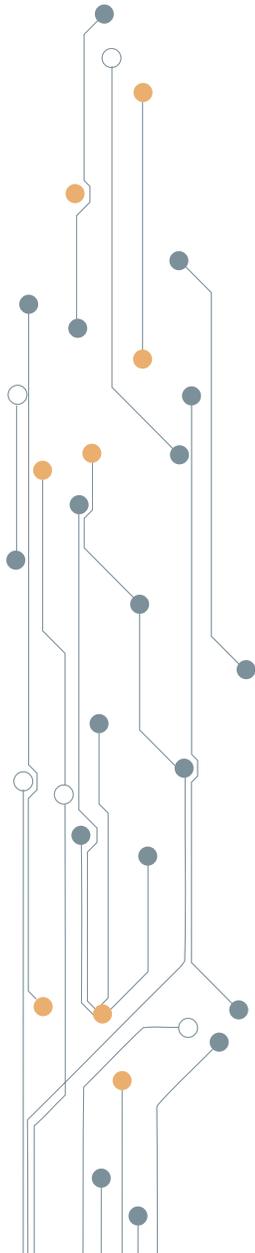




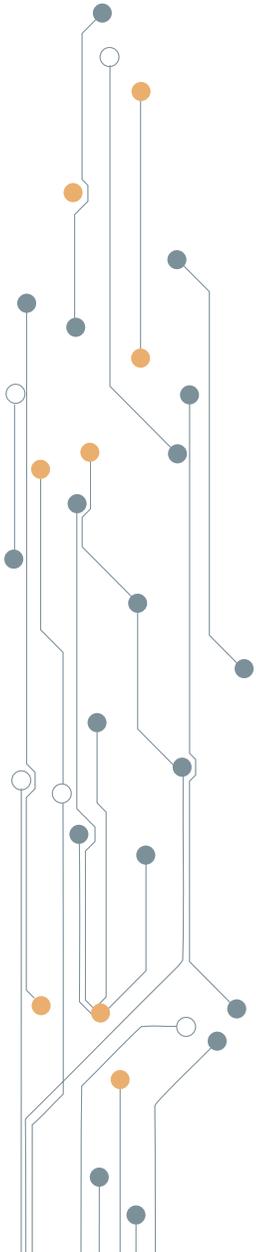
La librería AngularJS

Índice



1 La librería AngularJS	3
1.1 Descarga la librería	6
1.2 Principios de funcionamiento y elementos básicos	7
1.3 Directivas	8
ngApp (ng-app)	8
ngController (ng-controller)	8
La <u>sintaxis</u> de ngController:	8
ngModel (ng-model)	9
ngClick (ng-click)	9
ngInit (ng-init)	9
ngRepeat (ng-repeat)	10
ngChange (ng-change)	10
ngApp (ng-app)	10
En el lado del controlador	11
ngShow (ng-show) ngHide (ng-hide)	11
ngBind (ng-bind)	12
Creando nuestras propias directivas	12
1.4 Controllers (Controladores)	13
ngApp (ng-app)	13
<u>Propiedades del Controlador</u>	14
<u>Métodos del Controlador</u>	14
<u>Controladores en ficheros externos</u>	15

Índice



1.5 Módulos	15
<u>Ejemplo de Módulo en AngularJS</u>	15
Los Controladores ensucian el ámbito Global de JavaScript	16
Definición de los Módulos	17
Ficheros de una Aplicación <u>AngularJS</u>	18

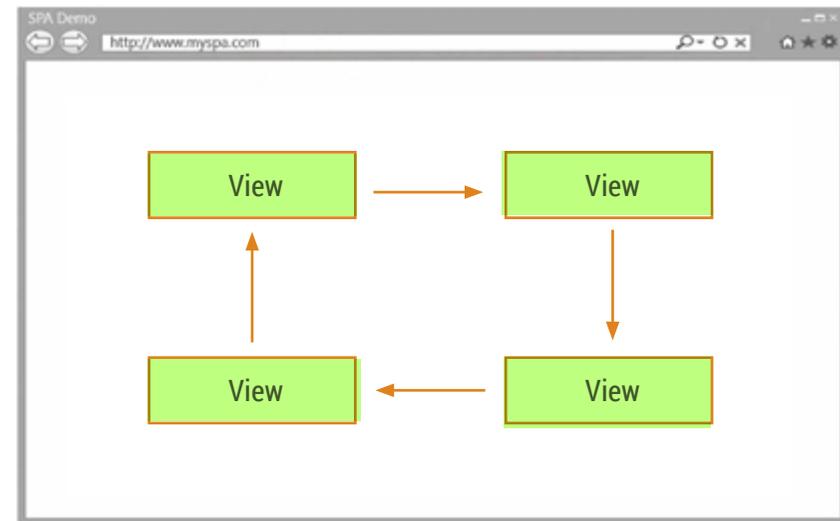
1. La librería AngularJS

Nos encontramos ante un **framework JavaScript** de **código abierto** que se denomina **AngularJS**, que está siendo **respaldado** por **Google**, y ayuda con la **construcción** de las **Single Page Applications** o aplicaciones de una sola página.

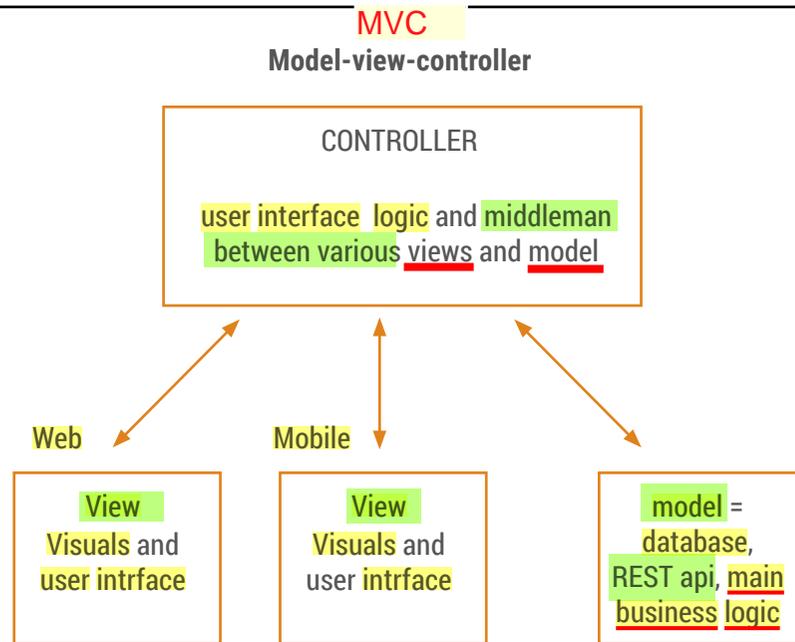
El **patrón Single Page Applications** define que podemos **construir** o **desarrollar aplicaciones web** en una **única página html**, teniendo **todo el ciclo de vida seleccionado en** dicha **página**, y **variando** los **componentes** y **controles** con **códigos JavaScript** y las **librerías** o **frameworks** como **AngularJS**.

Aparte, también es adecuado seguir el **patrón Modelo Vista Controlador (MVC)**, que muchos **otros frameworks** de desarrollo lo **programaban** en el **lado servidor**, pero que con **AngularJS** se hace **factible desarrollar** en el **lado cliente**.

Single Page Application (SPA)



LADO DE CLIENTE



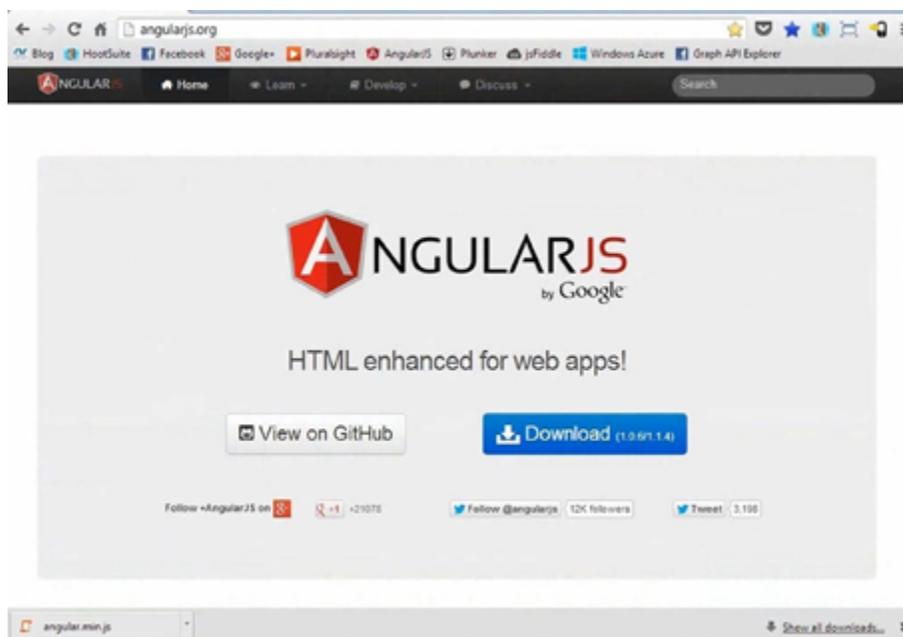
En un mundo en el que necesitamos mucha programación en la **capa** de las **vistas**, **con diferentes dispositivos** que **pueden llegar** a **nuestra aplicación** (desde **terminales móviles** hasta **ordenadores** de sobremesa), la **utilización** de **frameworks** como **AngularJS** no solo es buena sino que se hace inmediatamente necesaria.

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.



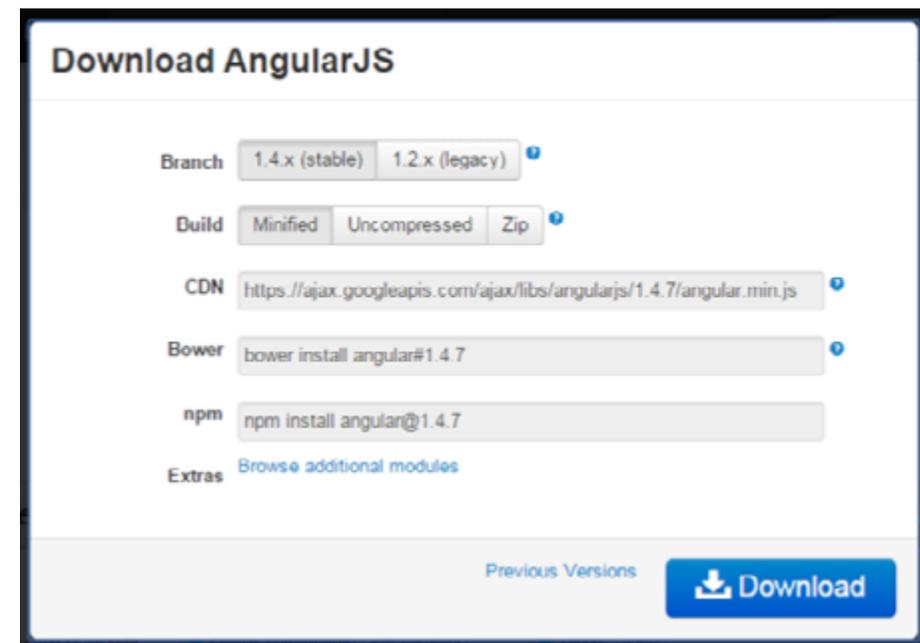
1.1 | Descarga la librería

Para empezar con AngularJS únicamente se tiene que descargar la librería que se encuentra en la página <https://angularjs.org/>



En esta misma web también podemos leer documentación muy variada y un gran número de ejemplos para poder comenzar.

Podemos descargar versiones mínimas o versiones más completas con librerías de terceros.



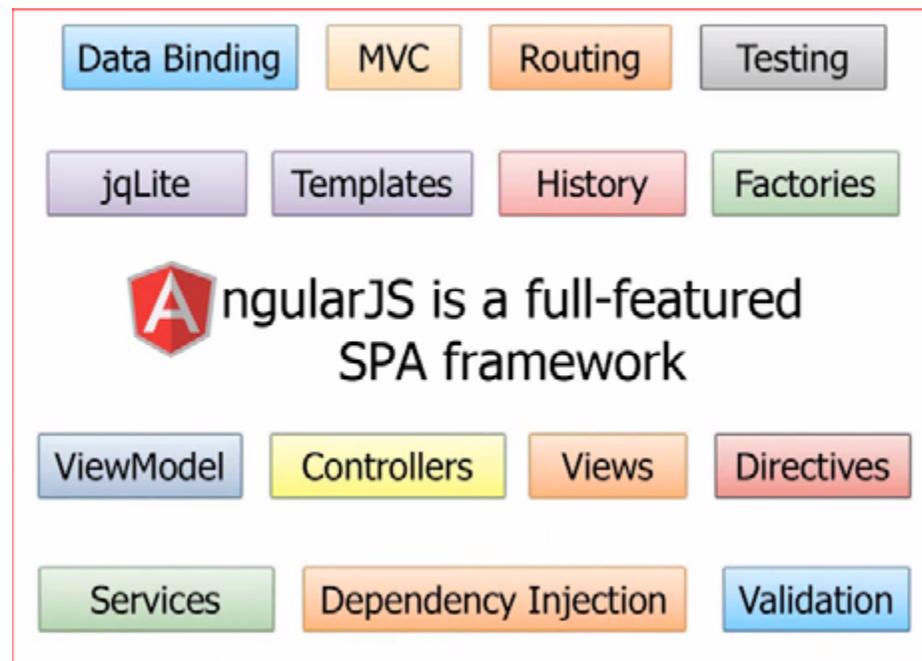
1.2 | Principios de funcionamiento y elementos básicos

AngularJS revisa el HTML que puede contener atributos de las etiquetas personalizadas adicionales (de la propia librería), obedece a las directivas de los atributos personalizados, y une los elementos de entrada o salida del documento a un modelo representado por las variables de JavaScript.

Los valores de las variables de JavaScript se pueden actualizar de manera manual, o ser recuperados de los JSON estáticos o dinámicos.

Como se puede ver en la imagen, hay muchos elementos y conceptos disponibles en la librería que pasaremos a comentar durante este manual.

enlace de datos



jqLite - tiny JavaScript DOM query library that uses pure CSS selectors

1.3 | Directivas

Las **directivas** son **atributos** que se **incluyen dentro** de las **etiquetas** de nuestro código **HTML** para **conectar** la **librería AngularJS** con esos **componentes** que **existen en** **nuestro mapa documental**.

Directivas hay muchas, y ahora pasaremos a redactar las más importantes.

ngApp (ng-app)

Auto arranca una **aplicación Angular**, nos **selecciona** el **elemento raíz** y se **sitúa como atributo dentro** de la **etiqueta que deseas** que sea la **raíz** de la **aplicación**.

Se declara tal como sigue:

```
<html ng-app>
```

directiva
↓

Aunque más adelante veremos cómo **declarar módulos**, se puede **enlazar** un **módulo** a esta **directiva** de la siguiente manera:

```
<html ng-app="nombre_del_modulo">
```

ngController (ng-controller)

Directiva que **se usa para** los **controladores**. Y aunque recuperaremos esta directiva en la sección posterior vamos a darle una definición:

La **directiva** que **indica a la vista** que **trabaja con** nuestro **controlador** y **enlaza** un **\$scope**, el **modelo** que **esté dentro del ámbito** de la **directiva** podrá **ser accedido** desde el **controlador que nos hemos asignado**.

La sintaxis de ngController:

```
<body>
  <div ng-controller="nombre_de_controlador">
    <h1>Hola </h1>
  </div>
</body>
```

ngModel (ng-model)

Directiva que muestra el modelo o dato, ^{input} obtiene la información insertada por el usuario en algún componente de nuestro formulario, sea cual fuere. Si se quiere obtener el texto que nuestro cliente rellena en un input, basta con asociarle un modelo y entonces será accedido tanto en el controlador como la vista mediante el nombre del modelo.

Su sintaxis:

```
<body>
  <div ng-controller="miControlador">
    <label>Nombre</label><input type="text" ng-
model="nombre">
    <span>Hola {{nombre}}</span>
  </div>
</body>
```

ngClick (ng-click)

Relacionado con el evento click, al que se le puede asociar cualquier tipo de funcionalidad.

```
<body>
  <div ng-controller="miControlador">
    <label>Nombre:</label><input type="text" ng-
model="nombre">
    <button ng-click="enviar()">Enviar</button>
  </div>
</body>
```

ngInit (ng-init)

Permite evaluar una expresión en el scope donde se está trabajando.

```
<body>
  <div ng-controller="miControlador">
    <div>
      <button ng-click="count = count + 1" ng-
init="count = 0">Enviar</button>
      <span>{{count}}</span>
    </div>
  </div>
</body>
```

ngRepeat (ng-repeat)

Permite **iterar** una **colección de datos**, **crear** una **platilla** (template) por cada elemento de la colección y mostrarlo en la vista.

```

<body>
  <div ng-controller="miControlador">
    <div ng-init="lista =
  [{nombre:Fer, edad:12},{nombre:Victor, edad:13},
    {nombre:Eva, edad:14},{nombre:Susana,
  edad:15},
    {nombre:Alejo, edad:16} ]"/>
    <ul>
  <li ng-repeat="a in lista">{{a.nombre}}: {{a.edad}}
  años</li>
    </ul>
  </div>
</div>
</body>

```

ngChange (ng-change)

Detecta **cualquier variación** que se **produzca dentro** de una **etiqueta de entrada**, la manera de usarla es la siguiente:

```

<body>
  <div ng-controller="miControlador">
    <input type="checkbox" ng-model="si" ng-
  change="favor()"> A favor
    <input type="checkbox" ng-model="no" ng-
  change="contra()"> En contra
    <h3>Total Votos: {{total}}</h3>
  </div>
</body>

```

EN EL LADO DEL CONTROLADOR:

```

app.controller('miControlador', function($scope){
  $scope.total = 0;
  $scope.favor = function (){$scope.total++;};
  $scope.contra = function (){$scope.total--};
});

```

ngShow (ng-show) | ngHide (ng-hide)

Permiten mostrar y ocultar alguna parte de la vista según una condición que podemos asignarle. `ngShow` muestra y `ngHide` oculta.

```
<body>
  <div ng-controller="miControlador">
    <input type="checkbox" ng-model="dato1">
Muestra
    <input type="checkbox" ng-model="dato2">
Oculto
    <h3 ng-show="dato1">Hola</h3>
    <h3 ng-hide="dato2">Adiós</h3>
  </div>
</body>
```

ngBind (ng-bind)

Esta `directiva` tiene la `misma función` que las `llaves {{}}`, sin embargo, `ng-bind` tiene un `mejor funcionamiento` en cuanto a tiempo.

```
<body>
  <div ng-controller="miControlador">
    <input type="text" ng-model="nom">
    <span>{{nom}}</span>
    <span ng-bind="nom">{{nom}}</span>
  </div>
</body>
```



ngBind (ng-bind)

Esta **directiva** tiene la **misma función** que las **llaves {{}}**, sin embargo, **ng-bind** tiene un **mejor funcionamiento** en cuanto a tiempo.

```
<body>
  <div ng-controller="miControlador">
    <input type="text" ng-model="nom">
    <span>{{nom}}</span>
    <span ng-bind="nom">{{nom}}</span>
  </div>
</body>
```

CREANDO NUESTRAS PROPIAS DIRECTIVAS

Muchas veces necesitamos construir nuestras **propias directivas** y por ello AngularJS nos facilita poder crearla a nuestro gusto a medida de nuestra necesidad. Mucho del código a continuación necesitará de conceptos posteriores por lo que animamos a recuperarlo en ese momento.

```
var app = angular.module('MiModulo', []);
app.controller('MiControlador', function($scope) {
  $scope.cliente = {
    nombre: 'Jhon',
    direccion: 'Av. Jose pardo 481'
  };
});
//Nuestra directiva
app.directive('miCliente', function() {
  return {
    template: 'Nombre: {{cliente.nombre}}
Dirección: {{cliente.direccion}}'
  };
});
```

En la parte de la vista usaríamos nuestra directiva así:

```
<body>
  <div ng-controller="MiControlador">
    <div mi-cliente></div>
  </div>
</body>
```

1.4 | Controllers (Controladores)

Los **controladores** en AngularJS son los **encargados** de **controlar** los **datos** de las **aplicaciones AngularJS**. Los **controladores** son **objetos** de **JavaScript**.

CONTROLADORES EN ANGULARJS

Las **aplicaciones AngularJS** son **controladas** por los **controladores**. Un **controlador** no es más que un **objeto JavaScript**, que se crea con el **constructor** de **objetos** de **JavaScript**. El **ámbito** de la **aplicación** está **definido** por **\$scope** y se **corresponde** con el **elemento HTML** **asociado** a la aplicación.

Como vemos en el ejemplo, la aplicación AngularJS está definida por la **directiva `ng-app`**. La aplicación **corre** dentro de **elemento `<div>`**. La **directiva `ng-controller`** indica el **nombre** del **objeto** correspondiente al controlador. La función **`cocheController`** es el **constructor estándar** del objeto JavaScript. El **objeto controlador** tiene una **propiedad: `$scope.coche`**. El objeto **coche** tiene dos propiedades: **marca** y **modelo**. La **directiva `ng-model`** enlaza los **campos `<input/>`** a las **propiedades** del controlador (marca y modelo).

```
<div ng-app="" ng-controller="cochesController">
  Marca: <input type="text" ng-model="coche.marca"><br>
  Modelo: <input type="text" ng-model="coche.
  modelo"><br>
  El Coche es: {{coche.marca + " " + coche.modelo}}
</div>
<script>
function cochesController($scope) {
  $scope.coche = {
    marca: "Audi",
    modelo: "A5"
  };
}
</script>
```

PROPIEDADES DEL CONTROLADOR

Un controlador también puede tener funciones como propiedades del controlador. Ejemplo:

```

<div ng-app="" ng-controller="cocheController">
  Marca: <input type="text" ng-model="coche.
  marca"><br>
  Modelo: <input type="text" ng-model="coche.
  modelo"><br>
  El coche es: {{coche.nombreCompleto()}}
</div>
<script>
function cocheController($scope) {
  $scope.coche = {
    marca: "Audi",
    modelo: "A3",
    nombreCompleto: function() {
      var x;
      x = $scope.coche;
      return x.marca + " " + x.modelo;
    }
  };
}
</script>

```

MÉTODOS DEL CONTROLADOR

Un controlador también puede tener métodos.

```

<div ng-app="" ng-controller="cocheController">
  Marca: <input type="text" ng-model="coche.
  marca"><br>
  Modelo: <input type="text" ng-model="coche.
  modelo"><br>
  El coche es: {{ nombreCompletoDelCoche() }}
</div>
<script>
function personController($scope) {
  $scope.coche = {
    marca: "Audi",
    modelo: "A3",
  };
  $scope.nombreCompletoDelCoche = function() {
    var x;
    x = $scope.coche;
    return x.marca + " " + x.modelo;
  };
}
</script>

```

CONTROLADORES EN FICHEROS EXTERNOS

En aplicaciones más grandes, es habitual tener los controladores en ficheros JavaScript externos. Solo debes copiar el código entre las etiquetas `<script>` en un fichero externo llamado `cocheController.js`.

```
<div ng-app="" ng-controller="cocheController">
Marca: <input type="text" ng-model="coche.marca"><br>
Modelo: <input type="text" ng-model="coche.
modelo"><br>
<br>
El Coche es: {{coche.marca + " " + coche.modelo}}
</div>
<script src="cocheController.js"></script>
```

1.5 | Módulos

Tus **aplicaciones** en AngularJS estarán **compuestas** por **módulos**. Todos los **controladores** de AngularJS deben **pertenecer** a un **módulo**. Con el **uso** de **módulos** **mantenemos** nuestro código **limpio** y bien **organizado**.

EJEMPLO DE MÓDULO EN ANGULARJS

En este ejemplo, "miAplicacion.js" contiene un **módulo** de definición de la aplicación "miAplicacion", "miControlador.js" **contiene** un **controlador**:

```
< div ng-app = "miAplicacion" ng-controller =
"miControlador" >
{{ nombre + " " + apellidos}}
</ div >
< s cript s rc= " //ajax.googleapis.com/ajax/libs/
angularjs/1.2.15/angular.min.js"> </s cript>
< script src = "miAplicacion.js" ></ script >
< script src = "miControlador.js" ></ script >
```

LOS CONTROLADORES ENSUCIAN EL ÁMBITO GLOBAL DE JAVASCRIPT

En todos los ejemplos que hemos ido viendo se han usado funciones Globales. Esto no está bien.

Las **variables globales** y las **funciones globales** no se deben usar en las aplicaciones de AngularJS, ya que podrán ser sobrescritos o eliminados por otros scripts.

Para corregir este problema, en AngularJS, hacemos uso de los módulos.

Usando un controlador simple:

```

<!DOCTYPE html>
< html >
< body >
< div ng-app = "" ng-controller = "miControlador" >
{{ nombre + " " + apellidos }}
</ div >
< script >
function miControlador ($scope) {
$scope.nombre = "Mario" ;
$scope.apellidos = "Flores" ;
}
</ script >
< script src = "//ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js" ></
script >
</ body >
</ html >

```

Usando un controlador perteneciente a un módulo:

```

<!DOCTYPE html>
< html >
< head >
< script src = "//ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js" ></
script >
</ head >
< body >
< div ng-app = "miAplicacion" ng-controller = "miControlador" >
{{ nombre + " " + apellidos }}
</ div >
< script >
var app = angular.module( "miAplicacion" , []);
app.controller( "miControlador" , function ($scope) {
$scope.nombre = "Mario";
$scope.apellidos = "Flores";
});
</ script >
</ body >
</ html >

```

DEFINICIÓN DE LOS MÓDULOS

Un consejo recomendado para las aplicaciones web, es colocar todos los `script` al final del documento HTML antes de cerrar la etiqueta `<body>`.

Esto provoca la página cargue mucho más rápido, ya que la carga del HTML no estará bloqueada por la carga de los scripts.

En muchas aplicaciones de *AngularJS* verás que la librería Angular es cargada en la sección `<head>` de la web.

En el ejemplo anterior, AngularJS es cargado en la sección `<head>`, lo hacemos así porque la llamada a `angular.module` sólo puede hacerse después de que haya cargado la librería Angular.

Como se podrá pensar, otra solución completamente válida, es cargar la librería de AngularJS en el `<body>`, antes de nuestros scripts de AngularJS.

```
<!DOCTYPE html>
< html >
< body >
< div ng-app = "miAplicacion" ng-controller = "miControlador" >
  {{ nombre + " " + apellidos }}
< / div >
< script src = "//ajax.googleapis.com/ajax/libs/angularjs/1.2.15/
angular.min.js" ></ script >
< script >
var app = angular.module( "miAplicacion" , []);
app.controller( "miControlador" , function ($scope) {
$scope.nombre = "Mario";
$scope.apellidos = "Flores";
});
< / script >
< / body >
< / html >
```



FICHEROS DE UNA APLICACIÓN ANGULARJS

Ahora que ya sabemos que son los módulos, y cómo funcionan, es hora de construir nuestros propios ficheros que compondrán la aplicación AngularJS.

Tu **aplicación** debe tener al menos un **archivo para el módulo** y **otro archivo** por cada **controlador**.

Primero creamos el script para el módulo, lo llamaremos *“miAplicacion.js”*:

```
var app = angular.module( "myApp" , []);
Luego creamos los controladores. En este caso
"miControlador.js":
app.controller( "miControlador" , function ($scope)
{
$scope.nombre= "Fernando";
$scope.apellidos= "Gil";
});
```

Por último, editamos la página HTML para cargar estos archivos:

```
<!DOCTYPE html>
< html >
< body >
< div ng-app = "myApp" ng-controller = "myCtrl" >
{{ firstName + " " + lastName }}
</ div >
< script src = "//ajax.googleapis.com/ajax/libs/
angularjs/1.2.15/angular.min.js" ></ script >
< script src = "myApp.js" ></ script >
< script src = "myCtrl.js" ></ script >
</ body >
</ html >
```



Telefonica

EDUCACIÓN DIGITAL