


Desarrollo declarativo




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

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

Directivas	atributos ng- específicos de angularJS que se pueden asignar a cualquier etiqueta HTML.
Expresiones	lenguaje de plantillas mediante {{}}

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



Directivas

- ngApp
- ngController
- ngInclude
- ngModel

- ngRepeat
- ngInclude

- ngBind – ngBindHtml
- ngCloak
- ngIf
- ngShow-ngHide

Directivas de eventos

- ngClick
- ngDbclick
- ngBlur
- ngFocus
- ngSubmit
- ngMouseEnter
- ngMouseLeave

- ngStyle - ngClass
- ngScr - ngHref

Directivas y formularios

Creación de Directivas propias

Filtros

Escalares

Arrays

Escalares (filtros de formato)

- lowercase / uppercase
- number
- date
- currency

Arrays

- filter
- limitTo
- orderBy

Creación de Filtros propios

Directivas básicas



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



Directivas y compilación



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

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

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

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

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



Más Directivas básicas



- **ng-model** relaciona elementos del DOM con modelos de datos, informando al compilador HTML de AngularJS de que se está declarando una variable del modelo.
Sirve de base al **doble binding**
 - ✓ Se utiliza en los controles de formulario, como INPUT, SELECT, TEXTAREA o controles personalizados.
 - ✓ crear una propiedad dentro del *scope* y la enlaza (*binding*) con el correspondiente control de formulario de la vista

- **ng-click** permite definir el manejador del evento clic de un determinado elemento del DOM



Doble binding



- expresiones
- directivas ng-model

```
<form name="formulario">
  <label for="fNombre" id="lblNombre">Indica tu nombre</label>
  <input type="text" id="fNombre" name="fNombre"
    ng-model="nombre">
</form>
<p>Hola {{nombre}}. </p>
```

binding

información desde
el *scope* a la vista

two-way binding

información desde
la vista al *scope*.



Expresiones



- En AngularJS las expresiones se evalúan respecto al **scope** en lugar de respecto al objeto global *window*
- Lógica limitada: no se pueden incluir en ellas condicionales (excepto el operador ternario), bucles o excepciones
- Se les puede dar formato mediante filtros

Referencias a modelos {{Dato}}

Operaciones
aritméticas básicas {{Dato + 4}}

Empleo de los métodos de
los objetos envolventes de
JS (e.g. String) {{ "Beginning AngularJS".toUpperCase() }}
{{ "ABCDEFGH".indexOf('D') }}

Uso del operador ternario {{Dato == 1 ? "Red" : "Blue"}}



Ejemplos de expresiones

Ejemplos de Expresiones

Operaciones aritméticas básicas

$6 - 4 = 10$

El resultado se obtiene con la expresión: `{{6 - 4}}`

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

BEGINNING ANGULARJS

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

3

Resultado de la expresión: `{{"ABCDEF6".indexOf('D')}}`

Uso del operador ternario

Red

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



Alejandro L. Cerezo - Madrid 2015

Iteraciones



ngRepeat

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

`ng-init ="aElementos = [...]"` Existe un *array* en el modelo

```
<tag_html ng-repeat="elemento in aElementos | filtros | ordenación">  
  ... {{elemento}} ...  
</tag_html>
```

Obsérvese su similitud con el bucle **for ... in** de JS

Más adelante veremos las opciones de filtrado y ordenación



Scope de las iteraciones



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

- en él se pueden definir propiedades específicas de este ámbito, que se inicializan mediante la directiva ngInit
- además existen una serie de propiedades ya predefinidas que toman valor dinámicamente conforme se realizan las sucesivas iteraciones

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

Las otras variables de este tipo son

\$first
\$middle
\$last
\$even
\$odd



Scope de las iteraciones



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

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

track by \$index →

permite que ngRepeat incluya elementos repetidos con el mismo contenido



Pensamientos: ejemplo de iteraciones

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

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

Pensamientos

Indica tu nombre

Comparte una idea

Hola Pepe

Pensamientos que has tenido

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



Alejandro L. Cerezo - Madrid 2015

Includes



ngInclude

incluye en el documento el contenido de otro archivo

Directivas

- como atributo (A)
- como etiqueta (E)

```
<div ng-include="include-me.html"></div>  
<ng-include src="include-me.html"></ng-include>
```



ATENCIÓN: el parámetro es un string: lleva sus propias " simples, además de las "" dobles propias de cualquier atributo HTML

Index.html

```
<ng-include src="cabecera.partial.html"
```

```
<ng-include src="pie.partial.html"
```



HTML Parciales: estructura

Creamos un index.html que incluye una cabecera y un pie en html parciales, que incluyen expresiones con variables Angular

cabecera.partial.html

Desarrollo Web con AngularJS

Ejemplo de html parciales incluidos en index.html

pie.partial.html

Alejandro Cerezo - Madrid 2017



Más directivas



ngBind equivale a las expresiones `{{}}` pero incluyendo la funcionalidad de ngCloak

`` ↔ `{{"2+2"}}`

ngCloak "camuflaje": evita que se vean momentáneamente las directivas antes de que angularJS las compile

`<p ng-cloak>{{ 2 + 2 }}</p>`

ngIf genera elementos del DOM de forma condicional

`<div ng-if="isCondicion"></div>`

ngShow / ngHide muestran / ocultan un elemento según se les asigne un valor true o false



Directivas y binding



ngBind

equivale a las expresiones `{{}}` pero incluyendo la funcionalidad de ngCloak

`` ↔ `{{"2+2"}}`

ngBindHtml

evalúa la expresión e inserta el HTML resultante de forma segura

``



Evaluación y seguridad



Al evaluar una expresión que incluye HTML (e.g. **ngBindHtml**) Angular fuerza a que se haga de forma segura. Esto es especialmente importante si se trata de valores introducidos por el usuario

El resultado de evaluar la expresión debe ser "saneado" utilizando el servicio **\$sanitize**.

Alternativamente, si los valores son seguros, puede evitarse el proceso anterior empleando el método **trustAsHtml()** del servicio **\$sce**

```
$sce.trustAsHtml(<expresión_HTML>);
```

Veremos es uso de este servicio en el ejemplo de presentar datos de diversos libros mediante el servicio **\$http**



Directivas y eventos



ngClick
ngDbclick
ngBlur
ngFocus
ngSubmit

Todos los eventos de JS tienen definida su correspondiente directiva

A nivel de la vista, puede definirse cual será la función manejadora del evento

ngMouseEnter
ngMouseLeave

funciones siempre incluidas en el *controller*, para que tengan acceso al *scope*



Directivas y referencias



ng-src

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

```

```

En lugar de

```

```

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

ng-href

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



Mostrario: mostrar y ocultar (2)

La directiva **ngShow / ngHide** aparece ahora asociándola a los eventos **ngMouseEnter** y **ngMouseLeave** en una imagen

Mostrario

Ratones inalámbricos disponibles en varios colores

Para verlos, coloca el cursor sobre la imagen



Rojo

Verde

Azul



Filtros de formato



Permiten dar formato a la información que se muestra en la vista

definen el uso de mayúsculas y minúsculas | lowercase / uppercase

definen el aspecto de los valores monetarios | currency / currency : '€'

definen el número de decimales de un dato | number / number : <n>

definen en diversos detalles el formato de una fecha; permiten utilizar una serie de formatos ya predefinidos | date:



Formatos de fechas



yyyy	Four-digit representation of year (for example, AD 1 => 0001, AD 2010 => 2010)
yy	Two-digit representation of year, padded (00-99) (for example, AD 2001 => 01, AD 2010 => 10)
y	One-digit representation of year (for example, AD 1 => 1, AD 199 => 199)
MMMM	Month in year (January-December)
MMM	Month in year (Jan-Dec)
MM	Month in year, padded (01-12)
M	Month in year (1-12)
dd	Day in month, padded (01-31)
d	Day in month (1-31)
EEEE	Day in week (Sunday-Saturday)
EEE	Day in week (Sun-Sat)

HH	Hour in day, padded (00-23)
H	Hour in day (0-23)
hh	Hour in AM/PM, padded (01-12)
h	Hour in AM/PM, (1-12)
mm	Minute in hour, padded (00-59)
m	Minute in hour (0-59)
ss	Second in minute, padded (00-59)
s	Second in minute (0-59)
.sss o ,sss	Millisecond in second, padded (000-999)
a	AM/PM marker
Z	Four-digit (+sign) representation of the time zone offset (-1200 – +1200)
ww	ISO 8601 week of year (00-53)
w	ISO 8601 week of year (0-53)



Formatos de fechas



Atajos: formatos predefinidos

Parameter	Description	Example
medium	equivalent to 'MMM d, y h:mm:ss a' for en_US locale	Sep 3, 2010 12:05:08 PM
short	equivalent to 'M/d/yy h:mm a' for en_US locale	9/3/10 12:05PM
fullDate	equivalent to 'EEEE, MMMM d, y' for en_US locale	Friday, September 3, 2010
longDate	equivalent to 'MMMM d, y' for en_US locale	September 3, 2010
mediumDate	equivalent to 'MMM d, y' for en_US locale	Sep 3, 2010
shortDate	equivalent to 'M/d/yy' for en_US locale	9/3/10
mediumTime	equivalent to 'h:mm:ss a' for en_US locale	12:05:08 PM
shortTime	equivalent to 'h:mm a' for en_US locale	12:05 PM



Tenemos que referenciar el script correspondiente

```
<script src="../../node_modules/angular-118n/angular-locale_es-es.js"></script>
```

Filtros (1)



Se indican a continuación de una expresión, después de |
Permiten dar formato o filtrar los resultados

filter Permiten definir una búsqueda dentro de los elementos de la colección que estamos iterando. No afectan a la colección en sí, sino a su presentación en la vista.

Se indica con el término **filter**: seguido por la expresión (la cadena o variable donde está la cadena) que nos sirve para filtrar.

```
<li ng-repeat="pensamiento in aPensamientos | filter: 'yo' ">
```

Solo mostraría los elementos que incluyeran 'yo'



Filtros (2)



limitTo: Permite limitar el número de iteraciones de un *array* que realiza ngRepeat

```
<li ng-repeat="pensamiento in aPensamientos | limitTo: 5">
```

Sólo mostraría los 5 primeros elementos

orderBy va seguido de la propiedad del modelo que se utiliza para ordenar y opcionalmente del carácter descendente indicado mediante un valor booleano true

carácter descendente indicado mediante un valor booleano true



Lista de autores: filtros, orden

El modelo incorpora una colección de objetos con la estructura

- nombre
- apellido
- fecha nacimiento
- país
- géneros: []

Permitimos en la salida ordenar los datos y filtrarlos según varios criterios

Lista de Autores

Elije el orden que deseas

Elije el genero literario que deseas

Campos a mostrar

País ☒

Fecha de nacimiento ☐

- Georges Simenon - France - Genero: policiaco.
- Isaac Asimov - Russia - Genero: policiaco, ciencia ficción.
- Agatha Christie - UK - Genero: policiaco.
- Stephen King - USA - Genero: policiaco, ciencia ficción, fantástico.
- Philip K. Dick - USA - Genero: ciencia ficción.
- Terry Pratchett - UK - Genero: fantástico.

Alejandro L. Cerezo - Madrid 2015



```
this.aAutores = [  
  { nombre: 'Georges',  
    apellido: 'Simenon',  
    fechaNacim: new Date(1903,0,1),  
    pais: 'France',  
    generos: ['policiaco']},  
  { nombre: 'Isaac',  
    apellido: 'Asimov',  
    fechaNacim: new Date(1919,0,1),  
    pais: 'Russia',  
    generos: ['policiaco', 'ciencia ficción']},  
  { nombre: 'Agatha',  
    apellido: 'Christie',  
    fechaNacim: new Date(1890,0,1),  
    pais: 'UK',  
    generos: ['policiaco']},  
  { nombre: 'Stephen',  
    apellido: 'King',  
    fechaNacim: new Date(1947,0,1),  
    pais: 'USA',  
    generos: ['policiaco', 'ciencia ficción', 'fantástico']},  
  { nombre: 'Philip K.',  
    apellido: 'Dick',  
    fechaNacim: new Date(1928,0,1),
```

```
    pais: 'USA',
    generos: ['ciencia ficción']],
{
  nombre: 'Terry',
  apellido: 'Pratchett',
  fechaNacim: new Date(1948, 0, 1),
  pais: 'UK',
  generos: ['fantástico']}
];
}
```


Directivas y CSS (1)



ngStyle

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

```
<p ng-style ="estilos">
```

En el *controller*:

```
$scope.estilos = {"background-color" : "green",  
"color": "silver"}
```

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



Directivas y CSS (2)



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

La propiedad del modelo puede tener varios formatos:

- una cadena de caracteres con uno o más nombres de clases
- un array con los diferentes nombres de clases
- un objeto, en el que
 - las claves permiten especificar nombres de clases y
 - sus valores corresponden a expresiones que deben cumplirse para que éstas se apliquen.

```
<p ng-class="{positivo: total>=0, negativo: total<0}">
```



Acumulador con clases

Ejemplos de clases
que se aplican
dinámicamente

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

Acumulador con clases

Acumulador

Elige en tamaño de letra del titular: Grande ▾

Control de operación:

Incremento:

Totales:

En el acumulador llevamos [10](#)

Alejandro L. Cerezo
CLE Formación
Madrid - 2015



Directivas y formularios



input
textarea
select

ngModel: indica la propiedad del modelo/*scope*. a la que se asocia el elemento

→ doble binding

```
<label for="nif">NIF:</label>
<input id="nif" type="text" id = "nif" name="nif"
ng-model="seguro.nif" />

<label for="casado">Casado:</label>
<input id="casado" type="checkbox" id="casado" name="casado"
ng-model="seguro.casado" >
```

En el caso de los **radio-buttons**,
cada conjunto de ellos comparte el mismo valor de ngModel

Se suele decir:

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



Inputs:

- text|password|email|number|submit|date|datetime|datetime-local|month|color|range|search|tel|time|url|week
- checkbox y radiobuttons

Formularios: Checkboxes



Checkboxes

ngModel: como en cualquier control de formulario, indica la propiedad del modelo/*scope*. a la que se asocia el elemento

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

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

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



campo, marcado a no marcado y viceversa. El valor puede ser una expresión o una llamada a una función del *scope*.

Formularios: *select* / *options*



En HTML, cada "*option*" puede tener 2 componentes

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

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

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

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



Select / options desde el modelo



ngOptions

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

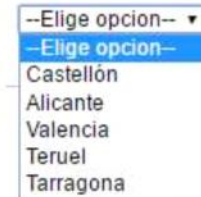
	valor as etiqueta
ngOptions	for item in altems
	track by claveunica

- ítem: Es la declaración de una variable que contendrá cada uno de los elementos de altems. (puede ser cualquier nombre que decidamos).
- valor: Una expresión que use la variable item que será el **valor** que se asignará al modelo del <select>. Este valor es opcional.
- etiqueta: Una expresión que use la variable item que será la **etiqueta** que se mostrará en cada uno de los <option>.
- claveunica: La propiedad que es clave única.



Ejemplo de select/options

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



--Elige opcion-- ▼
--Elige opcion--
Castellón
Alicante
Valencia
Teruel
Tarragona

```
<select ng-model = "miProvinciaSeleccionada"  
  ng-options = "provincia as provincia.nombre for provincia  
  in provincias track by provincia.idProvincia">  
  <option value="">--Elige opcion--</option>  
</select>
```



Formulario: Selección de opciones

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

Formulario

Selección de opciones

- ☐ Imprimir resultado
- ☐ Tono claro

Provincia

Resultado

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

Alejandro L. Cerezo - Madrid 2015



Formularios: validación



Directivas

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

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



Validación: propiedades



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

\$pristine : true si el usuario aún no ha interactuado con el formulario o control

\$dirty : es el opuesto al anterior

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

\$valid : true si se cumplen las condiciones impuestas por la directivas

\$invalid : es el opuesto al anterior

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



Validación: momento



Inicialmente ng-model vincula el proceso de validación con el evento keydown

Es posible modificar estas circunstancias mediante la directiva ng-model-options

objeto entre cuyas propiedades está

updateOn

permite indicar que evento / eventos quedan vinculados con el proceso de validación

```
<input type="text" ng-model="firstname"
ng-model-options=" {updateOn: 'blur'}" />
```

La propiedad debounce permite definir un intervalo en milisegundos antes de la respuesta al evento



Validación: ejemplo



```
<form name="myform" novalidate>
```

← Anulamos la validación estándar HTML5

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

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

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



Práctica: validación de formularios

Realizamos un formulario con:

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

Formulario

Datos personales

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

Resultado

- Nombre:
- Apellido:
- Teléfono:

Alejandro L. Cerezo - Madrid 2015

