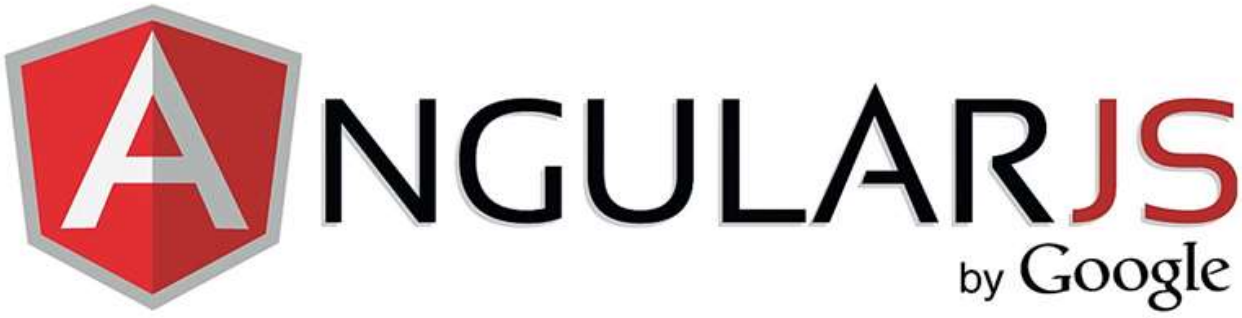
2015



CÓMO EMPEZAR

Índice de contenidos

¿Qué es AngularJS?

Sin extenderme mucho, AngularJS o simplemente Angular es un Framework creado por Google, está basado en el lenguaje JavaScript y con él podremos desarrollar aplicaciones web dinámicas de una manera más sencilla que para eso son los frameworks.

Lo importante de Angular es que su arquitectura está basada en el patrón de diseño MVC (Modelo, Vista, Controlador), permite inyección de dependecias como ya se verá más adelante y posee directivas HTML, que no son más que atributos de etiquetas HTML pero con unas funcionalidades específicas de Angular.

Angular se ha vuelto muy popular en los últimos años gracias a su eficiencia y a que permite coexistir con otros frameworks, como pueden ser JQuery, Bootstrap, etc. Y por qué no decirlo, si es de Google, ya se habrán encargado ellos de que tuviera un tirón más fuerte.

Para terminar con la presentación de Angular y comenzar con el manual o tutorial, decir, que lo que voy a comentar aquí es para iniciarse con Angular, no voy a explicar nada avanzado, para eso ya dejaré otra entrada, así como una entrada para seguir buenas prácticas con Angular y otra sobre cómo crear una aplicación base de Angular con Yeoman.

Lo primero

Lo primero, o al menos eso creo yo, es ver como añadir Angular a una aplicación Web. Bueno pues esto es muy sencillo, y no me voy a extender, si ya conoces JQuery y Bootstrap y has utilizado alguna CDN de estos frameworks, la manera es exactamente igual, aunque también es posible, evidentemente, descargarte el archivo físicamente y vincularlo a tu aplicación, al fin y al cabo, la instrucción es la misma, solo cambia la ruta. Así que tanto para descargarlo como para usar la CDN debemos ir al sitio oficial de Angular, y desde ahí ya podemos hacer lo que mejor nos convenga.

En el documento HTML de la aplicación web haremos lo siguiente:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Angulas JS: AngularJS con Bootstrap</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <!-- Con la siguiente línea AngularJS en nuestra aplicación Web mediante CDN -->
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>

        <!-- Con la siguiente línea AngularJS en nuestra aplicación Web mediante mediante el archivo físico -->
        <script src="libs/angular.1-4-8.min.js"></script>

    </head>
    <body>
    ...
    </body>
</html>
```

Como se ve en el ejemplo, he incluido Angular de dos maneras diferentes. En el contenido de la etiqueta HTML `<head>`, vemos dos etiquetas `<script>`, en la primera estamos usando la CDN que se ha obtenido de la página oficial. En la segunda he descargado Angular e incluido en la ruta o carpeta de mi aplicación `/libs`. Es evidente que las dos formas NO deben existir en una aplicación Angular, lo comento por si acaso.

A la hora de decidir cómo añadir Angular en la aplicación, veremos en el sitio oficial que da varias posibilidades, incluso se puede elegir la versión de este framework.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Angulas JS: AngularJS con Bootstrap</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <!-- Con la siguiente línea AngularJS en nuestra aplicación Web
mediante CDN -->
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min
.js"></script>
    <!-- Con la siguiente línea AngularJS en nuestra aplicación Web
mediante mediante el archivo físico -->
    <script src="libs/angular.1-4-8.min.js"></script>

    </head>
    <body>
    ...
    </body>
</html>
```

Expresiones en AngularJS

Se llaman expresiones a la manera o forma que tiene Angular de representar valores dentro de un documento o aplicación HTML. Es la forma en la que se puede mostrar dentro de esa aplicación o documento HTML, los resultados de los procesos que estamos realizando dentro de la aplicación web.

En el siguiente ejemplo se ve perfectamente lo que quiero decir.

Hacemos una salida de texto al Navegador

```
{{ 'Salida de texto al Navegador' }}
```

Hacemos un cálculo y una salida de texto al Navegador con formato HTML

```
<strong id="cosa-loca">{{ 10 + 10 }}</strong> Una vez hecho el calculo
hacemos la salida
```

Muy sencillo.

Filtros con AngularJS

Con Angular podemos dar formato a la información que queremos representar mediante los filtros de las expresiones. Por ejemplo, si quiero que un número tenga el formato moneda sin tener que hacer cualquier tipo de transformación, basta con usar un filtro de AngularJS.

Hay muchos tipos de filtros, éstos permiten hacer muchas cosas, pero lo más habitual es utilizarlos para modificar los valores a presentar o aplicarles formato como ya he dicho antes. En Angular, tenemos filtros nativos y también la posibilidad de desarrollar nuestros propios filtros según nuestras necesidades.

En el ejemplo siguiente muestro las sintaxis de cómo usar algunos de esos filtros nativos pero también dejo un enlace para poder ver la lista de filtros nativos de Angular en la documentación oficial.

Sintaxis básica

```
{{ expresión | filtro }}

{{ expresión | filtro1 | filtro2 | ... }}

{{ expresión | filtro:argumento1:argumento2:... }}
```

Filtro de moneda

```
{{ 309 | currency }} (por defecto en dólares)

{{ 399 | currency:"€" }} (modificación de moneda)
```

Filtro numérico

```
{{ 1235467890 | number:2 }}
```

Formato de texto, mayúsculas y minúsculas

```
{{ 'Aplicando filtro uppercase' | uppercase }}

{{ 'APLICANDO FILTRO LOWERCASE' | lowercase }}
```

No voy a comentar nada más sobre esto, creo que resulta bastante fácil de entender. Como dije al principio de esta entrada, no voy a comentar ni explicar nada más avanzado. En otra entrada que quiero escribir sobre Angular más avanzado (a ver cuándo lo hago), ya comentaré como hacer un filtro personalizado.

Directivas nativas

Todas las directivas nativas de AngularJS comienza por el prefijo **ng**. Estas directivas tienen un incoveniente y es que la validación del código HTML arrojará errores. Estos validadores entienden que estas directivas son atributos de la etiqueta en cuestión y no son válidos en el estándar HTML.
Para ello podemos poner delante del prefijo [**ng-**] otro prefijo, [**data-**], ya que los atributos que comiencen por este prefijo si están reconocidos por el estándar HTML.

ng-app

```
<!-- La validación HTML mostraría mensajes de error -->
<html ng-app>
    <body>
        ... tu contenido aquí ...
    </body>
</html>
```


```
<!-- La validación HTML mostraría mensajes de error -->
<html ng-app>
    <body>
        ... tu contenido aquí ...
    </body>
</html>
```


```
<!-- Dentro del estándar HTML -->
<html data-ng-app>
    <body>
        ... tu contenido aquí ...
    </body>
</html>
```


```
<!-- Dentro del estándar HTML -->
<html data-ng-app>
    <body>
        ... tu contenido aquí ...
    </body>
</html>
```


También es posible definir la directiva de la aplicación Angular en la etiqueta Body de HTML

```
<html>
    <!-- La validación HTML mostraría mensajes de error -->
    <body ng-app>
        ... tu contenido aquí ...
    </body>
</html>
```

```
<html>
    <!-- La validación HTML mostraría mensajes de error -->
    <body ng-app>
        ... tu contenido aquí ...
    </body>
</html>
```

```
<html>
    <!-- Dentro del estándar HTML -->
    <body data-ng-app>
        ... tu contenido aquí ...
    </body>
</html>
```

```
<html>
    <!-- Dentro del estándar HTML -->
    <body data-ng-app>
        ... tu contenido aquí ...
    </body>
</html>
```

La directiva **ng-app** es la que arranca o ejecuta la aplicación de AngularJS estableciendo el elemento raíz de la misma.
Vemos en el ejemplo, que podemos usar la directiva nativa **ng-app** tanto en la etiqueta HTML como BODY, aunque puede usarse en cualquier otra etiqueta HTML pero debemos tener en cuenta que el ámbito de la aplicación de AngularJS se localizaría dentro de esa etiqueta.

ng-controller

```
<html data-ng-app>
    <body data-ng-controller="controlador">
        ... tu contenido aquí ...
    </body>
</html>
```

```
<html data-ng-app>
    <body data-ng-controller="controlador">
        ... tu contenido aquí ...
    </body>
</html>
```

La directiva **ng-controller** es una directiva nativa de AngularJS que asocia el controlador que gestiona los eventos de control a la vista.

ng-model

```
<html data-ng-app>
    <body data-ng-controller="controlador">
        <label data-ng-model="nombreEtiqueta"></label>
    </body>
</html>
```


```
<html data-ng-app>
    <body data-ng-controller="controlador">
        <label data-ng-model="nombreEtiqueta"></label>
    </body>
</html>
```


La directiva **ng-model** es una directiva nativa que vincula una propiedad en concreto del modelo declarado en el controlador a un componente de la vista. Lo más lógico es agregarlo a un objeto de formulario para obtener información.

ng-show
Se pueden definir directivas dentro de otras directivas como ya hemos visto.

```
<html data-ng-app>
    <body data-ng-controller="controlador">
        <label data-ng-model="nombreEtiqueta"></label>
        <div data-ng-show="true">
            El contenido de este DIV es visible
        </div>
    </body>
</html>
```


```
<html data-ng-app>
    <body data-ng-controller="controlador">
        <label data-ng-model="nombreEtiqueta"></label>
        <div data-ng-show="true">
            El contenido de este DIV es visible
        </div>
    </body>
</html>
```


La directiva **ng-show** muestra u oculta el elemento HTML determinado en base a la expresión proporcionada al dicho atributo o directiva.

Para ver las directivas nativas de Angular JS, dejo este enlace a la documentación oficial.

Enlace a datos (Data Binding)

Los modelos son elementos donde almacenamos la información de una aplicación. Estos pueden contener muchos tipos de datos, desde complejas bases de datos a pequeños datos individuales. Los modelos de Angular JS tienen la capacidad de generar los llamados **Data Binding** o **enlace a datos**, características que nos permite unir en tiempo real los datos de un modelo dentro de una expresión o una variable.

```
<!DOCTYPE html>
<html data-ng-app>
    <head>
        <title>Angulas JS: Enlace a datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <label>Nombre:</label>
        <input type="text" data-ng-model="nombre">
        <div>
            <h1>Mi nombre es:</h1>
            <h2>{{ nombre }}</h2>
        </div>
    </body>
</html>
```


```
<!DOCTYPE html>
<html data-ng-app>
    <head>
        <title>Angulas JS: Enlace a datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <label>Nombre:</label>
        <input type="text" data-ng-model="nombre">
        <div>
            <h1>Mi nombre es:</h1>
            <h2>{{ nombre }}</h2>
        </div>
    </body>
</html>
```

Controladores

AngularJS utiliza el patrón de diseño MVC (Modelo, Vista, Controlador) para el desarrollo de aplicaciones.
Para incluir un controlador en un documento HTML debemos seguir una serie de pasos que son necesarios para que AngularJS tenga acceso al dicho controlador.
Vamos a ver como incluir un controlador y utilizarlo para procesar la información dentro de una aplicación.
Lo primero es definir el nombre de la aplicación, para eso, usamos la directiva nativa **ng-app**, pero si queremos, como comenté anteriormente, que los validadores HTML que existen en la Red no nos den problemas debemos usar el prefijo **data-**, con lo que la directiva pasará a llamarse **data-ng-app**.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Controladores</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h1>Promoció Loca</h1>

            <label>Precio Loco 4: 2.5</label><br>
            <label>Cantidad Loca <input type="text" data-ng-model="vm.cantidad"></label>

            <h2>Total: {{ vm.precio * vm.cantidad | currency:'€'}}</h2>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco);

                function controladorLoco(){
                    var vm = this;
                    vm.precio = 2.5;
                    vm.cantidad = 1;
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Controladores</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h1>Promoció Loca</h1>
```

```
            <label>Precio Loco 4: 2.5</label><br>
            <label>Cantidad Loca <input type="text" data-ng-
model="vm.cantidad"></label>

            <h2>Total: {{ vm.precio * vm.cantidad |
currency:'€'}}</h2>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco);

                function controladorLoco(){
                    var vm = this;
                    vm.precio = 2.5;
                    vm.cantidad = 1;
                }
            })();
        </script>
    </body>
</html>
```

En el ejemplo simplemente vemos cómo se calcula el precio total a partir de un precio unitario y una cantidad. Para ello he definido un variable o atributo del **Scope**, al final comentaré qué es esto, del **controladorLoco**, **vm.precio** y **vm.cantidad** como cantidad inicial. Mediante una entrada de texto HTML a la que he asignado al atributo o directiva nativa **ng-model** con el valor del atributo del Scope, vm.cantidad calculamos el precio total.

En cuanto al **Scope** antes comentado, es un objeto que hace referencia al modelo de la aplicación. Mi definición sería como si fuera un DOM de HTML pero resumido al contexto del controlador que ha sido asignado a un elemento HTML específico. El Scope puede tener expresiones y propagar eventos.

Uso de múltiples controladores

Una aplicación desarrollada en AngularJS puede tener un número ilimitado de controladores. En este paso vamos a ver cómo trabaja independientemente cada uno de ellos y también cómo se relacionan entre sí.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Uso de múltiples Controladores</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorGatoLoco as vm">
            <h3>[controladorGatoLoco] Una gata loca: <u>{{ vm.nombre }}</u></h3>
        </div>
        <div data-ng-controller="controladorGatoLoco as vm">
            <h3>[controladorGatoLoco] La misma gata loca utilizando el mismo controlador loco en otro loco elemento HTML: <u>{{ vm.nombre }}</u></h3>
        </div>
        <div data-ng-controller="controladorPerroLoco as vm">
            <h3>[controladorPerroLoco] Perro loco: <u>{{ vm.nombre }}</u></h3>
        </div>
        <script type="text/javascript">
            'use strict';
            angular
                .module('aplicacionLoca', [])
                .controller('controladorGatoLoco', controladorGatoLoco);
                .controller('controladorPerroLoco', controladorPerroLoco);

            function controladorGatoLoco(){
                var vm = this;
                vm.nombre = 'Amparo';
            }
            function controladorPerroLoco(){
                var vm = this;
                vm.nombre = 'Curco';
            }
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Uso de múltiples Controladores</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorGatoLoco as vm">
            <h3>[controladorGatoLoco] Una gata loca: <u>{{ vm.nombre
}}</u></h3>
        </div>
        <div data-ng-controller="controladorGatoLoco as vm">
            <h3>[controladorGatoLoco] La misma gata loca utilizando el
mismo controlador loco en otro loco elemento HTML: <u>{{ vm.nombre
}}</u></h3>
        </div>
        <div data-ng-controller="controladorPerroLoco as vm">
```

```
            <h3>[controladorPerroLoco] Perro loco: <u>{{ vm.nombre
}}</u></h3>
        </div>
        <script type="text/javascript">
            'use strict';
            angular
                .module('aplicacionLoca', [])
                .controller('controladorGatoLoco',
controladorGatoLoco);
                .controller('controladorPerroLoco',
controladorPerroLoco);

            function controladorGatoLoco(){
                var vm = this;
                vm.nombre = 'Amparo';
            }
            function controladorPerroLoco(){
                var vm = this;
                vm.nombre = 'Curco';
            }
        </script>
    </body>
</html>
```

Los controladores pueden ser reciclados en una aplicación de AngularJS por lo que nos podemos encontrar que un controlador afecte muchas veces a varios elementos dentro de un documento HTML.
En el ejemplo podemos ver que el controlador, controladorGatoLoco, se usa un par de veces en dos elementos HTML diferentes, y si cambiamos el valor del atributo nombre, vemos cómo afecta a esos dos elementos.

Métodos de un Controlador

Los controladores de AngularJS son elementos diseñados para realizar unas funciones específicas, puede almacenar valores, generalmente incluyen procesos y cálculos internos. Los métodos son funciones internas que viven dentro del ámbito de su controlador y pueden realizar acciones o cálculos muchos más complejos, tan complejos como la situación lo necesite.
En este apartado vamos a ver cómo declarar y utilizar los métodos de un controlador en un ejemplo que no tiene nada de loco pero si es muy sencillo. Vamos a calcular la circunferencia de un círculo al hacer clic en un botón.
¿No es muy loco verdad?

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Métodos de un Controlador</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <label>Radio Loco</label><br>
            <input type="text" data-ng-model="vm.radio">
```

```
            <button ng-click="vm.calcularCircunferencia()">Calcular
circunferencia loca</button>
            <h3>Resultado: {{ vm.circunferencia | number:8 }}</h3>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco);

                function controladorLoco(){
                    var vm = this;
                    var pi = 3.1416;

                    vm.calcularCircunferencia = function(){
                        vm.circunferencia = 2 * pi * vm.radio;
                    };
                }
            })();
        </script>
    </body>
</html>
```

Aunque en el ejemplo, nada loco, estamos utilizando un único método dentro del controlador, cualquier controlador puede contener una cantidad indefinida de métodos.
Estos métodos pueden estar relacionados siempre y cuando los valores estén dentro del controlador o estén relacionados de alguna forma con el ámbito o el área de alcance para poder acceder a dichos valores.

Colecciones de datos

Una aplicación desarrollada con AngularJS, normalmente, se suelen usar colecciones de datos o información tales como Arrays u Objetos.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Colecciones de datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor }}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorMasLoco as vm">
            <h3>Nombre de Actores aún más loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor.nombre }} {{ actor.apellidos }}</li>
            </ul>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco);

                function controladorLoco(){
                    var vm = this;

                    vm.actores = ['Robert de Niro', 'Harrison Ford', 'Sean Connery'];
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Colecciones de datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor
}}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorMasLoco as vm">
            <h3>Nombre de Actores aún más loco</h3>
            <ul>
```

```
                <li data-ng-repeat="actor in vm.actores">{{
actor.nombre }} {{ actor.apellidos }}</li>
            </ul>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco);

                function controladorLoco(){
                    var vm = this;

                    vm.actores = ['Robert de Niro', 'Harrison Ford',
'Sean Connery'];
                }
            })();
        </script>
    </body>
</html>
```

Como vemos en el ejemplo, algo más loco que el anterior, tenemos una nueva directiva nativa de AngularJS, **ng-repeat**, repito que coloco el prefio **data-** delante de la directiva para no llevarnos una sorpresita con los validadores HTML.
Esta directiva permite iterar una colección de datos, ahora la vemos aplicada al elemento **
** del documento HTML pero podemos asignarla otros elementos que nos interese según las necesidades.
En el ejemplo también vemos que se ha iterado dos colecciones, que aunque ambas sean dos Arrays, el contenido de éstas son distintos. En el primero tenemos una colección de Cadenas o Strings y en el segundo vemos que tenemos una colección de objetos y cada uno con atributos Nombre y Apellidos, la manera de acceder a los valores de ambas colecciones son distintas.

Filtrado de datos

Cuando disponemos de una gran cantidad de datos es muy posible que necesitemos agruparlos o filtrarlos para analizar únicamente los datos que necesitamos.
Con AngularJS veremos que estos procesos de búsqueda y filtrado de datos sea muy sencillo.
Así que vamos a hacer un ejemplo muy loco que nos permita buscar estos datos.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Filtrado de datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor }}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorMonoLoco as vm">
            <h3>Más que un mono loco</h3>
            <label>Buscar</label>
            <input type="search" data-ng-model="vm.busqueda">
            <ul>
                <li data-ng-repeat="mono in vm.monoslocos | filter: vm.busqueda">{{ mono.nombre }}, destaca por {{ mono.destaca }}</li>
            </ul>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorMonoLoco', controladorMonoLoco)

                function controladorMonoLoco(){
                    var vm = this;
                    vm.monoslocos = [
                        {nombre: 'King Kong', destaca: 'Cargarse un Tiranosaurio'},
                        {nombre: 'Abu', destaca: 'Tener siempre hambre'},
                        {nombre: 'Cheeta', destaca: 'Ser la parienta de Tarzán'},
                        {nombre: 'Mr. Teeny', destaca: 'Fumón'},
                        {nombre: 'Amedio', destaca: 'Bujarra seguro'},
                        {nombre: 'Maguila', destaca: 'Gorila'},
                        {nombre: 'El mono malvado del armario', destaca: 'Tirarte caca de mono'},
                        {nombre: 'Gunter', destaca: '... Más que un mono loco'},
                    ];
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Filtrado de datos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
```

```
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor
}}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorMonoLoco as vm">
            <h3>Más que un mono loco</h3>
            <label>Buscar</label>
            <input type="search" data-ng-model="vm.busqueda">
            <ul>
                <li data-ng-repeat="mono in vm.monoslocos | filter:
vm.busqueda">{{ mono.nombre }}, destaca por {{ mono.destaca }}</li>
            </ul>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorMonoLoco',
controladorMonoLoco)

                function controladorMonoLoco(){
                    var vm = this;
                    vm.monoslocos = [
                        {nombre: 'King Kong', destaca: 'Cargarse un
Tiranosaurio'},
                        {nombre: 'Abu', destaca: 'Tener siempre
hambre'},
                        {nombre: 'Cheeta', destaca: 'Ser la parienta
de Tarzán'},
                        {nombre: 'Mr. Teeny', destaca: 'Fumón'},
                        {nombre: 'Amedio', destaca: 'Bujarra seguro'},
                        {nombre: 'Maguila', destaca: 'Gorila'},
                        {nombre: 'El mono malvado del armario',
destaca: 'Tirarte caca de mono'},
                        {nombre: 'Gunter', destaca: '... Más que un
mono loco'},
                    ];
                }
            })();
        </script>
    </body>
</html>
```

En el ejemplo, lleno de monos locos, vemos un nuevo filtro nativo de AngularJS, **filter**, al que le pasamos el argumento de **ng-model** del elemento input HTML, **vm.busqueda**. Este filtro selecciona un subconjunto de los elementos de una colección y devuelve el resultado en una nueva colección. En la documentación oficial de AngularJS, se refiere a la colección de filtra como Array. El filtrado también se hace dentro del parámetros destaca de los objetos de la colección.
Esta es una forma muy sencilla de realizar procesos de búsqueda muy rápidos que nos brinda AngularJS
La directiva, ng-repeat, la tenemos aplicada al elemento **
** del documento HTML pero repito, podemos asignarla otros elementos que nos interese según las necesidades por ejemplo podemos cambiarlo un el elemento ** **.

Eventos

Una aplicación se desarrolla para facilitar y automatizar tareas complejas, o no tan complejas, que debe desarrollar un individuo. Así, una aplicación, tiene que interactuar con dicho individuo o Usuario.
Las diferentes acciones que pueden realizar este usuario están programadas en la aplicación mediante los denominados eventos. AngularJS, por supuesto, también tiene elementos que escuchan estos eventos y adaptan la aplicación a lo que está ocurriendo en ese momento.
Vamos a hacer que la aplicación de AngularJS del ejemplo que dejo a continuación, escuche esos eventos.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Eventos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor }}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Unos eventos muy locos</h3>
            <button data-ng-click="vm.clicSimpleBoton()">A ver si te atreves a hacer clic!!!</button><br>
            <a data-ng-dblclick="vm.clicDobleBoton()" style="cursor:pointer;text-decoration: underline;">A ver si te atreves dos veces!!!</a><br>
            <img src="images/hola-que-hase.jpg" data-ng-mouseleave="vm.quitateDeEncima()" data-ng-mouseenter="vm.eresPesadito()"><br>
            <h3 style="color:red;">{{ vm.mensaje }}</h3>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;

                    vm.clicSimpleBoton = function(){
                        vm.mensaje = "Pues sí, te has atrevido...";
                    };
                    vm.clicDobleBoton = function(){
                        vm.mensaje = "Pues sí, te has atrevido... y encima dos veces!!!";
                    };
                    vm.quitateDeEncima = function(){
                        vm.mensaje = "Por fin me has dejado en paz";
                    };
                    vm.eresPesadito = function(){
                        vm.mensaje = "Quitate de encima miaaa!!!";
                    };
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Eventos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Nombre de Actores a loco</h3>
            <ul>
                <li data-ng-repeat="actor in vm.actores">{{ actor
}}</li>
            </ul>
        </div>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Unos eventos muy locos</h3>
            <button data-ng-click="vm.clicSimpleBoton()">A ver si te
atreves a hacer clic!!!</button><br>
            <a data-ng-dblclick="vm.clicDobleBoton()"
style="cursor:pointer;text-decoration: underline;">A ver si te atreves
dos veces!!!</a><br>
            <img src="images/hola-que-hase.jpg" data-ng-
mouseleave="vm.quitateDeEncima()" data-ng-
mouseenter="vm.eresPesadito()"><br>
            <h3 style="color:red;">{{ vm.mensaje }}</h3>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;

                    vm.clicSimpleBoton = function(){
                        vm.mensaje = "Pues sí, te has atrevido...";
                    };
                    vm.clicDobleBoton = function(){
                        vm.mensaje = "Pues sí, te has atrevido... y
encima dos veces!!!";
                    };
                    vm.quitateDeEncima = function(){
                        vm.mensaje = "Por fin me has dejado en paz";
                    };
                    vm.eresPesadito = function(){
                        vm.mensaje = "Quítate de encima miaaa!!!";
                    };
                }
            })();
        </script>
    </body>
</html>
```

Como vemos en el ejemplo, los eventos se definen como atributos de los elementos HTML, casi todos los elementos HTML es posible asignarle estos eventos.
Aquí únicamente he querido mostrar las directivas de los eventos más comunes como son, el evento **ng-click** que se lanza cuando hacemos clic con el botón derecho del ratón sobre el elemento en cuestión, doble **ng-dblclick**, lanzado cuando hacemos doble clic sobre un elemento, **ng-mouseleave** que se lanza justamente cuando colocamos el punto del ratón sobre

el elemento y **ng-mouseenter**, lanzado justo quitamos el punto del ratón de encima de un elemento.

Estilos CSS

Al ir mejorando las aplicaciones vamos a necesitar modificar la apariencia de las mismas, y con qué si no la vamos a modificar si no es con las hojas de estilos y su código CSS. Con AngularJS tiene control sobre elementos gráficos, por lo que vamos a poder cambiar o modificar las propiedades o estilos dinámicamente.
En el ejemplo que he desarrollado, vemos que podemos cambiar tanto las propiedades de un elemento HTML, como un estilo completo del mismo u otro elemento al cambiar la clase.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Estilos CSS</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>

        <style>
            div#container{ width: 600px; margin: 10px auto; text-align: center; }
            div#container .result{ border: 2px #000 solid; }
            .culo-de-mono{ background: #FFF; font-size:36px; padding: 0 40px; color: chocolate; }
        </style>
    </head>
    <body>
        <div id="container" data-ng-controller="controladorLoco as vm">
            <div class="result">
                <p data-ng-style="vm.estilos">Controlamos el estilo del <span class="{{ vm.claseDinamica }}">un mono</span> con AngularJS</p>
            </div>

            <br><button data-ng-click="vm.changeClass()">Cambiar clases</button>
            <button data-ng-click="vm.changeProperties()">Cambiar propiedades</button>
            <button data-ng-click="vm.resetStyles()">Reiniciar estilos</button>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;

                    vm.changeClass = function(){
                        vm.claseDinamica = 'culo-de-mono';
                    };
                    vm.changeProperties = function(){
                        vm.estilos = {'background':'#222', 'color':'#F0F0F0'};
                    };
                    vm.resetStyles = function(){
                        vm.estilos = '';
                        vm.claseDinamica = '';
                    };
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Estilos CSS</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
```

```
        <style>
            div#container{ width: 600px; margin: 10px auto; text-
align: center; }
            div#container .result{ border: 2px #000 solid; }
            .culo-de-mono{ background: #FFF; font-size:36px; padding:
0 40px; color: chocolate; }
        </style>
    </head>
    <body>
        <div id="container" data-ng-controller="controladorLoco as
vm">
            <div class="result">
                <p data-ng-style="vm.estilos">Controlamos el estilo
del <span class="{{ vm.claseDinamica }}">un mono</span> con
AngularJS</p>
            </div>

            <br><button data-ng-click="vm.changeClass()">Cambiar
clases</button>
            <button data-ng-click="vm.changeProperties()">Cambiar
propiedades</button>
            <button data-ng-click="vm.resetStyles()">Reiniciar
estilos</button>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;

                    vm.changeClass = function(){
                        vm.claseDinamica = 'culo-de-mono';
                    };
                    vm.changeProperties = function(){
                        vm.estilos = {'background':'#222',
'color':'#F0F0F0'};
                    };
                    vm.resetStyles = function(){
                        vm.estilos = '';
                        vm.claseDinamica = '';
                    };
                }
            })();
        </script>
    </body>
</html>
```

Comentando el ejemplo y para destacar algo, vemos que cambiar la clase no es nada complicado pero para modificar unas propiedades ya vemos que hemos usado una nueva directiva que es **ng-style**. Esta directiva nos permite asignar un estilo CSS de manera condicional a un elemento HTML.
Vemos cómo con el método **chageProperties()** asignamos los valores CSS en forma de objeto a la propiedad del **Scope vm.estilos** y ésta a su vez a la directiva ng-style del elemento HTML.

Mostrar y Ocultar elementos

En muchas ocasiones tenemos la necesidad de activar y desactivar o, mejor dicho, mostrar y ocultar elementos dependiendo de cómo interactúe una aplicación con su usuario y la acción que éste lleve a cabo. Con AngularJS veremos que esto es muy sencillo de desarrollar con sólo unas líneas de código.
En mi más modesta opinión esto es muy fácil en todos los lenguajes o en la mayoría, al menos, en todos los que yo conozco.
En el ejemplo siguiente vemos cómo se muestra y oculta el elemento HTML con id, **message**, mediante tres botones.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Mostrar y Ocultar elementos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min.js"></script>
        <style>
            div#message{ width:200px; height: 70px; background: #222; }
            div#message > p{font-size: 22px; padding: 20px; text-align: center; color:#FFF;}
        </style>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Mostrar y ocultamos las cosas locas</h3>
            <button data-ng-click="vm.mostrar()">Mostrar</button>
            <button data-ng-click="vm.ocultar()">Ocultar</button>
            <button data-ng-click="vm.mostrarOcultar()">Mostrar/Ocultar tó junto!!!</button>
            <div id="message" data-ng-show="vm.showHideStatus">
                <p>Caca de mono</p>
            <div>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;
                    vm.showHideStatus = false;

                    vm.mostrar = function(){
                        vm.showHideStatus = true;
                    };
                    vm.ocultar = function(){
                        vm.showHideStatus = false;
                    };
                    vm.mostrarOcultar = function(){
                        vm.showHideStatus = vm.showHideStatus ? false : true;
                    };
                }
            })();
        </script>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Mostrar y Ocultar elementos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
        <style>
            div#message{ width:200px; height: 70px; background: #222;
}
            div#message > p{font-size: 22px; padding: 20px; text-
align: center; color:#FFF;}
        </style>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Mostrar y ocultamos las cosas locas</h3>
            <button data-ng-click="vm.mostrar()">Mostrar</button>
            <button data-ng-click="vm.ocultar()">Ocultar</button>
            <button data-ng-
click="vm.mostrarOcultar()">Mostrar/Ocultar tó junto!!!</button>
            <div id="message" data-ng-show="vm.showHideStatus">
                <p>Caca de mono</p>
            <div>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;
                    vm.showHideStatus = false;

                    vm.mostrar = function(){
                        vm.showHideStatus = true;
                    };
                    vm.ocultar = function(){
                        vm.showHideStatus = false;
                    };
                    vm.mostrarOcultar = function(){
                        vm.showHideStatus = vm.showHideStatus ? false
: true;
                    };
                }
            })();
        </script>
    </body>
</html>
```

Comentando el desarrollo del ejemplo, como se puede observar, he asignado un valor booleano al atributo o directiva nativa de AngularJS, **ng-show** (con el prefijo **data-** delante, no me voy a cansar de decirlo), mediante **vm.showHideStatus**, inicializado a **False**, es decir, que el estado natural del mensaje es oculto.

Tenemos tres métodos: **mostrar()**,**ocultar()** y **mostrarOcultar()**. Cada método asignado a un elemento botón HTML gracias al evento **ng-click**. Con el primer método, mostramos el mensaje simplemente cambiando el valor del atributo o variable del Scope, esto lo comentaré más adelante, del controladorLoco, que hemos declarado como **vm.showHideStatus** a un valor **True**. Con el segundo método, hacemos exactamente lo mismo, pero el valor esta vez lo asignamos a **False**. La funcionalidad del tercer método asignado al botón Mostrar/Ocultar, es exactamente igual que la del método mostrar y la del método ocultar pero unido todo en una sola línea de código, más limpio y más optimizado, mediante una comparación ternaria del valor de vm.showHideStatus. Si el valor esta variable es True, lo ponemos a False, si no, le asignamos True.
La funcionalidad de una comparación ternaria es igual que la de un **if** y pueden, igualmente, ser anidados, aunque mi consejo particular es que no se hagan más de tres niveles de profundidad.

Cambiar imágenes

Cuando necesitamos intercambiar imágenes, vamos a encontrar que AngularJS tiene una simple técnica para que podamos administrar dinámicamente dichas imágenes. En el ejemplo que muestro a continuación voy a hacer que un elemento imagen de HTML cambie su origen, es decir, voy a cambiar la imagen mediante un par de botones mediante el evento **ng-click** y también voy a usar los eventos **ng-mouseleave** y **ng-mouseenter**.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Cambiar imágenes</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.min
.js"></script>
    </head>
    <body>
        <div data-ng-controller="controladorLoco as vm">
            <h3>Intercambiamos imágenes de cosas locas</h3>
            <img src="images/{{ vm.imagenDefecto }}" data-ng-
mouseleave="vm.cambiar('hola-que-hase.jpg')" data-ng-
mouseenter="vm.cambiar('angular-o-que-hase.jpg')"><br>
            <button data-ng-click="vm.cambiar('hola-que-
hase.jpg')">Cosa loca 1</button>
            <button data-ng-click="vm.cambiar('angular-o-que-
hase.jpg')">Cosa loca 2</button>
        </div>
        <script type="text/javascript">
            (function() {
                'use strict';
                angular
                    .module('aplicacionLoca', [])
```

```
                    .controller('controladorLoco', controladorLoco)

                function controladorLoco(){
                    var vm = this;
                    vm.imagenDefecto = 'hola-que-hase.jpg';

                    vm.cambiar = function( img ){
                        vm.imagenDefecto = img;
                    };
                }
            })();
        </script>
    </body>
</html>
```

Vemos cómo la función, o mejor dicho, el método **cambiar()** del Scope del **controladorLoco** recibe un parámetro, **img**, que no es más que el nombre del archivo de la imagen y mediante los eventos click, mouseleave y mouseenter, cada uno con el método asignado y el parámetro de la nueva imagen, cambiamos ésta en el documento HTML en muy pocos pasos.

Incluir archivos externos

Yo como desarrollador, siempre estoy intentando reciclar partes de códigos o archivos, bien para no repetir el mismo código en varias localizaciones o bien para reutilizar dichos códigos ya desarrollado, lo que viene llamándose diseño modular. Para esto, AngularJS provee la manera de incluir documentos externos dentro de nuestras aplicaciones para crear estos diseños modulares y plantillas.
En el ejemplo siguiente vemos cómo he incluído tres archivos: **header.html**, **menu.php** y **footer.html**; algo muy típico cuando estamos desarrollando, por ejemplo un sitio web.

```
<!DOCTYPE html>
<html data-ng-app>
    <head>
        <title>Angulas JS: Incluir archivos externos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min
.js"></script>
        <script src="https://code.angularjs.org/1.4.8/angular-
route.min.js"></script>
        <link rel="stylesheet" href="styles/estilos.min.css">
    </head>
    <body>
        <header>
            <p>Aquí incluimos el Header</p>
            <div data-ng-include src="'plantillas/header.html'">
</div>
        </header>
        <nav>
            <p>Aquí un menu de ejemplo</p>
            <ng-include src="'plantillas/menu.php'"></ng-include>\n\
```

```
        </nav>
        <div class="body-container">

        [...]

        </div>
        <footer>
            <p>Y aquí un pie de página</p>
            <div data-ng-include="'plantillas/footer.html'"> </div>
        </footer>
    </body>
</html>
```

Me gustaría remarcar, dos cosas en este ejemplo.
La primera es que las inclusiones de archivos pueden tener más de una sintaxis y que detallo a continuación:

```
1 <div data-ng-include src="'documento.html'"> </div>
2 <div data-ng-include="'documento.html'"> </div>
3 <ng-include src="'documento.php'"></ng-include>
```

```
<div data-ng-include src="'documento.html'"> </div>
<div data-ng-include="'documento.html'"> </div>
<ng-include src="'documento.php'"></ng-include>
```

Es importante que dentro de las comillas dobles la ruta del documento o archivo, debe estar a su vez, incluida dentro de comillas simples.
Por otra parte, la hoja de estilos debería estar en el Layout, es decir, la página que contiene los documentos o archivos incluidos.
Y como soy muy pesado, vuelvo a decir que yo coloco el prefijo **data-** delante de la directiva **ng-** para evitar problemas con los validadores HTML.

Enrutamiento (Routing)

Para seguir las buenas prácticas con AngularJS, aunque no lo haga al pie de la letra en este tutorial, voy a dividir este ejemplo en varios archivos y más adelante dedicaré una entrada para explicar estas buenas prácticas a la hora de desarrollar con AngularJS.
Una de las funcionalidades o técnicas más interesantes, entre otras, de AngularJS es el enrutamiento de las direcciones URL. Esta funcionalidad nos permite ejecutar todo el código de una aplicación en un solo archivo HTML, cargando asincrónicamente el contenido y reduciendo los tiempos de respuesta de dicha aplicación. A diferencias de un simple Ajax, AngularJS, gestiona la ruta del documento y permite que los contenidos sean indexados individualmente por los motores de búsqueda.
En el ejemplo, todo loco, que he hecho a continuación se ve perfectamente el método de enrutamiento que utilizo, cargando así, los diferentes módulos o documentos HTML en el mismo layout.

INDEX.HTML

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Incluir archivos externos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
        <script src="https://code.angularjs.org/1.4.8/angular-route.min.js"></script>
        <script src="app/app.js"></script>
        <script src="app/router.js"></script>
        <style>
            body{ font-family: 'arial'; }
            img{ width:400px; }
            section ul,li{ margin:0; padding: 0; }
            section li{ list-style-type: none; background: #333; float: left; height: 30px; width: 25%; text-align: center; }
            section li a{ color: #fff; text-decoration: none; line-height: 30px; }
            section .menu{}
            section .menu li{ width: 100px; background: #666; }
        </style>
    </head>
    <body>
        <header>
            <div data-ng-include src="'plantillas/header.html'"> </div>
        </header>

        <section>
            <div data-ng-view>
            </div>

            <ul class="menu">
                <li><a href="#/animales/monos">Monos</a></li>
                <li><a href="#/animales/gatos">Gatos</a></li>
                <li><a href="#/animales/perros">Perros</a></li>
            </ul>
        </section>

        <footer>
            <div data-ng-include="'plantillas/footer.html'"> </div>
        </footer>
    </body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
    <head>
        <title>Angulas JS: Incluir archivos externos</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min
.js"></script>
        <script src="https://code.angularjs.org/1.4.8/angular-
route.min.js"></script>
        <script src="app/app.js"></script>
        <script src="app/router.js"></script>
        <style>
            body{ font-family: 'arial'; }
            img{ width:400px; }
            section ul,li{ margin:0; padding: 0; }
            section li{ list-style-type: none; background: #333;
float: left; height: 30px; width: 25%; text-align: center; }
            section li a{ color: #fff; text-decoration: none; line-
height: 30px; }
            section .menu{}
            section .menu li{ width: 100px; background: #666; }
        </style>
    </head>
    <body>
        <header>
            <div data-ng-include src="'plantillas/header.html'">
</div>
        </header>

        <section>
            <div data-ng-view>
            </div>

            <ul class="menu">
                <li><a href="#/animales/monos">Monos</a></li>
                <li><a href="#/animales/gatos">Gatos</a></li>
                <li><a href="#/animales/perros">Perros</a></li>
            </ul>
        </section>

        <footer>
            <div data-ng-include="'plantillas/footer.html'"> </div>
        </footer>
    </body>
</html>
```

APP/APP.JS

```
1 (function() {
2     'use strict';
3     angular.module('aplicacionLoca', ['ngRoute']);
4 })();
```

```
(function() {
    'use strict';
    angular.module('aplicacionLoca', ['ngRoute']);
})();
```

APP/ROUTER.JS

```
(function() {
    'use strict';
    angular
        .module('aplicacionLoca')
        .config(['$routeProvider', funcionLoca]);

    function funcionLoca($routeProvider){
        $routeProvider.
            when('/animales/monos', {
                templateUrl: 'plantillas/monos.html'
            }).
            when('/animales/gatos', {
                templateUrl: 'plantillas/gatos.html'
            }).
            when('/animales/perros', {
                templateUrl: 'plantillas/perros.html'
            }).
            otherwise({
                redirecTo: '/',
                templateUrl: 'plantillas/404.html'
            });
    }
})();
```


```
(function() {
    'use strict';
    angular
        .module('aplicacionLoca')
        .config(['$routeProvider', funcionLoca]);

    function funcionLoca($routeProvider){
        $routeProvider.
            when('/animales/monos', {
                templateUrl: 'plantillas/monos.html'
            }).
            when('/animales/gatos', {
                templateUrl: 'plantillas/gatos.html'
            }).
            when('/animales/perros', {
                templateUrl: 'plantillas/perros.html'
            }).
            otherwise({
                redirecTo: '/',
                templateUrl: 'plantillas/404.html'
            });
    }
})();
```


APP/MONOS.JS

```
<img src="images/mono-loco.jpg">
```


```
<img src="images/mono-loco.jpg">
```

APP/GATOS.JS

```
1 <img src="images/gato-loco.jpg">
```

<img src="images/gato-loco.jpg">

APP/PERROS.JS

```
1 <img src="images/perros-locos.jpg">
```

<img src="images/perros-locos.jpg">

APP/404.HTML

```
1 <h1>LA PAGINA NO EXISTE</h1>
```

<h1>LA PÁGINA NO EXISTE</h1>

En este ejemplo, destacar que en los diferentes enlaces del menú, por decir uno, el primero: "**#/animales/monos**", la almohadilla (**#**) tiene una funcionalidad y esta es que, si la ponemos en la dirección URL significa que vamos a cargar todo dentro del mismo documento.
Si nos fijamos bien, al hacer clic en una de las opciones del menú, cargamos únicamente los documentos o archivos HTML parciales, y los llamo parciales porque sólo contienen un trozo o script de código HTML, no un documento al completo. Al pinchar sobre un enlace no cargamos toda la página, sino únicamente uno de esos documentos parciales.

Angular y Bootstrap

AngularJS es un framework de JavaScript enfocado a la lógica de programación y carece de componentes gráficos pero una de sus ventajas es la capacidad de integrarse con otros complementos o frameworks.
AngularJS puede usarse solo o en conjunto con cualquier framework que existe actualmente en el ámbito de las nuevas tecnologías. Tradicionalmente AngularJS se utiliza con Bootstrap para general la capa gráfica.
En el ejemplo que presento a continuación se puede ver como integro los dos frameworks, AngularJS y Bootstrap.

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
<head>
<title>Angulas JS: AngularJS con Bootstrap</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<!-- BOOTSTRAP CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css">
<!-- JQUERY JS -->
<script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
<!-- BOOTSTRAP JS -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<script src="https://code.angularjs.org/1.4.8/angular-route.min.js"></script>
</head>
<body data-ng-controller="controladorLoco as vm">
    <!-- NavBar -->
    <div ng-include src=" 'includes/bootstrap-y-angular/menu.html' "> </div>
    <!-- /NavBar -->

    <!-- Jumbotron -->
    <div class="jumbotron">
        <div class="container">
            <h1>Bienvenido a la aplicación Loca</h1>
            <input type="text" class="form-control" data-ng-model="vm.extra">
            <a class="btn btn-warning btn-primary" data-ng-click="vm.add()">Listado de Animales locos</a>
        </div>
    </div>
    <!-- /Jumbotron -->

    <!-- Container -->
    <div class="container" >
        <ul class="list-group" data-ng-show="vm.visibles" >
            <li class="list-group-item" data-ng-repeat="item in vm.lista">{{ item }}</li>
        </ul>
    </div>
    <!-- /Container -->
    <script type="text/javascript">
        (function() {
            'use strict';
            angular
                .module('aplicacionLoca', [])
                .controller('controladorLoco', controladorLoco);

            function controladorLoco(){
                var vm = this;

                vm.lista = ['Monos', 'Gatos', 'Perros'];
                vm.visibles = true;

                vm.add = function () {

                    vm.lista.push( vm.extra );

                }
            }
        })();
    </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html data-ng-app="aplicacionLoca">
<head>
<title>Angulas JS: AngularJS con Bootstrap</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.mi
n.css">
<script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.
js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min
.js"></script>
<script src="https://code.angularjs.org/1.4.8/angular-
route.min.js"></script>
</head>
<body data-ng-controller="controladorLoco as vm">
    <!-- NavBar -->
    <div ng-include src=" 'includes/bootstrap-y-angular/menu.html' ">
</div>
    <!-- Jumbotron -->
    <div class="jumbotron">
        <div class="container">
            <h1>Bienvenido a la aplicación Loca</h1>
            <input type="text" class="form-control" data-ng-
model="vm.extra">
            <a class="btn btn-warning btn-primary" data-ng-
click="vm.add()">Listado de Animales locos</a>
        </div>
    </div>
    <!-- Container -->
    <div class="container" >
        <ul class="list-group" data-ng-show="vm.visibles" >
            <li class="list-group-item" data-ng-repeat="item in
vm.lista">{{ item }}</li>
        </ul>
    </div>
    <script type="text/javascript">
        (function() {
            'use strict';
            angular
                .module('aplicacionLoca', [])
                .controller('controladorLoco', controladorLoco);

            function controladorLoco(){
                var vm = this;

                vm.lista = ['Monos', 'Gatos', 'Perros'];
                vm.visibles = true;

                vm.add = function () {
                     vm.lista.push( vm.extra );
                 }
            }
        })();
    </script>
</body>
</html>
```

Autor: David Cueli

Web: dcueli.com

Noviembre 2015