Directives

Introducció a directives

Fins ara hem utilitzat les directives d'AngularJS , en aquesta unitat anem a veure com crear les nostres pròpies directives.

Quina és la utilitat de crear les nostres pròpies directives? Doncs és simplement és poder crear les pròpies etiquetes HTML i poder fer coses com:

- Directiva per indicar que un input és per dates i que salta el desplegable: `ca-fecha`

```
<input ng-model="fechaNacimiento" ca-fecha>
```

- Directiva para fer pestanyes: `ca-tabs` y `ca-tab`

```
<ca-tabs>
  <ca-tab nombre="Compras">
    Contenido de la segunda pesaña
  </ca-tab>
  <ca-tab nombre="Ventas">
    Contenido de la segunda pesaña
  </ca-tab>
</ca-tabs>
```

- Directiva per la paginació: `ca-paginacion`

```
<ca-paginacion numero-paginas="10"></ca-paginacion>
```

Directives amb dades

Passem ara a veure alguna característica de les directives

Formes

El primer que fem és que hem utilitzant una directiva com un atribut, com en `ca-fecha` però també es pot fer com si fos directament un nou tag HTML , com en `ca-paginacion`.
AngularJS permet ambdues formes e inclús com classes CSS i com comentaris [1)], però és tant estrany que millor no parlem d'elles. Així que a efectes pràctics una directiva es pot utilitzar com:

- Un atribut: Es posa com un atribut d'un element HTML

```
1    <input ng-model="fechaNacimiento" ca-fecha>
```

- Un Element o tag: És directament un nou element o tag però té la desavantatge que en versions antigues de Internet Explorer pot donar problemes

```
1    <ca-paginacion numero-paginas="10"></ca-paginacion>
```

Prefixes en Directives

El equip d' Angular recomana que les directives tinguin sempre un prefix de forma que no xoquin amb altres directives creades per altres desenvolupadors o amb actuals o futures etiquetes de HTML. Per això las directives de AngularJS comencen sempre por "ng". En les directives que creem en aquest curs sempre començaran per "ca" per "Curso de Angular".

Exemples:

- `ng-view` : Prefixa `ng` per a indicar que és una directiva estàndard de AngularJS
- `ng-repeat` : Prefixa `ng` per a indicar que és una directiva estàndard de AngularJS
- `ca-paginacion` : Prefixa `ca` per a indicar que és una directiva creada en el Curs d'AngularJS
- Etc.

Nom

Fins ara hem vist els noms de les directives com guions per a separar paraules.

Ex:

- `ng-repeat`
- `ng-if`
- `ca-paginacion`
- `ca-tabs-vertical`
- Etc.

El nom *verdader* de les directives quant es defineixen realment és el següent

Ex:

- `ng-repeat` → **`ngRepeat`**
- `ng-if` → **`ngIf`**
- `ca-paginacion` → **`caPaginacion`**
- `ca-tabs-vertical` → **`caTabsVertical`**
- Etc.

Es a dir el nom de les directives segueix el format dels identificadors en JavaScript. Aquest és el nom que s'utilitza en la documentació de AngularJS quan es refereixen a una directiva i el que utilitzem quan creiem una nova directiva.

Variacions del nom en HTML

Donat el nom de la directiva tal i com es veu en JavaScript es pot utilitzar en la pàgina HTML seguint les següents regles:

- Afegir el prefix `data-` o `x-` per a que els validadors HTML sàpiguen que és una directiva pròpia nostra.
- Separar cada paraula amb alguns dels següents caràcters "-" , ":" o "_".

Per lo tant la directiva `ngView` pot utilitzar-se en una pàgina HTML amb les següents formes:

- `ng-view`
- `ng:view`
- `ng_view`
- `data-ng-view`
- `data-ng:view`
- `data-ng_view`
- `x-ng-view`
- `x-ng:view`
- `x-ng_view`

Encara que lo normal és restringir-se només a les següents formes.

- `ng-view`
- `data-ng-view`

Directives ng

Sobre les directives NG, hi ha poc més a dir i podem resumir-les en les natives del propi AngularJS.

Definició de directives

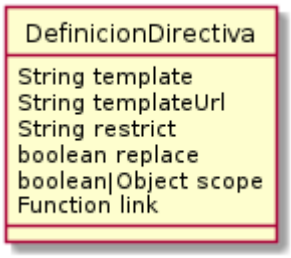
Creant una directiva

Per crear una directiva hem de definir-la dins d'un mòdul al igual que fèiem per els controladors, filtres, etc.

El mètode a utilitzar del modulo és directive(nombre,funcionFactory) sent:

- nombre: El nom de la directiva seguiment la forma dels identificadors de JavaScript.
- funcionFactory: És una funció que ens retorna un objecte amb la definició de la directiva. Com totes les funcions factory pot ser un array per poder injectar-li dependències.

El objecte amb la definició de la directiva és el verdader objecte que conté tota la informació de la nostra directiva i comprendre totes les seves propietats implicaria saber tot sobre les directives així que en aquesta unitat només anem a veure unes quantes propietats del objecte amb la definició de la directiva.
En aquesta unitat només veurem les següents propietats:



En aquest tema només veurem les següents propietats:

- `template` : És un String com el HTML per el que es substituirà la directiva. Si està aquesta propietat no pot estar `templateUrl` ja que son excloents.
- `templateUrl` : És un String com una URL d'un fitxer que conté el HTML per el que es substituirà la directiva. Si està aquesta propietat no pot estar `template` ja que són excloents.
- `restrict`: Un String que indica com pot utilitzar-se la directiva si com un atribut o com un element
  - Si val "E" només es podrà utilitzar com element
  - Si val "A" només podrà utilitzar-se com atribut
  - Si val "EA" o "AE" es podrà utilitzar com element i com atribut. Aquest és el funcionament per defecte si no es posa res en la propietat `restrict`.
- `replace` : Si val `false` el contingut del *template* s'afegirà dins del tag de la pròpia directiva. Però si val true es traurà el tag de la directiva i només estarà el contingut del *template*.

En els 2 temes següents veurem les propietats link i scope.

Exemple

Anem a fer la nostre primera directiva anomenada "acTitulo" que simplement mostra el text "Hola Mundo" com a títol `<h1>`.

```
1   app.directive("acTitulo",[function() {
2
3     var directiveDefinitionObject ={
4       restrict:"E",
5       replace : true,
6       template:"<h1>Hola Mundo</h1>"
7     }
8
9     return directiveDefinitionObject;
10  }]);
```

- Línia 1: Crearem la directiva de nom “acTitulo” i com segon paràmetre està la funció factory.
- Línia 3: Crearem el objecte `directiveDefinitionObject` que contindrà la definició de la directiva
- Línia 4: Al donar valor a la propietat `replace` el valor de `true` s’eliminarà de la pàgina el tag de la pròpia directiva i només estarà el contingut del *template*.
- Línia 5: Indicarem mitjançant la propietat `restrict` que l’element només pot ser un element o tag i no pot estar com atribut d’un altre element o tag.
- Línia 8: Retornem el objecte amb la definició de la directiva.

Utilitzarem la directiva de la següent forma:

```
1   <ac-titulo></ac-titulo>
```

I la directiva generarà el següent HTML

```
1   <h1>Hola Mundo</h1>
```

Si el valor de `replace` hagués sigut `false` s’hagués generat el següent HTML:

```
1   <ac-titulo>
2       <h1>Hola Mundo</h1>
3   </ac-titulo>
```

Directives i scopes

9.2 scope

En el exemple de directiva, `acTitulo`, que acabem de veure en el tema anterior el text del títol sempre era “Hola Mundo”, Anem a modificar la directiva per a que el text pugui ser definit per el usuari de la directiva.
La forma de modificar la directiva es simplement la següent:

```
1   app.directive("acTitulo",[function() {
2
3     var directiveDefinitionObject ={
4       restrict:"E",
5       replace : true,
6       template:"<h1>{{texto}}</h1>"
7     }
8
```

```
9       return directiveDefinitionObject;
10    }]);
```

- Línia 6: El text a mostrar ara s'obté de la propietat `texto` que hi ha en el `scope` de la directiva.

Ara ja podem tenir diferents text i per a això només s'ha de canviar la propietat `texto` del `scope` de la directiva. Però ara apareix el problema ja que AngularJS ens permet moltes formes de crear aquest nou `scope`, les diferents formes són:

- Tenir exactament el mateix `scope` que hi ha en el controlador
- Tenir un nou `scope` però que hereti del `scope` del controlador, tal i com passava amb la directiva 4.6 ng-if
- Tenir un nou `scope` que NO hedera del controlador.

Para establir quin de les 3 utilitzarà la nostra directiva , s'utilitzarà la propietat `scope` del objecte amb la definició de la directiva.

Mateix scope del controlador

Si la propietat `scope` del objecte amb la definició de la directiva val el valor de `false` , la directiva utilitzarà com `scope` exactament el mateix `scope` que té el controlador. Aquest és el funcionament per defecte en cas de que no posem res en la propietat `scope` del objecte amb la definició de la directiva.
Ara la directiva quedaria així:

```
1    app.directive("acTitulo",[function() {
2
3      var directiveDefinitionObject ={
4        restrict:"E",
5        replace : true,
6        template:"<h1>{{texto}}</h1>",
7        scope:false
8      }
9
10     return directiveDefinitionObject;
11   }]);
```

- Línia 7: Indicarem que el `scope` de la directiva és el mateix objecte que el `scope` del controlador

Ara ja podrem modificar el text de la directiva únicament posant :

```
1   app.controller("PruebaController", function($scope) {
2       $scope.texto="Adios mundo";
3   });
```

No es recomanable utilitzar aquesta opció ja que la directiva serà molt dependent de lo que hi ha en el `scope` del controlador i el que s'utilitzi la directiva podria no ser ni conscient d'aquesta relació.
Fer això seria algo així a utilitzar variables globals dins d'una funció. Sent la variable global el `scope` del controlador.

Nou scope però que hereta del scope del controlador

Si la propietat `scope` del objecte amb la definició de la directiva val el valor de `true`, la directiva utilitzarà como `scope` un nou `scope` però que ha heretat del `scope` del controlador. Aquest és el cas com el que explicarem en Herència de $scope.

Ara la directiva quedaria així:

```
1    app.directive("acTitulo",[function() {
2
3      var directiveDefinitionObject ={
4        restrict:"E",
5        replace : true,
6        template:"<h1>{{texto}}</h1>",
7        scope:true
8      }
9
10     return directiveDefinitionObject;
11   }]);
```

- Línia 7: Indicarem que el `scope` de la directiva és un nou `scope` però que hereta del `scope` del controlador

Com en el cas anterior ja podem modificar el text de la directiva únicament posant :

```
1   app.controller("PruebaController", function($scope) {
2       $scope.texto="Adios mundo";
3   });
```

Tampoc recomanem utilitzar aquesta opció ja que la directiva serà molt dependent del que hi ha en el `scope` del controlador i el que s'utilitzi la directiva tampoc podria no ser ni conscient d'aquesta relació.
Segueix sent com utilitzar variables globals dins d'una funció. Sent la variable global el `scope` del controlador.

Nou scope

Com ja hem dit, les dos formes anteriors no les recomanem ja que utilitzarem la directiva `acTitulo` de la següent forma:

```
1    <ac-titulo></ac-titulo>
```

I llavors en la pàgina HTML apareix el text “Adios Mundo”, De ón ha aparegut el text aquest? Sembla una mica estrany. La forma correcta de utilitzar la directiva hauria de haver sigut la següent:

```
1    <ac-titulo texto="Adios Mundo"></ac-titulo>
```

És dir , les directives no tenen que utilitzar variables globals com pot ser el `scope` del controlador sinó que tot el que necessitem es deu passar com *arguments* en forma d’atributs en la pròpia directiva. Per això tenim que crear un nou objecte JavaScript con `{ }` en la propietat `scope` del objecte amb la definició de la directiva.
Ara la directiva quedaria així.

```
1    app.directive("acTitulo",[function() {
2
3      var directiveDefinitionObject ={
4        restrict:"E",
5        replace : true,
6        template:"<h1>{{texto}}</h1>",
7        scope:{
8
9        }
10     }
11
12     return directiveDefinitionObject;
13   }]);
```

- Línia 7: Indicarem que el `scope` de la directiva és un nou `scope` que no té res a veure amb el `scope` del controlador.

Però ara ens falta dir-li al controlador que volem que el contingut del atribut `texto` de la directiva sigui una propietat del `scope` de la directiva. Per fer-ho hi ha 2 formes diferents que veurem a continuació.

Enllaç unidireccional

Simplement tenim que afegir un atribut al `scope` on el valor sigui una arrova ”@”. D’aquesta forma decidim que volem copiar el valor del atribut en el `scope`.
Ara la directiva quedaria així:

```
1   app.directive("acTitulo",[function() {
2
3     var directiveDefinitionObject ={
4       restrict:"E",
5       replace : true,
6       template:"<h1>{{texto}}</h1>",
7       scope:{
8         texto:"@"
9       }
10    }
11
12    return directiveDefinitionObject;
13  }]);
```

- Línia 8 : Al posar texto:”@” el que li hem dit és que el **valor** del atribut texto es copiï en la propietat texto del scope de la directiva.

Ara ja podrem utilitzar la directiva així:

```
1   <ac-titulo texto="Texto definido en el atributo"></ac-titulo>
```

I es generarà el següent HTML:

```
1   <h1>Texto definido en el atributo</h1>
```

Però, que passa si el text a mostrar està en una propietat del `$scope` del controlador? Per exemple en la propietat `mensaje` del `$scope` del controlador.

```
1   app.controller("PruebaController", function($scope) {
2       $scope.mensaje="Texto definido en el $scope del controlador";
3   });
```

Doncs no hi ha més que utilitzar la directiva utilitzant les ja conegudes claus ”{ }” de AngularJS:

```
<ac-titulo texto="{{mensaje}}"></ac-titulo>
```

I es generarà el següent HTML:

```
<h1>Texto definido en el $scope del controlador</h1>
```

Enllaç bidireccional

El enllaç unidireccional que acabem de veure al utilitzar "@" té 2 problemes:

- És unidireccional com el seu nom indica, si canviarem el valor de la propietat `texto` del `scope` de la directiva no es canvia la propietat `mensaje` del `$scope` del controlador
- Tenir que utilitzar "{ }" pot resultar una mica estrany si el que volem és sempre enllaçar a una propietat del `$scope` del controlador.

Doncs bé, AngularJS ens permet utilitzar el caràcter "=" per enllaçar directament una propietat del `scope` de la directiva com una propietat del `$scope` del controlador.
El que volem posar és el següent:

```
<ac-titulo texto="mensaje"></ac-titulo>
```

Volem que lo que posem en el atribut `texto` sigui tractat com el nom d'una propietat del `$scope` del controlador i no com el text realment. És com afegir una indirecció.
Ara la directiva quedaria així:

```
app.directive("acTitulo",[function() {

  var directiveDefinitionObject ={
    restrict:"E",
    replace : true,
    template:"<h1>{{texto}}</h1>",
    scope:{
      texto:"="
    }
  }

  return directiveDefinitionObject;
```

```
13    }]);
```

- Línia 8 : Al posar `texto:"="` el que li hem dit és que la propietat `texto` del `scope` de la directiva sigui exactament la propietat del `$scope` del controlador especificat en el atribut `texto` de la directiva. (sembla un embarbussament).

Al fer aquest canvi ja podrem utilitzar la directiva de la següent forma:

```
1    <ac-titulo texto="mensaje"></ac-titulo>
```

I si en el controlador establim el valor de la propietat `mensaje`:

```
1    app.controller("PruebaController", function($scope) {
2        $scope.mensaje="Texto definido en el $scope del controlador";
3    });
```

Es generarà el següent HTML:

```
1    <h1>Texto definido en el $scope del controlador</h1>
```

Però ara el interesant es que si canviéssim el valor de la propietat `texto` en el `scope` de la directiva també es canviaria en la propietat `mensaje` del `$scope` del controlador.
Anem ara a canviar la directiva per afegir un botó que canviï el valor del `scope` de la directiva.

```
1     app.directive("acTitulo",[function() {
2
3       var directiveDefinitionObject ={
4         restrict:"E",
5         replace : true,
6         template:"<div><h1>{{texto}}</h1><button ng-click=\"texto='Texto cambiado desde dentro
    directiva</button></div>",
7         scope:{
8           texto:"="
9         }
10      }
11
12      return directiveDefinitionObject;
13    }]);
```

- Ara la directiva inclou un botó que al clicar-lo fa el següent: `texto='Texto cambiado desde dentro de la directiva'` però com s'executa dins de la directiva , el que canvia és la propietat `texto` del `scope` de la directiva.

Si cliquem en el botó es canviaría també la propietat `mensaje` del `$scope` del controlador. És a dir que ja tenim el enllaç bidireccional:

- Si hi ha un canvi en el `$scope` del controlador es modifica el `scope` de la directiva
- Si hi ha un canvi en el `scope` de la directiva es modifica el `$scope` del controlador.

¿Que passa si utilitzant `texto:"="` no volem tenir el text en el `$scope` del controlador sinó posar-lo directament en la directiva. AngularJS té la següent sintaxis:

```
1    <ac-titulo texto="'Texto directamente en el atributo'"></ac-titulo>
```

S'ha de fixar-se en els 2 apòstrofs que hem posat abans i després del text .

Enllaç unidireccional versus bidireccional

Ara que hem vist les tipus de enllaços unidireccionals y bidireccionals anem a fer una comparativa entre ells

	Enllaç unidireccional	**Enllaç bidireccional**
Caràcter utilitzat	`@`	`=`
Expressió per enllaçar a valors literals	`valor`	`'valor'`
Expressió per enllaçar a propietats del `$scope` del controlador	`{{nombrePropiedad}}`	`nombrePropiedad`
Si es modifica la propietat del `$scope` del controlador	Si es modifica la propietat del `scope` de la directiva	Si es modifica la propietat del `scope` de la directiva
Si es modifica la propietat del `scope` de la directiva	**NO** es modifica la propietat del `$scope` del controlador	**Si** es modifica la propietat del `$scope` del controlador

Ara la pregunta que ens queda és: Quan hem d'utilitzar un o l'altre? La respuesta és senzilla, només depèn de si necessitem canviar una propietat del `$scope` del controlador desde la directiva.

- Si volem canviar una propietat del `$scope` del controlador des de la directiva haurem d'utilitzar el enllaç bidireccional
- Si **NO** volem canviar una propietat del `$scope` del controlador desde la directiva haurem d'utilitzar l'enllaç unidireccional i així evitarem la possibilitat d'un error i que poguéssim canviar el `$scope` del controlador de la directiva sense voler-ho.

Diferents noms

Una última característica que ens permet AngularJS en enllaçar amb les propietats del Scope, és fer que tinguin diferent nom.

Imaginem que hem decidit que l'atribut text de la directiva ara es digui txt.

La directiva ara s'utilitzaria de la següent manera:

```
1    <ac-titulo txt="{{mensaje}}"></ac-titulo>
```

Però hem decidit que no volem canviar la propietat del scope de la directiva que volem que segueixi sent text. Doncs només hem de per afegir a la "@" el nom de l'atribut, en cas que no sigui igual que el de la propietat `scope`.

```
1     app.directive("acTitulo",[function() {
2
3       var directiveDefinitionObject ={
4         restrict:"E",
5         replace : true,
6         template:"<h1>{{texto}}</h1>",
7         scope:{
8           texto:"@txt"
9         }
10      }
11
12      return directiveDefinitionObject;
13    }]);
```

- Línia 8: Al ser diferent el nom del atribut i el nom de la propietat del `scope` , li posem junt a la ”@” el nom del atribut de la directiva.

I ara ja podrem utilitzar la directiva com el atribut “txt” en canvi de “texto”.

Exemple

El exemple consisteix en fer 2 directives com la última que hem utilitzat , la del botó, però que una de elles tingui enllaç unidireccional i la altre enllaç bidireccional.

- acTituloUnidireccional
- acTituloBidireccional

Tindrem 2 propietats en el `$scope` del controlador per enllaçar en cada una de las directives. Las propietats són:

- `mensajeUnidireccional`
- `mensajeBidireccional`

En prémer sobre els botons "Canviar valor de `scope.texto` de la directiva" es canviaran els valors de les propietats del `scope` de les directives i es podrà comprovar si també es canvien les propietats del `$ scope` del controlador. D'aquesta manera comprovarem l'enllaç des de la directiva cap al controlador.

En prémer sobre els botons "Canviar valor del `$ scope`.mensaje del controlador" es canviaran els valors de les propietats del `$ scope` dels controladors i es podrà comprovar si també es canvien les propietats del `scope` de les directives. D'aquesta manera comprovarem l'enllaç des del controlador cap a la directiva.

API.

A la hora de declarar una directiva podem assignar les següents propietats:

- **restrict**: per delimitar l'àmbit de declaració de la directiva a un atribut, element, estil o comentari. És la tipologia de directives que vam veure anteriorment i s'especifica amb un literal 'A', 'E', M, 'C' o 'EA'.
- **template**: especifica el codi HTML que es compilarà en l'àmbit de la directiva aplicant les propietats definides també en l'àmbit de la mateixa
- **templateUrl**: la recomanació és que quan el codi de la plantilla es fa massa gran s'externalitzi en un fitxer html a part al qual podria apuntar la propietat templateUrl que és como un ngInclude,
- **scope**: totes les directives tenen associat un àmbit que per defecte és el del seu pare, el normal és que sigui un controlador. Mitjançant aquest atribut que pot adoptar els següents valors es pot ajustar:
- **false**: és el comportament per defecte, hereta l'àmbit i qualsevol modificació tant dins com fora de la directiva d'una propietat del model es veurà reflectida en els dos sentits,
- **true**:, es crea un nou àmbit fent una còpia de l'àmbit del pare de manera tal que, en ser un nou àmbit les modificacions en el model que es realitzin dins de la directiva no es veuran reflectides cap a l'exterior, però les modificacions en el model del àmbit del pare si tindran reflex en l'àmbit de la directiva

  {}: es crearà un àmbit nou independent del pare; és l'àmbit recomanat perquè realment una directiva pugui ser reutilitzada i no depengui ni hereti tot l'àmbit del seu pare. Si bé, dins de la declaració d'àmbit es poden passar valors especificant una sèrie de prefixos que permetran no només passar propietats del model de l'àmbit del pare sinó vincular-les a un o dos sentits. Aquestes propietats es declaren com atributs de l'element HTML de la directiva. així:

  @: vinculació en un sol sentit, en crear-se l'àmbit de la directiva el valor d'aquesta propietat s'assigna al nou àmbit de tal manera que les modificacions dins de la directiva no afecten a l'àmbit del pare però sí a la contra. El valor de la propietat ha de ser avaluada {{}} en la declaració de l'atribut.

  =: vinculació en tots dos sentits, espera que el valor de la propietat sigui una referència al model, no una expressió avaluable com en el cas anterior. La vinculació es fa en els dos sentits de tal manera que les modificacions tant fora com dins de la directiva del model afecten a tots dos.

  &: vinculació de mètodes que permet invocar des de la directiva a mètodes declarats en l'àmbit del pare

  Finalment, tots aquests prefixos poden anar acompanyats d'una redefinició del nom de la propietat en l'àmbit.

  Sobre l'exemple inicial farem les següents modificacions en la declaració de la directiva:

  Adaptem el codi HTML als paràmetres que ara rep la directiva i comprovem que ara sí, la directiva és totalment reutilitzable:

```
(function() {

  angular.module('tnt.ui.components')

  .directive('userInfo', [function() {
    return {
      restrict: 'E',
      scope:{
        name: '@',
        email: '@',
        rate: '=',
        click: '&'
      },
      template:'<label>Nombre: {{name}}, email: <a href="mailto:{{email}}">{{email}}</a></label><input type="text" size="2" ng-model="rate" /><button ng-click="click({name: name, rate: rate})" >Votar</button><br />'
    };
  }]);

  angular.module('tnt.ui.components').controller('DemoDirectivesCtrl', function($scope){
  $scope.users = [{
    name: 'Jose',
    email: 'jmsanchez@autentia.com',
    rate: '1'
  },
  {
    name: 'Gustavo',
    email: 'gmartin@autentia.com',
    rate: '2'
  }
  ];

  $scope.vote = function(user){
    console.log(user);
  };
});

})();
```

```
<div ng-app="tnt.ui.components" ng-controller="DemoDirectivesCtrl">
  <user-info ng-repeat="user in users" name="{{user.name}}" email="{{user.email}}" rate="user.rate" click="vote(user)"></user-info>
</div>
```

- **link**: defineix una funció com el següent contracte function link(scope, element, attrs) { … } rebent l'àmbit de la directiva, l'element de l'arbre DOM al qual està vinculat la directiva, envoltat per un objecte de la implementació de jQuery de Angular 'jqLite' i un objecte amb els parells de nom i valor dels atributs de la directiva declarats en l'element HTML. Mitjançant el paràmetre element i podent accedir tant a l'àmbit i, com a conseqüència, el valor de les propietats com als atributs que rep, estiguin o no declarats en l'àmbit es pot manipular el node HTML i tots els seus fills abans de renderitzar-se.

- **transclude**: adoptant el valor a true permet crear directives que encapsulant altres directives o emboliquen porcions de codi amb un embolcall permetent inserir el codi embolicat en la plantilla amb la directiva <ng-transclude>
- **require**: permet declarar com obligatoris paràmetres en la directiva que fan referència a altres directives.
- **controller**: permet declarar un controlador a nivell de la directiva. La diferència entre la funció de link és que el codi del controlador s'executa abans de la compilació i el del link després. No s'hauria de incloure manipulació de arbre DOM en un controlador que hauria reservar-se per la inicialització o manipulació de las propietats del àmbit de la directiva.

Cicle de vida

Exercici5.html i Ejercici5.js.