2017 14:10

<https://emmet.io/>

- Herramienta de transformación a HTML con su sintaxis específica
- Escribe Emmet, aplicas el comando de transformación (expandir) y obtienes HTML
- Tienes que instalarse en el editor de código o IDE

The screenshot shows the official Emmet website. At the top left is the Emmet logo (a stylized 'E' icon) and the word "Emmet". Below it is the tagline "Tools for web-developers". On the left sidebar, there are links for "DOCUMENTATION", "DOWNLOAD", "CREDITS", and "BLOG". Under "More developer tools:", there are links for "Emmet LiveStyle" (Real-time bi-directional edit tool for CSS, LESS and SCSS), "Emmet Review" (Fast and easy way to test responsive design side-by-side), and "Rollbar" (Catch Code Errors Before Users Do, Track 5,000 Errors (per month) Free!). The main content area features a large heading "Emmet — the essential toolkit for web-developers" and a subtext: "Emmet is a plugin for many popular text editors which greatly improves HTML & CSS workflow:". To the right of this is a large button with a play icon and the text "Watch demo". Below this are three cards: "HTML from CSS", "Dynamic snippets", and "Ultra-fast coding".

Emmet — the essential toolkit for web-developers

Emmet is a plugin for many popular text editors which greatly improves HTML & CSS workflow:

Watch demo

HTML from CSS
Each abbreviation is transformed in runtime: just slightly change its name [to get a different result](#).

Dynamic snippets

Ultra-fast coding
With Emmet you can quickly [write a bunch of code](#), [wrap code with new tags](#), quickly [traverse](#) and [select](#) important code parts [and more!](#)

Disponibilidad

domingo, 14 de mayo de 2017 16:32

Download

 Sublime Text cross-platform	 Atom cross-platform	 Coda OS X
 Eclipse/Aptana cross-platform	 TextMate 2 OS X	 Espresso OS X
 Chocolat OS X	 Komodo Edit cross-platform	 Notepad++ Windows
 PSPad Windows	 textarea browser based, cross-platform	 CodeMirror browser based, cross-platform
 Brackets cross-platform	 NetBeans cross-platform	 Adobe Dreamweaver Windows, OS X

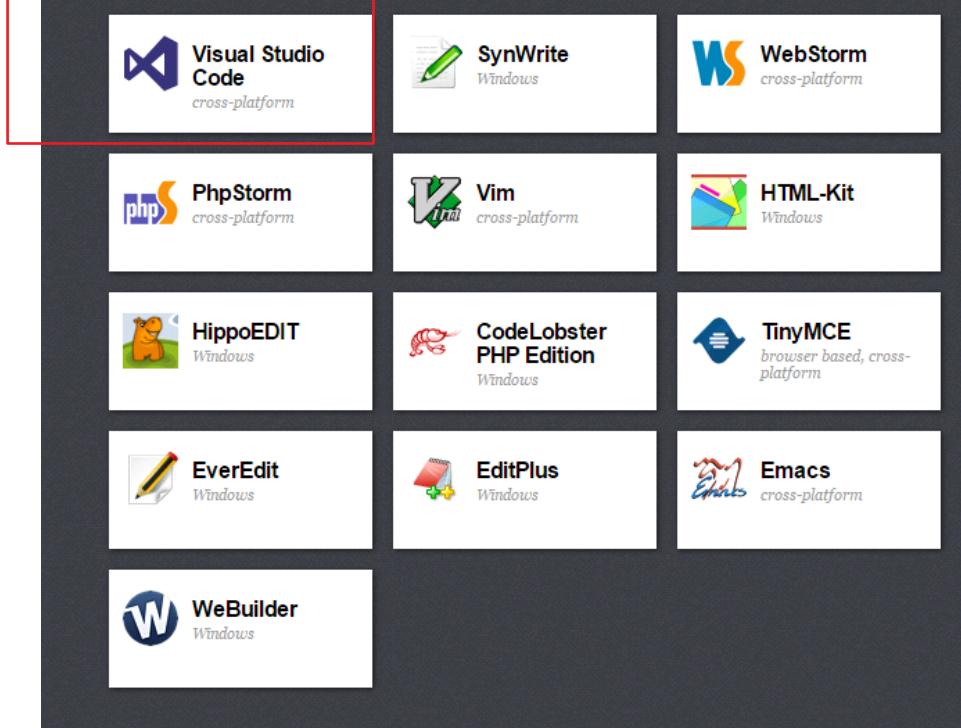
Online services

 jsfiddle the web playground	 JS Bin A JavaScript, HTML and CSS playground	 CodePen A front-end playground
 ICEcoder Online IDE	 Divshot The Interface Builder for Web Apps	 Codio Web IDE for web development, teaching and learning
 Codeanywhere Multi-platform cloud editor	 SourceLair In-browser IDE for web technologies	

Third-party support

The plugins for these editors are developed by third-party developers. May not support all Emmet features and actions.

En el caso de [Visual Studio Code](#), su propia implementación de [Emmet](#) ya está integrada en el editor, sin necesidad de añadir ningún plugin



Sintaxis

domingo, 14 de mayo de 2017 16:43

<https://docs.emmet.io/>

Muy similar a los selectores CSS

Selectores individuales

elemento
elemento.clase.clase
elemento#id
elemento[atributo atributo]

Contenido y repetición

{contenido}
* n : número de veces que se repite
\$: referencia al número de item

Selectores múltiples

Child: >
Sibling: +
agrupamientos ()



Git es un control de versiones distribuido para la gestión eficiente de flujos de trabajo distribuidos 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.
- Solaris.

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

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



Estados del archivo

Modificado Preparado Confirmado

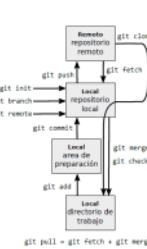
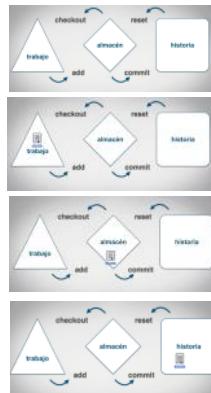
Comandos

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 -> comando para administrar repositorios remotos

git branch -> crear y listar ramas.
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 -> comando que combina git fetch y git merge para traer cambios de un repositorio local

Este comando es exactamente 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.

Ciclo de operaciones



El repositorio creado tiene tres partes principales: El directorio en lo que se llama directorio de trabajo, que es la carpeta .git que ha sido creada por el comando git init; el repositorio remoto que se encuentra en la zona 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 implementación de shell que son fácilmente encadenables para formar nuevos comandos. Los scripts tienen mecanismos para llamar scripts de usuario cuando suceden ciertos eventos en el flujo de trabajo (denominados puntos de ejecución hooks).

Resultado



Comprobación

```
git status
# On branch master
# Your branch is up-to-date with 'origin/master'.
#
```


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 of the text is a form for creating a new account, which includes fields for 'Username' (with placeholder 'Pick a username'), 'Email' (with placeholder 'you@example.com'), 'Password' (with placeholder 'Create a password' and a note 'Use at least one letter, one numeral, and seven characters'), and a green 'Sign up for GitHub' button. Below the button is a small legal notice.

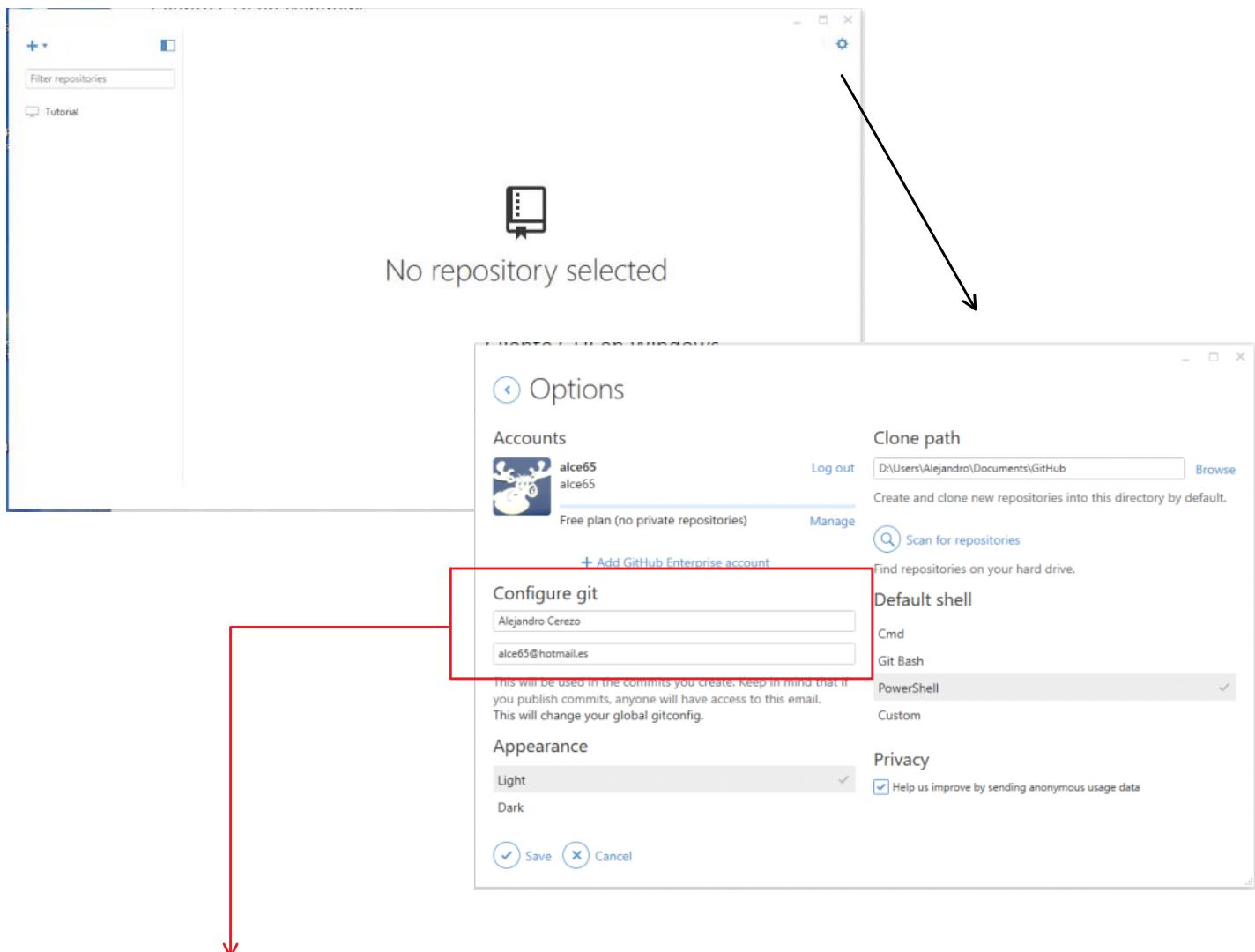
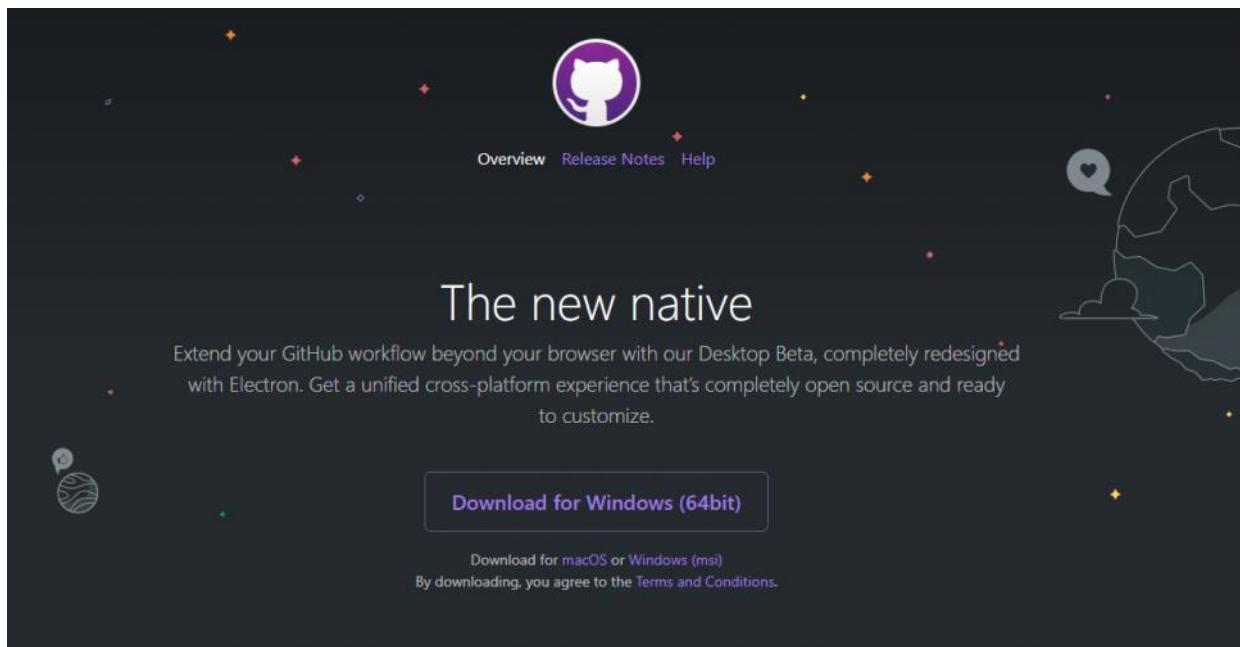
<https://github.com/alce65>

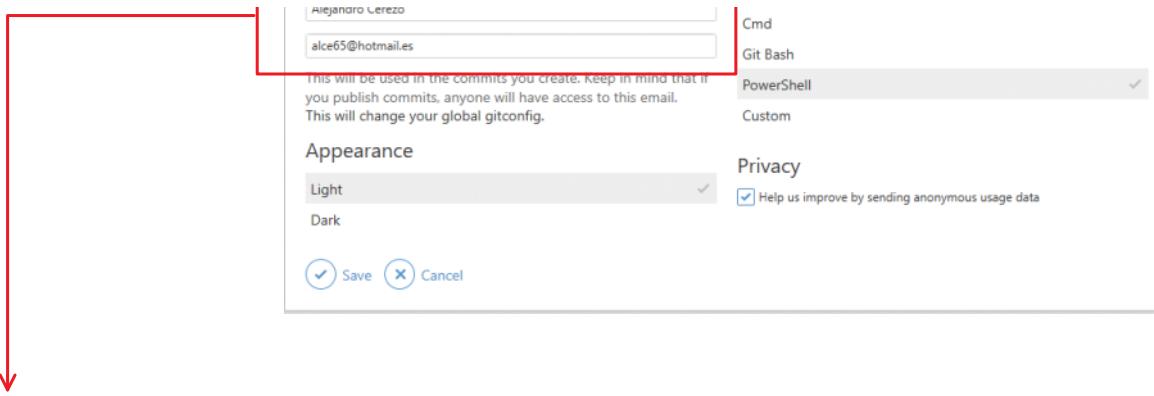
The screenshot shows the GitHub profile page for the user 'alce65'. At the top, there's a navigation bar with links for Overview, Repositories (18), Stars (2), Followers (2), and Following (6). On the left, there's a profile picture of a reindeer and the user's name 'alce65'. Below the profile picture are buttons for 'Block or report user' and a location indicator 'Madrid, Spain'. The main content area displays three repository cards: 'Curso_Angular2' (by 'Icono Training Consulting', updated 25 minutes ago), 'angularjs_2017' (by 'Tecnocom', updated on 11 Aug), and 'CursoPelayo' (by 'HTML', updated on 16 Jun).

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

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner / Repository name

Great repository names are short and memorable. Need inspiration? How about probable-spork.

Description (optional)
Curso de Angular2 en Icono Training Consulting

Public Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with a README This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Node | Add a license: MIT License | ⓘ

Create repository

This repository | Search

alce65 / Curso_Angular2

Pull requests Issues Marketplace Explore

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Curso de Angular2 en Icono Training Consulting

1 commit 1 branch 0 releases 1 contributor

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

.gitignore Initial commit just now

LICENSE Initial commit just now

README.md Initial commit just now

README

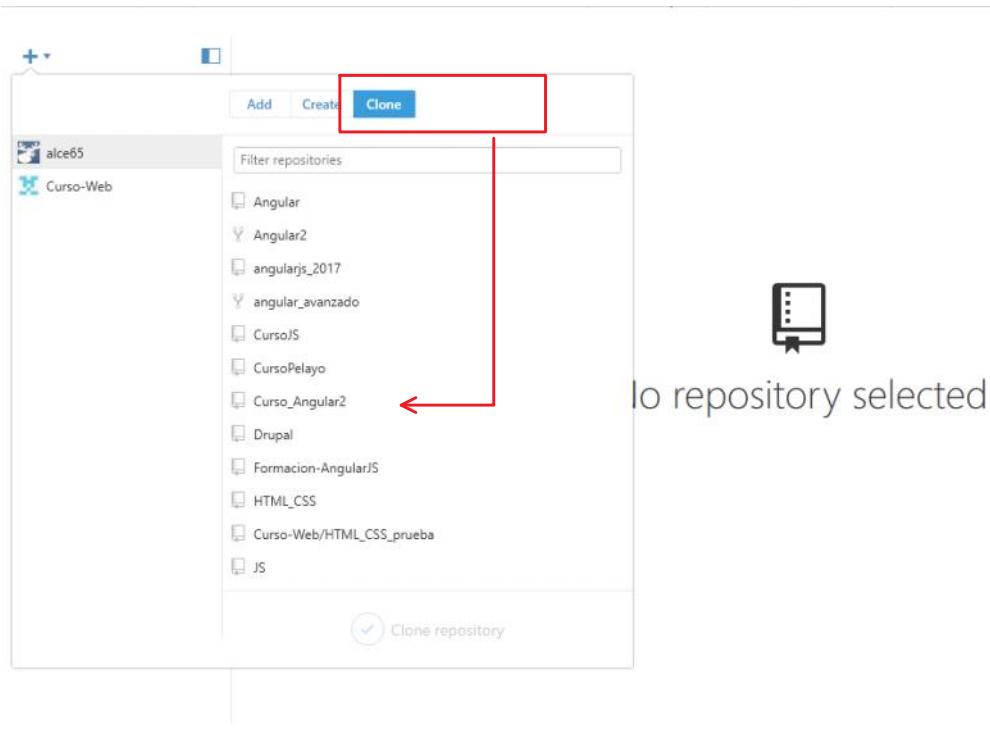
Curso_Angular2

Curso de Angular2 en Icono Training Consulting

Clonación local del repositorio

Clonar un repositorio

domingo, 10 de septiembre de 2017 17:22



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

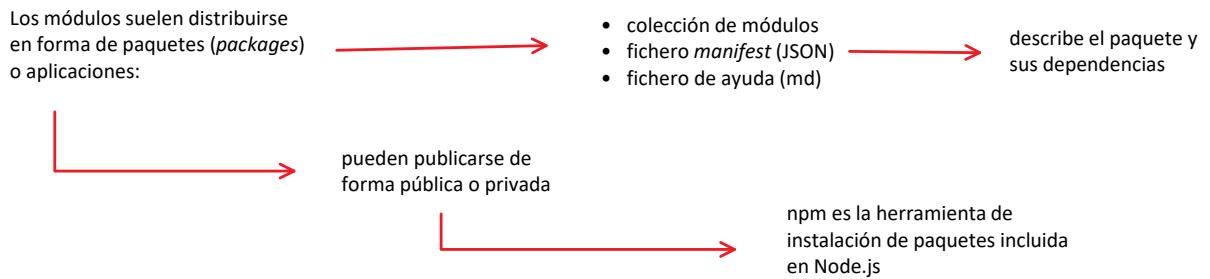
Objetivo del proyecto

→ 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

04 Modulo Web Server

```
▶ 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

v8.4.0 Current
Latest Features

node-v8.4.0-x64.msi

Node.js Setup
Installation Folder
Welcome to the Node.js Setup Wizard
The Setup Wizard will install Node.js on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Install Node.js to:
Program Files\nodejs
Change...
• node
• npm

Node.js command prompt
Node.js documentation
Node.js website
Node.js
Uninstall Node.js

Node.js command prompt
Your environment has been set up for using Node.js 8.1.4 (x64) and npm.
C:\Users\alce6>

Símbolo del sistema
C:\Users\alce6>node -v
v8.1.4
C:\Users\alce6>npm -v
5.0.3

Node.js command prompt - node
D:\Users\Alejandro>node
> var a = 12;
undefined
> var b = 3;
undefined
> console.log(a+b);
36
undefined
>

Salida ctrl-C o 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

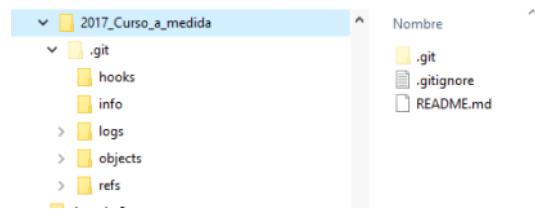
El mismo comando node <fichero.js>, permite la ejecución de un fichero

Ejemplo: Inicio del proyecto

domingo, 30 de julio de 2017 20:44

- Instalación: NodeJS + npm
- Versiones: [GIT](#)
- Despliegue: (Grunt, Gulp) npm

Carpeta raíz del proyecto / curso después de instalar GIT



Uso de npm para crear el fichero de configuración package.json

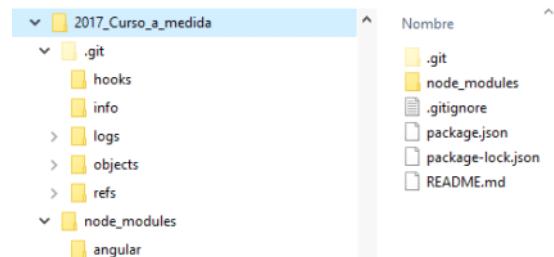
```
Selezionar Administrador: C:\Windows\system32\cmd.exe
D:\Desarrollo\Angular\2017_Curso_a_medida>npm init
{
  "name": "2017_curso_a_medida",
  "version": "1.0.0",
  "description": "Curso de AngularJS",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "Alejandro Cerezo",
  "license": "ISC"
}

Is this ok? (yes)
```

Uso de npm para instalar Angular

```
D:\Desarrollo\Angular\2017_Curso_a_medida>npm install -s angular
+ angular@1.6.5
added 1 package in 2.568s
D:\Desarrollo\Angular\2017_Curso_a_medida>
```

Resultado: node_modules incluye angular junto con cualquier otro elemento que sea necesario instalar



Configuración de proxies

lunes, 7 de agosto de 2017 21:38

Configuración git

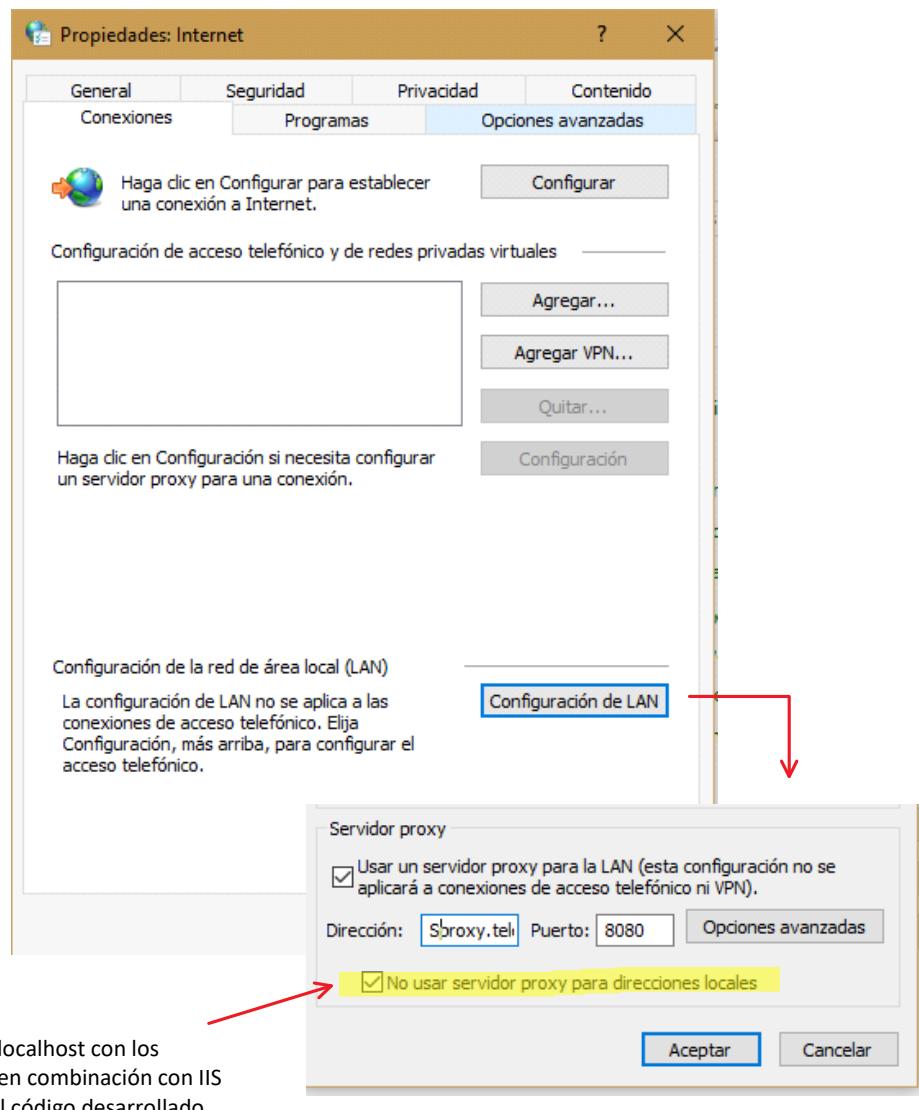
```
git config --global http.proxy http://user:passw@proxy.empresas.es:8080
```

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



Permite usar localhost con los navegadores en combinación con IIS para probar el código desarrollado

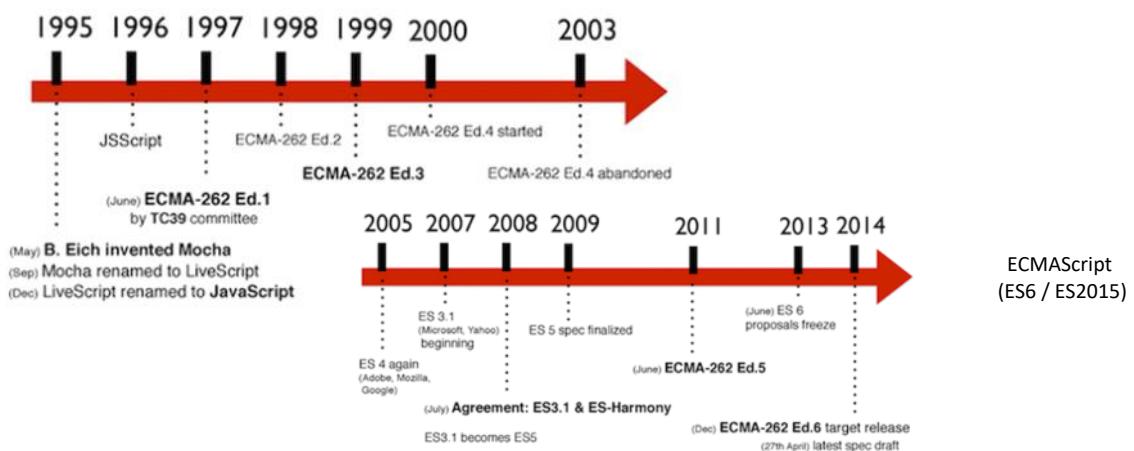
ECMAScript 6 (ES6 / ES2015)

sábado, 9 de septiembre de 2017 13:33

Brendan Eich
Netscape



European Computer Manufacturers' Association



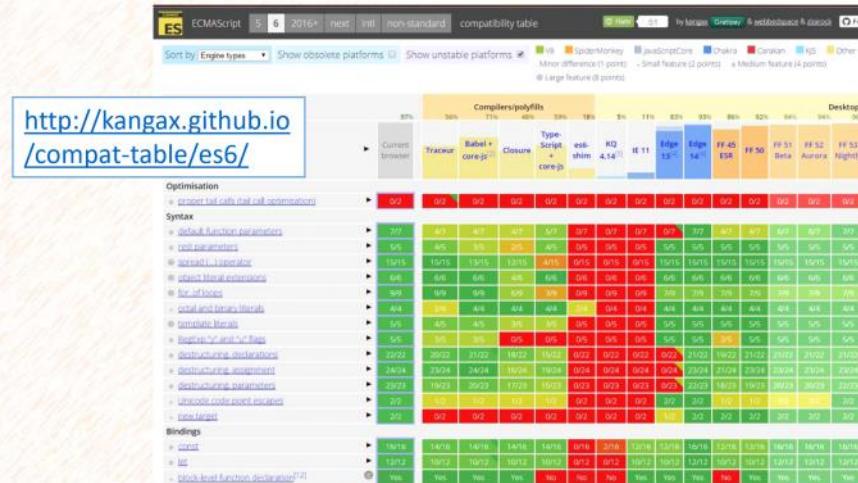
- Constates (`const`). Variables con ámbito (`let`)
- Función Arrow. This "semántico"
- *Template Strings*: interpolación de variables
- Valores por defecto
- Clases (`class`)
- Módulos (`export / import`)
- Promesas (`promise`)
- *Destructuring* ...

Más información

JS

ECMAScript 6 — New Features: Overview & Comparison

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



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 ES5 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

```
// Ejemplo de código en ES6
Clase "padre" → class Libro {}

Clase que hereda de → class LibroTecnico extends Libro {
    la anterior

Constructor → constructor(tematica, paginas) {
    super(tematica, paginas);
    this.capitulos = [];
    this.precio = "";
    // ...

Método que define → metodo(pValor = "foo") {
    valores por defecto
    // ...
}
```

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(callback)

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)) ← Utilización de las promesas, encadenando las llamadas a ellas
  .then((msg) => {
    console.log(`done after 300ms:${msg}`)
  })
```

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

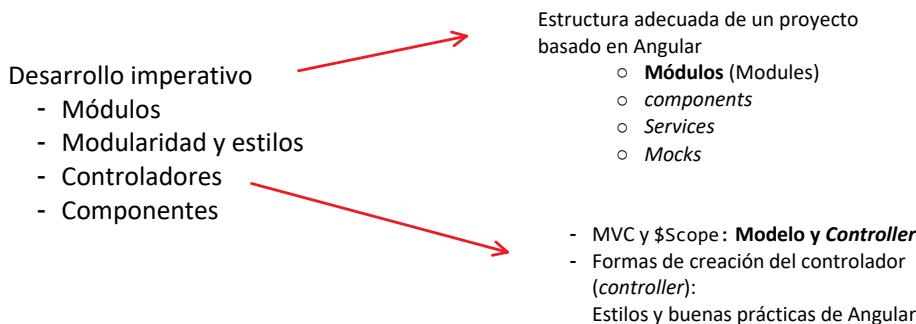
En este caso, la promesa siempre se resuelve correctamente, creando un mensaje de saludo a un usuario

```
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

Ξ Arquitectura de aplicaciones

sábado, 16 de septiembre de 2017 20:30



Arquitectura de proyectos en AngularJS (*Scaffolding*).

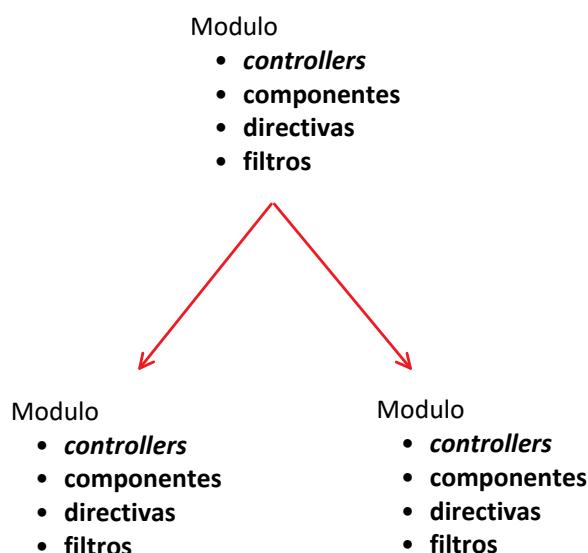
Desarrollo imperativo: JavaScript

La forma de definir las funcionalidades mediante lenguajes de programación "tradicionales", en este caso **JS**, es lo que se denomina desarrollo imperativo o por procedimientos.

Todos los elementos de JS deben incluirse en un bloque <script>, que podría incluir el código en el propio fichero HTM, pero que en la práctica será siempre una referencia a un fichero externo.

Con independencia de su distribución en uno o varios ficheros, el código JS solo es accesible para Angular si pertenece a un **módulo** referenciado en la estructura **jerárquica de módulos** de la aplicación

Por tanto los **módulos** van a ser el elemento utilizado en la organización de todo el código JS. En ellos tendrán cabida cada uno de los demás elementos que componen la aplicación, comenzando por **controllers** y **componentes**, como piezas centrales del desarrollo imperativo es el :



Módulos

sábado, 16 de septiembre de 2017 23:50

contenedor donde se sitúa el código de los controladores, componentes, directivas, etc., de forma que queda aislado, evitando colisiones con nombres repetidos en otras partes del código:

- aportan un espacio de nombres
- permiten que el código sea más reutilizable.

Evolución

En JS, la implementación inicial de módulos fue definida en *CommonJS*.
Posteriormente ha sido reinterpretada en entornos como Node.js o AngularJS.
Finalmente gracias a los métodos *export / import* se ha incorporado al estándar de ES6, aunque aún no ha sido implementado por los navegadores.

Creación de módulos en AngularJS

Utilizamos el método *module()* del objeto global angular, creado al cargar AngularJS mediante la directiva *ngApp* permite instanciar el objeto correspondiente al módulo

HTML

```
<body ng-app="appMain"> ← index.html carga angular con una referencia a cuál  
será el módulo principal de la aplicación
```

Código JS

```
var oModulo = angular.module('miAplicacion',  
    [ ... ],  
    function(...){ ... }) ← 3 argumentos:  
    ↑  
    • nombre de la aplicación  
    • [ inyección de dependencias, e.g. otros módulos ]  
    • función anónima opcional que configura el módulo)
```

En lugar de "cachearlo" como una variable, el acceso posterior al módulo se puede realizar con el mismo método que lo crea pasándole como único argumento el nombre del módulo

Instanciación

```
angular.module('miAplicacion', [ ... ], function(...){ ... }) ← El método module utiliza un patrón  
factoría (factory), y nos devuelve  
una instancia de la "clase" module  
(Objeto module)
```

Posterior uso

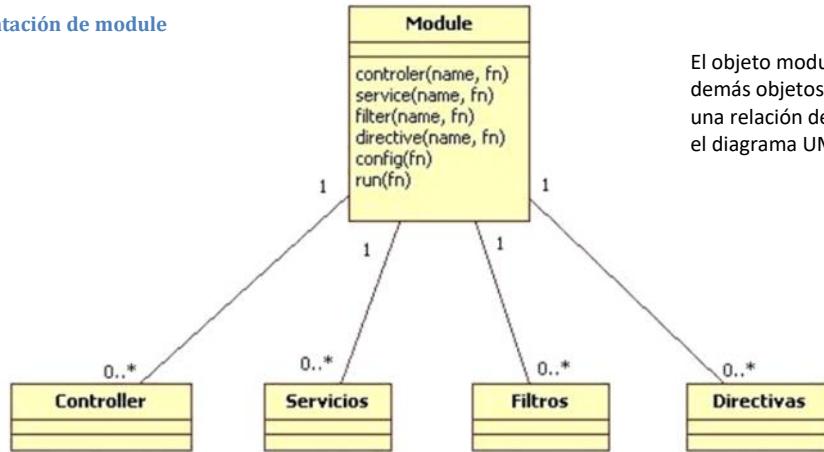
```
angular.module('miAplicacion').  
metodo(...).  
metodo() ← La sintaxis denominada estilo "Fluent"  
concatena tras la instanciación los  
métodos invocados del objeto
```

El objeto module tiene una serie de métodos que permiten controlar la lógica de la aplicación

Estos métodos devuelven al propio objeto que los invoca, con lo que pueden encadenarse sucesivamente

- *config*
- *run*
- *controller*
- *component*
- *directive*
- *constant*
- *value*
- *service*
- *factory*
- *provider*

Representación de module



El objeto module instancia como atributos todos los demás objetos de la aplicación, creando con ellos una relación de asociación, tal como se muestra en el diagrama UML

Módulos y ficheros

Esto permite distribuir el código JS de un módulo en tantos ficheros como sea necesario

Fichero del módulo:

- nombre.module.js

```
angular.module('miAplicacion', [ ... ], function(...){ ... })
```

Otros ficheros:

- nombre.controller.js
- nombre.componente.js
- nombre.servicio.js ...

```
angular.module('miAplicacion')
```



Ficheros y módulos

En cualquiera de las distribuciones, hay que diferenciar 2 procesos

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

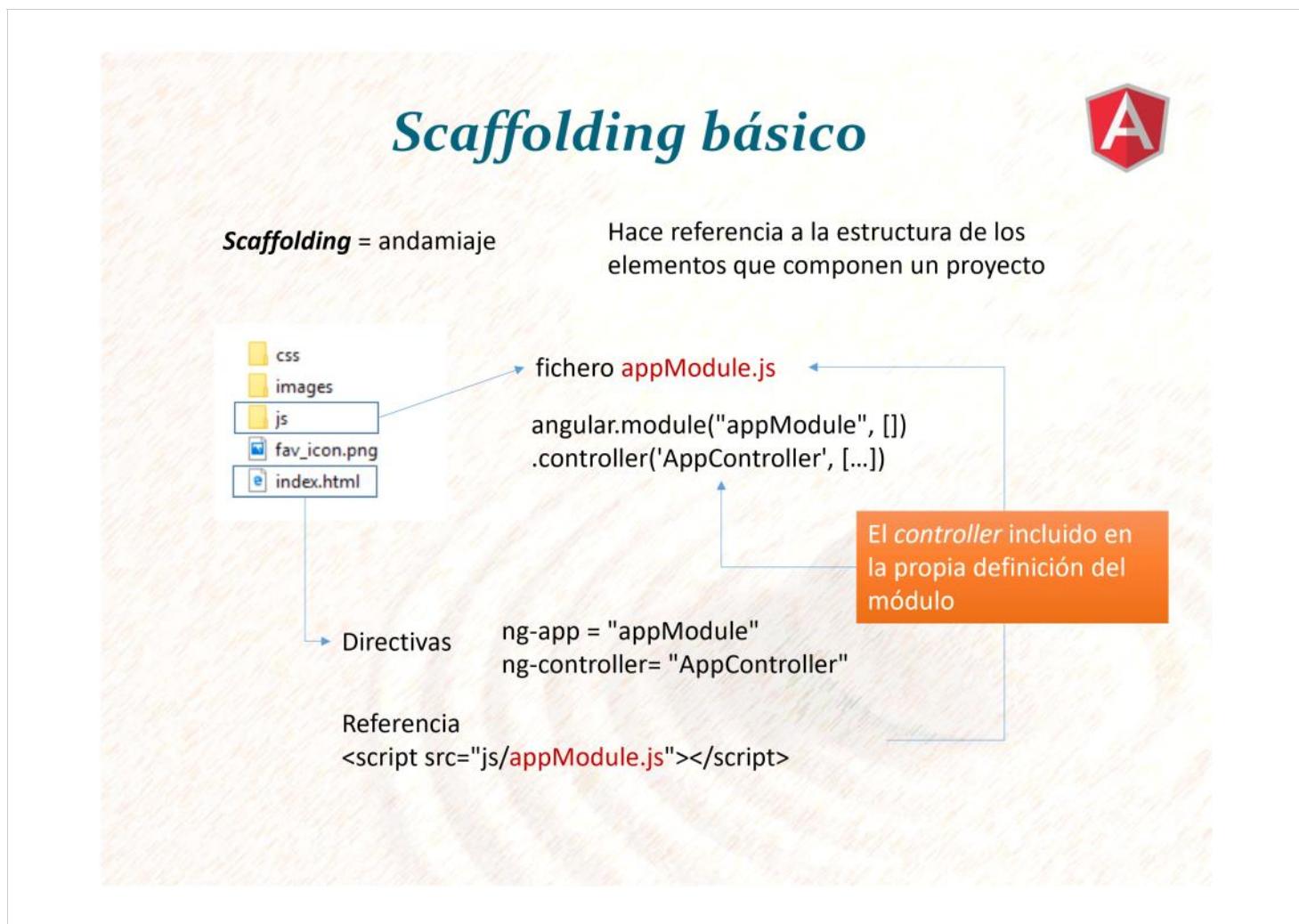
index.html → ng-app='mainApp'

app.js →

```
angular.module('mainApp', ['loginModule','commentModule']);
```

loginModule.js →

```
angular.module('loginModule', ['mainApp.login.controllers', 'mainApp.login.directives']);
```



Ejemplo

Acumulador

Control de operación:

Incremento:

+ -

Totales:

En el acumulador llevamos 10

Alejandro L. Cerezo - Madrid 2017

- Acumulador_Modulos
- css
- # default.css
- js
- JS acumulador.controller.js
- JS app.js
- fav_icon.png
- index.html

Modularidad



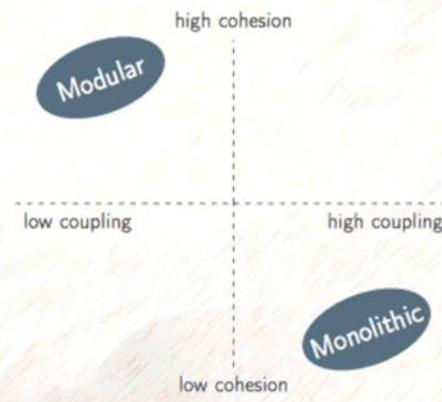
cohesión es la medida de lo que los módulos hacen

acoplamiento es la medida de la dependencia entre los distintos módulos

En Angular

```
angular.module("appModule", [...])
```

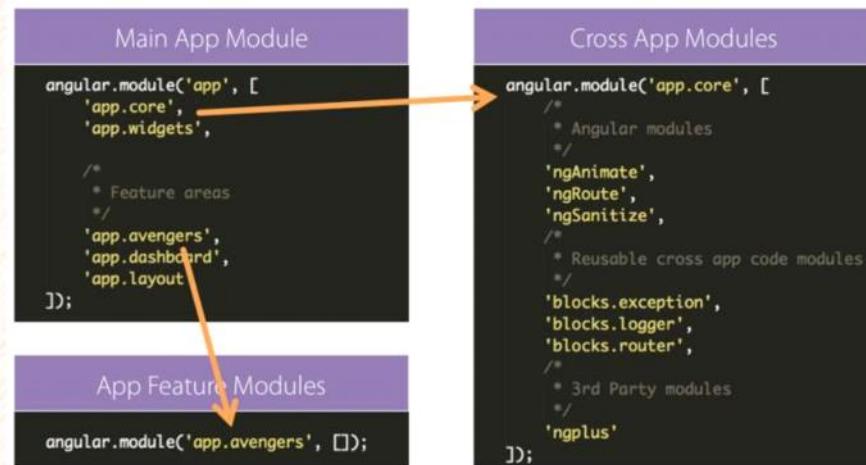
Reflejo en la guía de estilo de John Papa



Modularity and AngularJS -
João Figueiredo

<https://medium.com/@lucalanca/modularity-and-angularjs-454e457c6df#.3ksndvsmx>

Módulos según John Papa



Módulo APP [Style Y161]

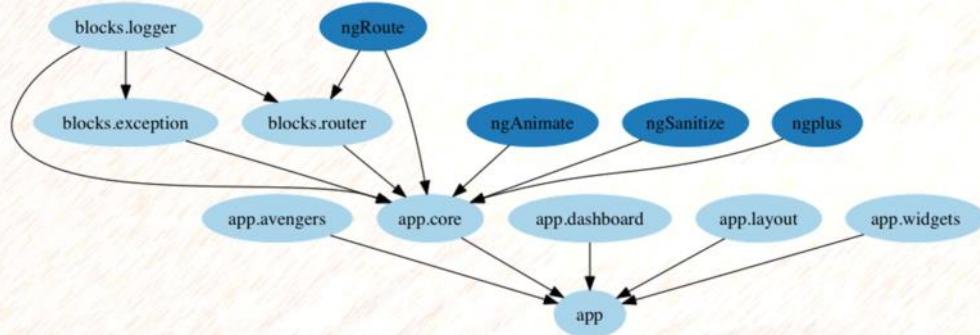
Módulos de "características" [Style Y163]

Módulos reutilizables [Style Y164]

Nomenclatura

Scaffolding elegido

Diagramas de módulos



A screenshot of a GitHub repository page for 'lucalanca/grunt-angular-architecture-graph'. The repository has 93 commits, 2 branches, 6 releases, and 5 contributors. It was last updated on April 10, 2015. The page shows options to 'New pull request', 'Find file', and 'Download ZIP'. The URL of the repository is <https://github.com/lucalanca/grunt-angular-architecture-graph/>.

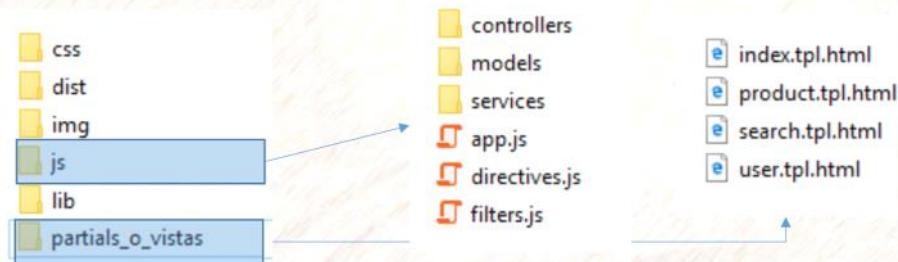
Estructura de carpetas

domingo, 17 de septiembre de 2017 0:53



Scaffolding por capas

Distribución por capas



assets = css + img

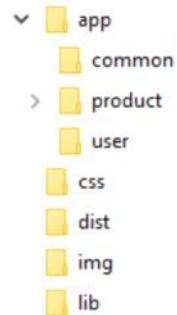
dist: imprescindible si hay procesos automatizados (grunt)

partials o views: se recomienda alguna nomenclatura para distinguir con precisión las vistas y la ruta a la que corresponden

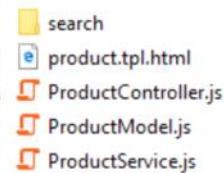
Scaffolding por características



Distribución por características



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



Cada bloque incluye los archivos o incluso carpetas correspondientes a cada capa:

- *controller*
- *model*
- *partials*
- ...

Scaffolding y plantillas

domingo, 17 de septiembre de 2017 0:58

Generador es de plantillas

- ❑ Angular seed
<https://github.com/angular/angular-seed>
- ❑ ngBoilerplate
<https://github.com/ngbp/ngbp>
- ❑ Yeoman
<http://yeoman.io/>
- ❑ CleverStack
<http://cleverstack.io/>

Angular seed



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>

The screenshot shows the GitHub repository page for 'angular / angular-seed'. At the top, there's a navigation bar with links for 'Explore', 'Features', 'Enterprise', 'Pricing', 'Sign up', and 'Sign in'. Below the navigation, the repository name 'angular / angular-seed' is displayed along with a star count of 16,112 and a fork count of 5,751. A large green button labeled 'Code' is visible. On the right side, there are links for 'Issues' (38), 'Pull requests' (27), 'Wiki', 'Graphs', and 'Clone in Desktop' (with a circled 'HTTPS clone URL' link). The main area shows a list of 179 commits from the 'master' branch, with the latest commit being 'chore(protractor): update to use protractor v2.1.0' by 'petebacondarwin' on June 15. Other commits include 'chore(bower son): update angular dep to 1.3.x. html5-boilerplate', 'chore(protractor): update to use protractor v2.1.0', 'chore(bower): move bower_components under the app folder', 'chore(dependencies): remove inline dependencies (use npm & bower inst...', 'refact("): further architectural improvements', 'refact("): further architectural improvements', 'docs(LICENSE): update copyright year to 2014', 'docs(README): add alternative git clone method', 'chore(bower.json): Update angular dependency to ~1.4.0', 'refact("): further architectural improvements', and 'chore(protractor): update to use protractor v2.1.0'.

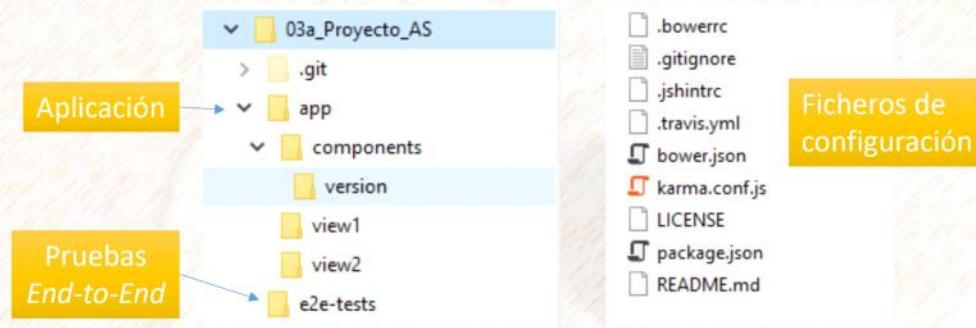
Angular seed: Instalación (1)



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.
```

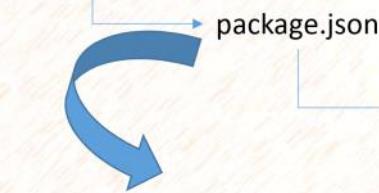


No incluye aún ninguna librería

Angular seed: Instalación (2)



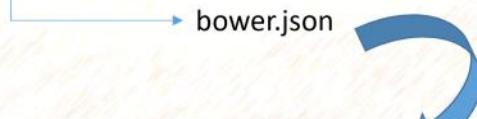
npm install



```
"bower": "^1.3.1",
"http-server": "^0.6.1",
"jasmine-core": "^2.3.4",
"karma": "~0.12",
"karma-chrome-launcher": "^0.1.12",
"karma-firefox-launcher": "^0.1.6",
"karma-jasmine": "^0.3.5",
"karma-junit-reporter": "^0.2.2",
"protractor": "^2.1.0",
"shelljs": "^0.2.6"
```

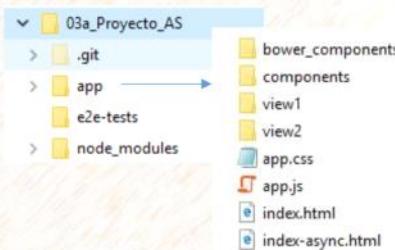
Instalación de las librerías necesarias

incluye y ejecuta bower



```
"angular": "~1.4.0",
"angular-route": "~1.4.0",
"angular-loader": "~1.4.0",
"angular-mocks": "~1.4.0",
"html5-boilerplate": "~5.2.0"
```

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.

The screenshot shows the GitHub repository page for `ngbp / ngbp`. The page title is **ngBoilerplate**, with a large Angular logo icon. Below the title, it says "esqueleto de una aplicación desarrollada ...". A link to the repository is provided: <https://github.com/angular/angular-seed>.

The GitHub interface includes a search bar, navigation links for Explore, Features, Enterprise, and Pricing, and buttons for Sign up and Sign in.

The repository summary indicates 103 commits, 6 branches, 1 release, and 16 contributors. The branch dropdown shows `v0.3.2-release` is selected. The commit list shows recent activity, with a commit from 25 Jan highlighted. A callout box points to the HTTPS clone URL, which is <https://github.com/ngbp/ngbp>. Other options shown include Clone in Desktop and Download ZIP.

Incluye

- Bootstrap
- UI Bootstrap
- Angular UI
- Font Awesome
- LESS
- Grunt
- Angular Placeholders

ngBoilerplate: Instalación



git clone origen destino

```
git clone https://github.com/ngbp/ngbp.git 03b_Proyecto_ngB
```

Es necesario tener instalado / instalar globalmente

Si ya tenemos los dos primeros, en vez de

- grunt
- bower
- karma

```
npm -g install grunt-cli karma bower
```

Ejecutamos **npm -g install karma**

A nivel del proyecto, completamos todas las dependencias

npm install → lee el fichero package.json

bower install → lee el fichero bower.json



ngBoilerplate: Uso

Como está basado en *Grunt*, es necesario ejecutar
(sin cerrar la correspondiente consola)

grunt watch

- supervisa constantemente las carpetas *src*, donde se debe hacer cualquier modificación
- actualiza el contenido de la carpeta *build*, donde se deben comprobar los resultados
- levanta un servidor web
- inicia Firefox con un acceso a karma, para las pruebas



- inicia LiveReload

Proyecto con ngBoilerplate



Accedemos a la carpeta *build* del proyecto

Comprobamos como se refleja en ella cualquier cambio en los ficheros .tpl.html de la carpeta src

Yeoman



Ecosistema de generadores de código y de proyectos de distintos lenguajes y plataforma, incluyendo uno específico para Angular.JS

<http://yeoman.io/>

A screenshot of the Yeoman website homepage. The header features the Yeoman logo (a cartoon character wearing a top hat) and the word "YEOMAN". Navigation links include "Using Yeoman", "Discovering generators", "Creating a generator", "Blog", and "Contributing". The main title "THE WEB'S SCAFFOLDING TOOL FOR MODERN WEBAPPS" is displayed in large white text on a teal background. To the right, there's a cartoon illustration of three characters working on a large white rocket or satellite. Below the title, a call-to-action box contains 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](#). One-line install using [npm](#): `npm install -g yo`".



Yeoman: instalación (1)

Dependencias previas

- Git
 - Node.js
- 
- Ruby
 - Compass
- Pre-procesador
SaSS
- ```
graph LR; A[Git, Node.js] --> B[Ruby, Compass]; B --> C[Pre-procesador SaSS]
```

Si ya tenemos los *Grunt*, *Bower*, en vez de

```
npm -g install yo grunt-cli bower
```

Ejecutamos      **npm -g install yo**

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

```
npm -g install generator-karma generator-angular
```



# 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 quedar 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 [████] 1%
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

Always

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 apps or sites.

Angular

AngularJS is a toolset for building

HTML5 Boilerplate

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

Angular

AngularJS is a toolset for building

Karma

Spectacular Test Runner for JavaScript.

♥ 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



# Controlador (controller)

Es la parte que implementa la lógica de la presentación, de acuerdo con la forma en que angular interpreta el patrón MVC

- puede ser agregado al DOM mediante la directiva *ngController* (anti patrón)
- puede ser definido independientemente, como parte del objeto module de la aplicación

```
angular.module("miApp", []);
angular.module("miApp").controller('MainController', [])
```

En este caso, la directiva **ng-controller**, vincula el controlador a una determinada vista

```
body ng-app="miApp" ng-controller=" 'MainController ">
```





# Controlador y vista

```
<body ng-app="miApp" ng-controller=" 'MainController ">
```

La directiva **ngController** crea un objeto *scope* válido en todo el *body*, y se lo pasa al controlador.

→ Inyecta en el controlador la dependencia al *scope*

```
angular.module("miApp", []);
angular.module("miApp").controller('MainController',
 function ($scope) {
 "valores del Scope"})
```

El **controlador** es una función constructora, cuyo nombre y contenido se definen en el método *controller()*, que recibe el *scope* como parámetro.





# Controlador y Scope

## Scope

- objeto normal de JavaScript que almacena los datos de un modelo

El "scope" es el modelo en AngularJS

- recibe su nombre porque sustituye a `window` como ámbito global a nivel de las vistas

La vista tiene un acceso transparente a esos datos

- es el medio para que el controlador acceda estos datos

para ello el "scope" debe ser "inyectado" en el controlador



# Funciones del controlador



Funciones del controlador;

el controlador  
**no debería** en  
ningún caso

- inicializar los valores de las **variables** del *scope* (el estado del *scope*)
- crear los **métodos** que añaden funcionalidades al *scope*.

- manipular el DOM de la página (los controladores deben de ser agnósticos a cómo está construido el HTML)
- formatear la entrada de datos o filtrar la salida
- incluir código repetido en diversos controladores
- intercambiar estados entre distintos controladores.



# Creación con inyección de dependencias

martes, 1 de agosto de 2017 22:30

## Creación del controlador (1)



Basada en la inyección de dependencias

```
controller(<nombre>, función anónima) // $scope implícito
controller(<nombre>, [$scope, función anónima])
```

```
<div ng-app="pruebaApp" ng-controller="AppCtrl">
 {{algo}}
</div>
```

```
angular.module('pruebaApp', [])
.controller('AppCtrl', function($scope){
 $scope.algo = "Hola Angular";
});
```

La inclusión de las dependencias en un array evita los problemas en caso de minimificación.

```
angular.module('pruebaApp', [])
.controller('AppCtrl', ['$scope', function($scope){
 $scope.algo = "Hola Angular";
}]);
```



Vista (HTML)

```
<body ng-app = "appMain" ng-controller = "MainController">

<header>
 <h1>Formulario de saludo</h1>
</header>
<article>
 <form name="formulario">
 <p>
 <label for="inputNombre" id="lblNombre">Indica tu
 nombre</label>
 <input type="text" id="inputNombre" ng-model="nombre">
 </p>
 <p>
 <label for="inputApellido" id="lblApellido">Indica tu
 apellido</label>
 <input type="text" id="inputApellido" ng-
 model="apellido">
 </p>
 <label></label>
 <input type="button" id="boton"
 value="Borrar nombre" ng-click="borrar()">
 </form>
 <p>Hola {{nombre}} {{apellido}}.</p>
```

Controlador (JS)

```
angular.module('appMain', [])
// controller 2 argumentos:
// - su nombre
// - array de inyección de dependencias
// - nombre de los argumentos
```

```
// - función anónima con dichos argumentos
.controller('AppController', ['$scope', function ($scope)
{
 $scope.user = {
 name : 'Pepe',
 apellido : 'Perez'
 }
 $scope.borrar = function () {
 $scope.user.name='';
 $scope.user.apellido='';
 }
}])
```

## Ejemplo de controller: Acumulador

Utilizando el valor de incremento indicado, aumentamos o reducimos el valor inicial, reflejando el resultado en la parte inferior.

Comprobamos la creación del *controller* con y sin inyección de dependencias

Acumulador

Control de operación:

Incremento:

Totales:

En el acumulador llevamos 0

Alejandro L. Cerezo - Madrid 2015



## Creación sin inyección de dependencias

martes, 1 de agosto de 2017 22:31

John Papa

Modularización extrema

<https://github.com/johnpapa/angular-styleguide>

# Creación del controlador (2)



Alternativamente, el controlador puede utilizar propiedades y métodos de instancia, evitándose la inyección de dependencias

```
angular.module('pruebaApp', [])
.controller('AppCtrl', function(){
 this.algo = "Hola Angular";
});
```

En ese caso, varía la directiva, que emplea la palabra clave **as** para exponer a la vista la instancia de la función constructora correspondiente al controlador

```
<div ng-app="pruebaApp" ng-controller="AppCtrl as $ctrl">
 {{vm.algo}}
```

algunos autores utilizan vm, por *view-model*  
A partir de 1.5 suele usarse \$ctrl



```
<body ng-app="appMain" ng-controller="AppController as vm">
<form>
 <p>
 <label for="name">Nombre</label>
 <input id="name" type="text" ng-model="vm.user.name">
 </p>
 <p>
 <label for="apellido">Apellido</label>
 <input id="apellido" type="text" ng-model="vm.user.apellido">
 </p>
 <p>
 <button ng-click="vm.borrar()">Borrar</button>
 </p>
</form>
<p>Hola {{vm.user.name.toUpperCase()}} {{vm.user.apellido.toUpperCase()}}</p>
```

El controller mapea el \$scope (lo asocia a un alias) en el momento de instanciarlo

Todas las referencias al \$scope desde la vista se realizan a través de su alias (la variable que lo ha mapeado)

Primera opción

```
.controller('AppController', function () {
 this.user = {
 name : 'Pepe',
 apellido : 'Perez'
 }
 this.borrar = function () {
 this.user.name='';
 this.user.apellido='';
 }
})
```

La función constructora del controller instancia el \$scope de forma estándar, haciendo referencia a él mediante this

Segunda opción

```
.controller('AppController', AppController)

function AppController () {
 this.user = {
 name : 'Pepe',
 apellido : 'Perez'
 }
}
```

Como cualquier función anónima, la función constructora del controller puede declararse como función con nombre

```
this.borrar = function () {
 this.user.name='';
 this.user.apellido='';
}
}
```

# Empleo de ES6

martes, 1 de agosto de 2017 22:33

The slide features a portrait of Todd Motto at the top left. Below it, a blue arrow points down from his name to the text 'Estilo ES2015'. To the right is a screenshot of a GitHub repository page for 'toddmotto/angular-styleguide'. The repository has 186 commits, 3 branches, 0 releases, and 42 contributors. A box highlights the URL <https://github.com/toddmotto/angular-styleguide>. The page also includes a brief description: 'AngularJS styleguide for teams https://ultimateangular.com'.

Todd Motto

↓

Estilo ES2015

<https://github.com/toddmotto/angular-styleguide>

AngularJS styleguide (ES2015)

Up-to-date with AngularJS 1.6 best practices. Architecture, file structure, components, one-way dataflow, lifecycle hooks.



# Creación del controlador en ES6



El controlador basado en propiedades y métodos de instancia, (sin inyección de dependencias) puede reescribirse con el formato de ES6

```
Class AppCtrl{
 constructor () {}
 $onInit () {
 this.algo = "Hola Angular";
 }
};

angular.module('pruebaApp', [])
.controller('AppCtrl', AppCtrl);

<div ng-app="pruebaApp" ng-controller="AppCtrl as $ctrl">
 {{$ctrl.algo}}
</div>
```



```
<body ng-app="appMain" ng-controller="AppController as $ctrl"> ←
 <form>
 <p>
 <label for="name">Nombre</label>
 <input id="name" type="text" ng-model="$ctrl.user.name">
 </p>
 <p>
 <label for="apellido">Apellido</label>
 <input id="apellido" type="text" ng-model="$ctrl.user.apellido">
 </p>
 <p>
 <button ng-click="$ctrl.borrar()">Borrar</button>
 </p>
 </form>
 <p>Hola {{$ctrl.user.name.toUpperCase()}} {{$ctrl.user.apellido.toUpperCase()}}</p>
</body>
</html>
```

Por similitud con los componentes de AngularJS 1.5, el alias del \$scope suele ser \$ctrl

```
class AppController { ←
 constructor () {
 // Sólo se usa para inyección de dependencias
 }

 $onInit () { ←
 this.user = {
 name : 'Pepe',
 apellido : 'Perez'
 }
 }

 borrar () { ←
 this.user.name = '';
 this.user.apellido = '';
 }
}
```

La función responsable de instanciar el \$scope se refactoriza como una clase, de acuerdo con la nueva forma de escribir el código introducida en ES6

De acuerdo con la recomendación de Angular, toda la inicialización de propiedades del objeto se reubica en el método \$onInit, específico de Angular, dejando el constructor exclusivamente para la inyección de dependencias, en caso de que exista

Cualquier método que se utilice en la vista se declara como parte de la clase

```
angular.module('appMain')
.controller('AppController', AppController)
```

## Ejemplo de controller y clases

Utilizando el valor de incremento indicado, aumentamos o reducimos el valor inicial, reflejando el resultado en la parte inferior.

Comprobamos la creación del *controller* con y sin inyección de dependencias

**Acumulador**

Control de operación:

Incremento: 10

Totales:  
En el acumulador llevamos 0

Alejandro L. Cerezo - Madrid 2015

