Directivas propias

Directivas propias

- Como en los controladores, las directivas se registran en los módulos
- Para registrar una directiva se utiliza la API **`module.directive`**

```
myModule.directive('greeting', function() {...});
```

Directivas propias

- Una directiva es un nuevo markup al que se le asocia un comportamiento
- Es decir que lleva asociado un template Angular

Directivas propias

```
<body ng-app="myApp">
    <div ng-controller="greetingCtrl">
        <greeting></greeting>
    </div>
</body>
```

Directivas propias

```
var myApp = angular.module('myApp', []);
myApp.controller('greetingCtrl', function ($scope){
     $scope.greeting = function(){ $scope.greeting = 'Hola!!!
Waaasssup?';}
});
```

Directivas propias

```
myApp.directive('greeting', function() {
     return {
          restrict: 'E',
          template:  '<button type="submit" class="btn btn-default"
          ng-click="greetMe()"> Greet Me </button> {{ greeting }}'
     };
});
```

Ejercicios

- Utilizando "gna-como-directiva.html" crea una app Angular que esté compuesta de una directiva <gna></gna> creada por ti, que agrupe el código html que tenías hasta ahora

Compile, controller & link

- Son 3 tipos de funciones que se pueden declarar en una directiva
- Cada una se ejecuta en un momento determinado, en un orden determinado y sirven para diferentes propósitos

Directivas propias: compile

- Es la que se ejecuta primero de las 3
- Se llama sólo una vez por cada declaración de la directiva
- En la fase de compilación Angular devuelve el template
- Utiliza **$compile** cuando quieras manipular el DOM original, antes de que Angular cree una instancia de el y del scope

Directivas propias: controller

- Donde ponemos lógica asociada a la directiva
- Los controladores pueden ser compartidos/instanciados por otras directivas
- Dicen los de Angular:
    - Buena práctica: utiliza **“controller”** para exponer una **API** para otras directivas. De lo contrario usar **“link”**.

Directivas propias: link

- Utiliza link para manipular el **DOM**
- function **`link(scope, element, attrs) { ... }`** donde:
  - **scope** es un objeto $scope Angular
  - **element** es el elemento ‘jqLite-incrustado’
  - **attrs** es un objeto hash con pares clave-valor de los nombres de los atributos y sus correspondientes valores

Directivas propias: link

```
<body ng-app="myApp">
     <greeting></greeting>
</body>
```

Directivas propias: link

```
var myApp = angular.module('myApp', []);
myApp.directive('greeting', function() {
     return {
          restrict: 'E',
          template:  '<button type="submit" class="btn btn-default"' +
          'ng-click="greetMe()"> Greet Me </button> {{ greeting }}',
          link: function (scope) {
               scope.greetMe = function(){scope.greeting = 'Hola!!!'
               + 'Waaasssup?';}
          }
}});
```

Ejercicios

- Utiliza la función link, para poder asociar la lógica de generar números aleatorios a la directiva, y sin hacer más falta el controlador

Función link: attrs

- Desde la directiva podemos acceder a los **atributos del markup** que hemos definido

```
<body ng-app="myApp">
     <greeting name="Dani"></greeting>
</body>
```

Función link: attrs

```
var myApp = angular.module('myApp', []);
myApp.directive('greeting', function() {
     return {
          restrict: 'E',
          template:  '<button type="submit" class="btn btn-default" ng-click="
          greetMe()"> Greet Me </button> {{ greeting }}',
          link: function (scope, element, attrs) {
               scope.greetMe = function(){scope.greeting = 'Hola!!!' + attrs.name +
                    'Waaasssup?';}
          }
}});
```

Ejercicios

- Pásale a tu gna a través de un atributo, el intervalo de tiempo en el que quieres que se genere un nuevo número