<https://desarrolloweb.com/manuales/manual-angularjs.html>

ANGULAR BÁSICO

1. ¿Qué es AngularJs?
   1. Diferencia entre AngularJS (1.x) y Angular (2+)
   2. AngularJS vs jQuery
   3. API AngularJS
2. Primeros pasos
   1. Herramientas
      1. Navegador
      2. Editor
      3. Visual Studio Code
   2. Instalación
      1. Descargar AngularJS o CDN de AngularJS
      2. Añadir AngularJS
   3. Hola Mundo
3. Expresiones
4. Modules
   1. config
   2. run
5. Directives
   1. ngApp, ngController
   2. ngClick
   3. ngDisabled
   4. ngIf, ngShow, ngHide, ngSwitch
   5. ngInit, ngModel
   6. ngChange
   7. ngRepeat, ngOptions
   8. ngClass
   9. Otras directivas
6. Controllers
   1. $scope
   2. $rootScope
   3. $watch
7. Data binding
   1. One-way binding
   2. Two-way binding
8. Filters
   1. lowercase, uppercase
   2. number
   3. date
   4. custom filter
9. Services
   1. $timeout
   2. $http
   3. $filter
   4. Tipos de servicios
      1. Constant
      2. Value
      3. Service
      4. Factory
      5. Provider
10. Promises
    1. ¿Qué son?
    2. $q
11. Routing
    1. $states
    2. $stateParams
    3. $routeProvider
12. Estructura de una aplicación AngularJS
13. Otros
    1. Angular Translate
    2. Angular Material

# ¿Qué es AngularJS?

Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un **framework** para el desarrollo, en este caso sobre el lenguaje Javascript con programación del lado del cliente.

Un framework es un entorno de trabajo, un conjunto de herramientas, la cuales nos permiten trabajar de forma sencilla.

Angular promueve y usa patrones de diseño de software. En concreto implementa lo que se llama **MVC**. Este patrón marca la separación del código de los programas dependiendo de su responsabilidad. Eso permite repartir la lógica de la aplicación por capas, lo que resulta muy adecuado para aplicaciones de negocio y para las aplicaciones **SPA** (Single Page Aplication).

Las SPA o "Aplicaciones de una sola página", son sitios web donde los usuarios perciben una experiencia similar a la que se tiene con las aplicaciones de escritorio. En este tipo de sitios la página no se recarga, no existe una navegación de una página a otra totalmente diferente, sino que se van intercambiando las "vistas". Técnicamente podríamos decir que, al interactuar con el sitio, el navegador no recarga todo el contenido, sino únicamente vistas dentro de la misma página.

Primeramente tenemos que hablar sobre el gran patrón que se usa en Angular, el conocido Modelo, Vista, Controlador.

* **Vistas:** Será el HTML y todo lo que represente datos o información.
* **Controladores:** Se encargarán de la lógica de la aplicación y sobre todo de las llamadas "Factorías" y "Servicios" para mover datos contra servidores o memoria local en HTML5.
* **Modelo de la vista:** En Angular el "Modelo" es algo más de aquello que se entiende habitualmente cuando te hablan del MVC tradicional, o sea, las vistas son algo más que el modelo de datos. En modo de ejemplo, en aplicaciones de negocio donde tienes que manejar la contabilidad de una empresa, el modelo serían los movimientos contables. Pero en una pantalla concreta de tu aplicación es posible que tengas que ver otras cosas, además del movimiento contable, como el nombre de los usuarios, los permisos que tienen, si pueden ver los datos, editarlos, etc. Toda esa información, que es útil para el programador pero que no forma parte del modelo del negocio, es a lo que llamamos el "scope" que es el modelo en Angular.

Además del patrón principal, descrito hasta ahora tenemos los módulos:

**Módulos:** La manera que nos va a proponer AngularJS para que nosotros como desarrolladores seamos cada vez más ordenados, que no tengamos excusas para no hacer un buen código, para evitar el código espaguetti, ficheros gigantescos con miles de líneas de código, etc. Podemos dividir las cosas, evitar el infierno de las variables globales en Javascript, etc. Con los módulos podemos realizar aplicaciones bien hechas, de las que un programador pueda sentirse orgulloso y sobre todo, que nos facilite su desarrollo y el mantenimiento.

**HTML**

**JS**

**$scope**

**vistas**

**ctrl**

**directivas**

**factory**

**filtros**

**service**

## Diferencia entre AngularJS (1.x) y Angular (2+)

Antes de nada hay que tener claro que, a pesar de compartir el nombre, AngularJS y Angular no tienen nada que ver. El mal llamado "Angular 2", que se llama solamente Angular, no es una nueva versión de AngularJS. Es un nuevo framework, escrito desde cero y con conceptos y formas de trabajar completamente distintos.

Angular tiene una curva de aprendizaje bastante empinada para los desarrolladores. Está todo escrito en Typescript y cumple con la especificación ES6, además incluye muchas innovaciones que rompen con su predecesor.

[Algunas diferencias entre AngularJS y Angular](https://www.campusmvp.es/recursos/post/las-10-principales-diferencias-entre-angularjs-y-angular.aspx)

## AngularJS vs jQuery

Son librerías bien diferentes. El alcance y el tipo de cosas que se hacen con una y otra librería son distintos.

**jQuery** es una librería que nos sirve para acceder y modificar el estado de cualquiera de los elementos de la página. A través de jQuery y los selectores de CSS (así como los selectores creados por el propio jQuery) eres capaz de llegar a los elementos de la página, a cualquiera de ellos, y puedes leer y modificar sus propiedades, suscribirte a eventos que ocurran en esos elementos, etc. Con jQuery podíamos manejar cualquier cosa que ocurra en esos elementos de una manera mucho más cómoda que con Javascript "a pelo" y compatible con la mayor gama de navegadores.

Sin embargo **AngularJS** pasa de ser una librería para convertirse en un framework de aplicaciones web. No solo te permite una serie de funciones y mecanismos para acceder a los elementos de la página y modificarlos, sino que también te ofrece una serie de mecanismos por los cuales extender el HTML, para hacerlo más semántico, incluso ahorrarte muchas líneas de código Javascript para hacer las mismas cosas que antes hacías con jQuery. Pero la principal diferencia es que te marca una serie de normas y hábitos en la programación, principalmente gracias al patrón MVC. Las directivas que trae AngularJS usan jQuery por debajo, o algo muy similar.

De todos modos, si quieres utilizar jQuery dentro de AngularJS puedes hacerlo aunque no es necesario.

# Primeros pasos

## Herramientas

Lo único que necesitas para desarrollar con AngularJS es un editor de texto y un navegador. Así de sencillo, seguramente todo el mundo tenga ya configurado su editor preferido, así que te puedes saltar los siguientes pasos.

### Navegador

Para ver un desarrollo basado en Angular puedes utilizar cualquier navegador. Generalmente tendrás incluso varios navegadores para probar tu página en cada uno de ellos y comprobar que todo está en orden. Solo te recomendamos tener a Google Chrome entre tu batería de navegadores, pues luego hablaremos de una extensión que existe para éste que te permite examinar y depurar páginas donde AngularJS está trabajando.

### Editor

Puedes utilizar cualquiera, si estás acostumbrado a usar uno, no necesitas cambiarlo para programar con Angular. Lo que es interesante es que sea un editor para programadores, que te permita diversas ayudas al escribir el código, coloreado de sintaxis, ayudas contextuales, etc. Notepad++, Sublime Text, Komodo Edit, Brackets, Visual Studio Code, etc. Cualquiera es bueno. Si estás acostumbrado a IDEs más complejos como Eclipse o PhpStorm, puedes quedarte también con ellos.

### Visual Studio Code

Nosotros recomendamos este editor porque además de incluir todas las ventajas de cualquier editor incluye una consola para utilizar NPM, Git,...

[Descarga](https://code.visualstudio.com/Download)

## Instalación

Para trabajar con AngularJS tienes que incluir el script del framework en tu página. Esto lo puedes hacer de varias maneras, o bien te descargas la librería por completo y la colocas en un directorio de tu proyecto, o bien usas un CDN para traerte la librería desde un servidor remoto.

### Descargar AngularJS o CDN de AngularJS

Accedes a la página de AngularJS (<https://angularjs.org/>) y en función de la manera que vayas a utilizarlo, realizas una acción u otra. Nosotros vamos a comenzar del modo más sencillo, que es utilizar el CDN.

Pulsas el botón de descarga y encontrarás diversas opciones. Escoges la versión del framework (si es que te lo permite) y que esté minimizada (minified). Luego encontrarás un campo de texto donde está la URL de la librería (esa URL está marcada con las siglas "CDN" al lado). Ya sabes que el CDN te ofrece un contenido, en este caso un script Javascript, que está alojado en otro servidor, pero que lo puedes usar desde tu página para mejorar la entrega del mismo.

Será algo como <https://ajax.googleapis.com/ajax/libs/angularjs/1.7.5/angular.min.js> aunque todo depende de la versión que haya en el momento en el que tú leas esta guía de iniciación.

### Añadir AngularJS

Una vez tienes tu CDN puedes incluir el script de Angular en la página con la etiqueta SCRIPT. Ese script lo puedes colocar en el HEAD o bien antes del final del BODY, en principio no habría diferencias en lo relativo a la funcionalidad, pero sí hay una pequeña mejora si lo colocas antes de cerrar el cuerpo.



## Hola Mundo

# Expresiones

Las expresiones se utilizan para enriquecer el HTML, ya que nos permiten colocar cualquier cosa y que AngularJS se encargue de interpretarla y resolverla. Para crear una expresión simplemente la englobas dentro de dobles llaves, de inicio y fin.

Ahora dentro de tu aplicación, en cualquier lugar de tu código HTML delimitado por la etiqueta donde pusiste la directiva ng-app eres capaz de colocar expresiones. Hagamos una prueba colocando una expresión así.

Angular cuando se pone a ejecutar la aplicación buscará expresiones de este estilo y lo que tengan dentro, si es algo que él pueda evaluar, lo resolverá y lo sustituirá con el resultado que corresponda.

Puedes probar otra expresión como esta:

Al ejecutarlo, AngularJS sustituirá esa expresión por "Hola DesarrolloWeb". Estas expresiones no me aportan gran cosa de momento, pero luego las utilizaremos para más tipos de operaciones.

Lo habitual de las expresiones es utilizarlas para colocar datos de tu modelo, escribiendo los nombres de las "variables" que tengas en el modelo. Por ejemplo, si tienes este código HTML, donde has definido un dato en tu modelo llamado "valor":



Podrás volcar en la página ese dato (lo que haya escrito en el campo INPUT) por medio de la siguiente expresión:



Otro detalle interesante de las expresiones es la capacidad de formatear la salida, por ejemplo, diciendo que lo que se va a escribir es un número y que deben representarse dos decimales necesariamente. Vemos rápidamente la sintaxis de esta salida formateada, y más adelante la utilizaremos en diversos ejemplos:

Cabe decir que estas expresiones no están pensadas para escribir código de tu programa, osea, lo que llamamos lógica de tu aplicación. Como decíamos, están sobre todo pensadas para volcar información que tengas en tu modelo y facilitar el "binding". Por ello, salvo el operador ternario (x ? y : z), no se pueden colocar expresiones de control como bucles o condicionales. Sin embargo, desde las expresiones también podemos llamar a funciones codificadas en Javascript (que están escritas en los controladores, como veremos enseguida) para poder resolver cualquier tipo de necesidad más compleja.

# Modules

Los módulos son una de las piezas fundamentales en el desarrollo con AngularJS y nos sirven para organizar el código en esta librería.

Desde Javascript crearás los módulos usando el método angular.module() e indicándole una serie de parámetros. Echa un vistazo a esta sintaxis.

La variable "angular" la tienes como variable global cuando cargas AngularJS, dentro tiene un objeto que estará disponible en cualquier parte de tu código. Luego ves "module", que es un método del objeto "angular" y que sirve para crear el módulo. El primer argumento es el nombre del módulo, que corresponde con el nombre de tu aplicación. En el segundo parámetro puedes indicar una serie de módulos adicionales, separados por comas, que serían tus dependencias. Pueden ser módulos de otros autores o puede que tú mismo hayas decidido organizar tu código en diferentes módulos. El tercer parámetro es opcional y en él colocamos una función que sirve para configurar AngularJS.

Para comenzar crearás tus módulos probablemente de una manera sencilla, tal como puedes ver a continuación:

En el código anterior "nuevaApp" es el nombre de tu módulo. Como se comentó anteriormente, ese módulo se debe indicar en el bootstrap (arranque) de tu aplicación, comúnmente en la directiva ngApp.

## config

## run

# Directives

Las directivas son nuevos "comandos" que vas a incorporar al HTML y los puedes asignar a cualquiera de las etiquetas por medio de atributos. Son como marcas en elementos del DOM de tu página que le indican a AngularJS que tienen que asignarles un comportamiento determinado o incluso transformar ese elemento del DOM o alguno de sus hijos.

Cuando se ejecuta una aplicación que trabaja con Angular, existe un "HTML Compiler" (Compilador HTML) que se encarga de recorrer el documento y localizar las directivas que hayas colocado dentro del código HTML, para ejecutar aquellos comportamientos asociados a esas directivas.

AngularJS nos trae una serie de directivas "de fábrica" que nos sirven para hacer cosas habituales, así como tú o terceros desarrolladores pueden crear sus propias directivas para enriquecer el framework.

## ngApp

Esta es la marca que indica el elemento raíz de tu aplicación. Se coloca como atributo en la etiqueta que deseas que sea la raíz. Es una directiva que autoarranca la aplicación web AngularJS. Se leería "Enyi ap" y lo más común es ponerlo al principio de tu documento HTML, en la etiqueta HTML o BODY, pero también lo podrías colocar en un área más restringida dentro del documento en otra de las etiquetas de tu página.

**Nota:** Como puedes comprobar, ng-app es una directiva y se indica como si fuera un atributo el HTML. Pero no lo es ningún atributo real de HTML. Si usas un validador de HTML te advertirá que ese atributo es inexistente y se interpretará por un error de validación. Para solucionar este posible inconveniente, la práctica aconsejable es colocar el prefijo "data-" a cada directiva.

De ese modo tu código validará perfectamente, ya que en HTML5 se pueden crear cualquier tipo de atributos personalizados que comiencen por "data-" y asignarles cualquier tipo de dato que queramos almacenar en la etiqueta.

Para no causar confusiones, también podemos agregar que a nivel de Javascript las directivas las encontrarás nombradas con notación "camel case", algo como ngApp. En la documentación también las encuentras nombradas con camel case, sin embargo, como el HTML no es sensible a las mayúsculas y minúsculas no tiene tanto sentido usar esa notación y por ello se separan las palabras de las directivas por un guión "-".

Opcionalmente ngApp puede contener como valor un módulo de AngularJS a cargar. Esto lo veremos más adelante cuando trabajemos con módulos.

## ngController

## ngClick

Terminamos nuestra dosis de teoría con una explicación de la directiva ngClick. Como podrás imaginarte es utilizada para especificar un evento click. En ella pondremos el código (mejor dicho la expresión) que se debe ejecutar cuando se produzca un clic sobre el elemento donde se ha colocado la directiva.

Típicamente al implementar un clic invocarás una función manejadora de evento, que escribirás de manera separada al código HTML.

Esa función procesarClic() la escribirás en el controlador, factoría, etc. Sería el modo aconsejado de proceder, aunque también podrías escribir expresiones simples, con un subconjunto del código que podrías escribir con el propio Javascript. Incluso cabe la posibilidad de escribir varias expresiones si las separas por punto y coma.

No difiere mucho a como se expresan los eventos clic en HTML mediante el atributo onclick, la diferencia aquí es que dentro de tus expresiones podrás acceder a los datos que tengas en tu modelo.

**Nota:** Aunque técnicamente pueda escribir expresiones directamente en el código HTML, en el valor del atributo ng-click, tienes que evaluar con cuidado qué tipo de código realizas porque dentro de la filosofía de AngularJS y la del MVC en general, no puedes escribir en tu HTML código que sirva para implementar la lógica de tu aplicación. (El código que necesitas para hacer las funcionalidades de tu aplicación no lo colocas en la vista, lo colocas en el controlador).

## ngDisabled

## ngIf, ngShow, ngHide, ngSwitch

## ngInit

Esta directiva nos sirve para inicializar datos en nuestra aplicación, por medio de expresiones que se evaluarán en el contexto actual donde hayan sido definidas. Dicho de otra manera, nos permite cargar cosas en nuestro modelo, al inicializarse la aplicación.

Así de manera general podemos crear variables en el "scope", inicializarlas con valores, etc. para que en el momento que las vayas a necesitar estén cargadas con los datos que necesitas.

**Nota:** Realmente no hemos explicado todavía en el [Manual de AngularJS](http://www.desarrolloweb.com/manuales/manual-angularjs.html), al menos suficientemente bien, qué es eso del "scope" y ya estamos abusando en el número de veces que usamos el término. Por ahora hemos comentado que el scope mantiene los datos de tu aplicación, lo que en el paradigma de la programación MVC se llama "modelo". Así pues, si digo que creo variables en el scope lo puedes entender como la definición de un dato de tu aplicación, que se guarda en el denominado "modelo" del MVC. Para los que tienen dudas sobre qué es el MVC os recomendamos leer el artículo [Qué es MVC](http://www.desarrolloweb.com/articulos/que-es-mvc.html).

Con esto consigues que tu aplicación inicialice en el scope un dato llamado miArrayDatos como un array vacío. Pero no le prestes demasiada atención al hecho de haber colocado la directiva ngInit dentro de la misma etiqueta que inicializa la aplicación, pues podría ir en cualquier otra etiqueta de tu HTML. Realmente, colocarla en esa división marcada con ngApp es considerado una mala práctica. Ten en cuenta lo siguiente cuando trabajes con ngInit:

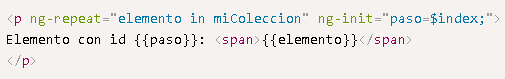
El único caso apropiado donde se debería de usar ngInit es en enlace de propiedades especiales de ngRepeat. Si lo que quieres es inicializar datos en tu modelo para toda la aplicación, el lugar apropiado sería en el controlador. Enseguida vemos un ejemplo de uso apropiado, cuando conozcamos la directiva ngRepeat.

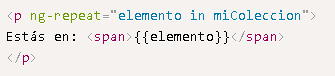
**Nota:** Aclaramos ya que en el ejemplo final de este artículo vamos a usar ngInit para asociar datos al scope, e inicializarlos. Sin embargo, debemos de ser conscientes que para ello deberíamos usar un controlador. Como no hemos llegado a los controladores todavía, y queremos limitarnos solo a trabajar con el HTML, nos sirve como método alternativo.

## ngModel

## ngChange

## ngRepeat

Esta directiva te sirve para implementar una repetición (un bucle). Es usada para repetir un grupo de etiquetas una serie de veces. Al implementar la directiva en tu HTML tienes que decirle sobre qué estructura se va a iterar. ngRepeat se usa de manera muy habitual y se verá con detenimiento en decenas de ejemplos. De momento puedes quedarte que es como un recorrido for-each en el que se itera sobre cada uno de los elementos de una colección.

La etiqueta donde has colocado el atributo ng-repeat y todo el grupo de etiquetas anidadas dentro de ésta, funciona como si fuera una plantilla. Al procesarse el compilador HTML de AngularJS el código HTML de esa plantilla se repite para cada elemento de la colección que se está iterando. Dentro de esa plantilla tienes un contexto particular, que es definido en la declaración de la directiva, que equivale al elemento actual en el bucle. Se ve mejor con un ejemplo.

El dato miColeccion sería un dato de tu modelo, habitualmente un array sobre el que puedas iterar, una vez por cada elemento. Pero también podría ser un objeto y en ese caso la iteración se realizaría en cada una de sus propiedades.

En lo relativo al contexto própio del bucle te puedes fijar que dentro de la iteración podemos acceder al dato "elemento", que contiene como valor, en cada repetición, el elemento actual de la colección sobre el que se está iterando.

Esta directiva es bastante sofisticada y explicar cada una de sus posibilidades nos llevaría algo de tiempo. Sin embargo, vamos a mostrar cómo podemos trabajar con ngRepeat en conjunto con ngInit, para completar la explicación de punto anterior.

La directiva ngRepeat maneja una serie de propiedades especiales que puedes inicializar para el contexto propio de cada repetición. Para inicializarlas usamos la directiva ngInit indicando los nombres de las variables donde vamos a guardar esas propiedades. En este caso estamos indicando que dentro de la repetición vamos a mantener una variable "paso" que tendrá el valor de $index, que equivale al número índice de cada repetición. Osea, en la primera iteración paso valdrá cero, luego valdrá uno y así todo seguido. Igual que tienes $index, Angular te proporciona otras propiedades útiles como $first (que valdrá true en caso que sea la primera iteración) o $last (true solo para la última iteración)

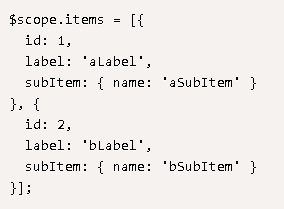
**Nota:** El que acabas de ver sería el único uso de ngInit considerado como buena práctica, en conjunto con ngRepeat para inicializar sus propiedades.

Dejamos de momento ngRepeat, aunque vuelvo a señalar que hay otra serie de cosas interesantes en la directiva para las repeticiones, como filtros, órdenes, etc.

## ngOptions

Se puede usar para generar dinámicamente una lista de <option> al usar un <select> siempre que usemos un array para evaluar la expresión.

En muchos casos, se puede usar [ngRepeat](https://docs.angularjs.org/api/ng/directive/ngRepeat) con varios <option> en lugar de ngOptions para lograr un resultado similar. Sin embargo, es más óptimo utilizar ngOptions:





## ngClass

Nos ayuda a definir clases CSS que se le deben aplicar a los elementos bajo ciertas circunstancias.

Existen tres posibles variantes:

* Asignarle una propiedad del scope que sea una cadena de texto. En este caso esa cadena se coloca como valor en el atributo class. Si en esa cadena existen varios nombres de clases separados por un espacio en blanco, esas clases se aplicarán en conjunto al elemento.
* Asignarle una propiedad del scope que contenga un array de cadenas. En ese caso se asignan como clases todos los nombres que haya en las casillas del array.

Es tan sencillo como definir un array de alguna manera y luego introducirlo dentro de nuestra directiva. Por facilitar las cosas voy a definir el array de manera literal en el controlador.

Luego podremos asociar ese array de cadenas, colocando todos los nombres de clases a un elemento de la página con la directiva ngClass.

Como resultado de esa directiva a nuestro encabezado H2 se le van a aplicar las tres clases "uno", "dos" y "tres".

* Asignarle como valor a ng-class un objeto. En ese caso tendrá pares clave valor para especificar nombres de clases y expresiones que deban cumplirse para que éstas se apliquen. Lo veremos mejor con un ejemplo.

## Otras directivas