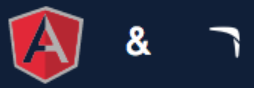
# TEMA 2

Daniel Estévez  @  & 

 & 

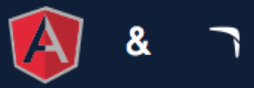
# Modulos (modules)

- Un módulo en angular es un contenedor para las diferentes partes de la aplicación: controladores, servicios, filtros, directivas, etc
- Angular no tiene método main
- En lugar demain, Angular define como deben inicializarse o arrancarse los diferentes componentes (Bootstraping)



# Modulos (modules)

- El bootstraping tiene varias ventajas:
  - Proceso declarativo más facil de entender
  - Modulos reusables
  - Independiente orden de carga
  - Unit tests solo cargan los modulos necesarios
  - E2E tests pueden usar módulos para sobrescribir configuración



# Modulos (modules)

.html

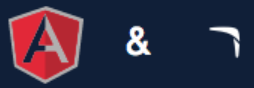
```
<body ng-app="myApp">
    <div id="prime">
        <tr ng-repeat="prime in primes |
    orderBy:$index ">
        <td>is prime? {{ prime | isPrime
        }}</td>
            </tr>
    </div>
</body>
```

.js

```
var myApp = angular.module('myApp', []);
myApp.filter('isPrime', function (){
    return isPrime();
});
```

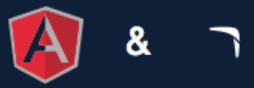
# Modulos (modules)

- ngApp en `<body ng-app="myApp">` inicia la aplicación usando tu módulo
- Y el array vacio en `var myApp = angular.module(myApp, []);` sirve para declarar las dependencias de dicho módulo



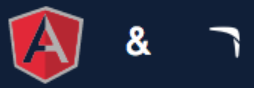
# Modulos (modules)

- Usamos módulos para implementar “Features”
- Usamos módulos para cada componente reusable (como por ejemplo directivas y filtros)
- Un módulo que depende de otros módulos por debajo de el y que contiene código de inicialización



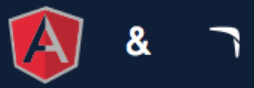
# Controladores (controllers)

- En angular, un controlador es una función constructor JavaScript que se utiliza para aumentar el scope.
- Cuando un controlador se asocia al DOM a través de la directiva ‘ngController', angular crea una instancia de un nuevo objeto controlador.
- Un subscope se crea como parámetro inyectable a la función constructor del controlador como $scope.



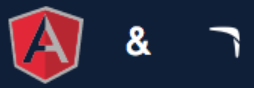
# Controladores (controllers)

- En angular, un controlador es una función constructor JavaScript que se utiliza para aumentar el scope
- Cuando un controlador se asocia al DOM a través de la directiva ‘ngController', angular crea una instancia de un nuevo objeto controlador
- Un subscope se crea como parámetro inyectable a la función constructor del controlador como $scope



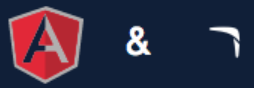
# Controladores (controllers)

- Utilizamos controladores Angular para:
  - Configurar el estado inicial del objeto $scope
  - Añadir comportamiento al objeto $scope



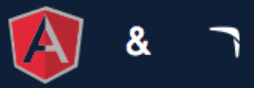
# Controladores (controllers)

- No utilizamos controladores Angular para:
  - Manipular DOM - Controladores deben contener sólo la lógica de negocio
  - Formatear inputs de formularios
  - Aplicar filtros
  - Compartir código entre controladores (Para eso servicios)



# Controladores (controllers)

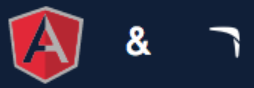
```
var myApp = angular.module('myApp', []);
myApp.controller('myAppCtrl', function ($scope){
    var greeting = 'Hola!!! Waaasssup?'
    console.log(greeting);
});
```



# Controladores (controllers)

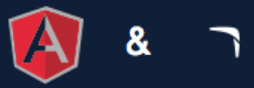
## Ejercicio

1) Utilizando "gna-uilogic-mio.html", crea una app Angular (usando module) y añádele un controlador que genere números aleatorios.



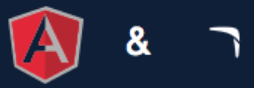
# Scope: Propiedades

- A menudo, al crear una app, necesitas inicializar el estado inicial del $scope de Angular
- $scope es simplemente un objeto
- Inicializas el estado añadiendo propiedades a este objeto



# Scope: Propiedades

- Las propiedades contienen el modelo de vista (el modelo que será presentado por la vista).
- Todas las propiedades del $scope estarán a disposición del template en el DOM, donde se ha registrado en el controlador.



# Scope: Propiedades

.html

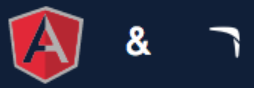
```
<body ng-app="myApp">
    <div ng-controller="'greetingCtrl">
        {{ greeting }}
    </div>
</body>
```

.js

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

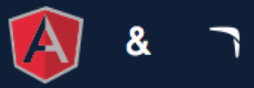
# Scope: Comportamiento

- Con el fin de reaccionar a los eventos o ejecutar cálculos durante la vista, debemos proporcionar un comportamiento al scope
- Añadimos comportamiento al alcance por métodos al objeto $ ámbito de fijación. Estos métodos son entonces disponible para ser llamado de la plantilla / vista.



# Scope: Comportamiento

- Con el fin de reaccionar a los eventos o ejecutar cálculos durante la vista, debemos proporcionar un comportamiento al scope
- Añadimos comportamiento al alcance por métodos al objeto $ ámbito de fijación. Estos métodos son entonces disponible para ser llamado de la plantilla / vista.



# Scope: Comportamiento

.html

```
<body ng-app="myApp">
    <div ng-controller="greetingCtrl">
        <button type="submit" class="btn btn-default" ng-click="greeting()">
            {{ greeting }}
        </button>
    </div>
</body>
```

# Scope: Comportamiento

.js

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

# Scope: comportamiento

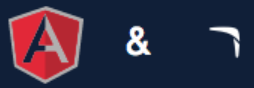
- ngClick: La directiva ngClick permite llamar una función específica del $scope cuando se hace click sobre el elemento

```
<button type="submit" class="btn btn-default"
ng-click="generateRandomNumber()">Generar Numero Aleatorio</button>
```

# Scope: Comportamiento

## Ejercicio

2) Añade a tu controlador un método que ejecute la función que calcula números aleatorios cuando este sea convocado. Crea un botón y utiliza ngClick para poder llamar a este método desde la UI.



# Servicios

- Son componentes Angular disponibles por Inyección de dependencia.
- Instanciación perezosa: Angular solo instancia un servicio cuando un componente depende de el.
- Son singletons.
- Angular ofrece muchos servicios útiles (como $http, $location etc).
- Al igual que otros identificadores básicos de Angular, los servicios integrados siempre comienzan con $ (por ejemplo, $source).

# Servicios

- Para utilizar un servicio angular, se agrega como una dependencia para el componente (controlador, servicio, filtro o directiva) que depende del servicio.
- La inyección de dependencias del angular se encarga del resto.
- Instanciación perezosa: Angular solo instancia un servicio cuando un componente depende de el.

# Servicios

.js

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

# Servicios Angular

- $interval: `$interval(fn, delay, [count], [invokeApply]);`

  Envoltorio Angular `window.setIinterval`. La función `fn` es ejecutada cada `delay` en milisegundos

```
$scope.greeting = function() {
    $interval(function() {
        $scope.greeting = 'Hola!!! Waaasssup?'
    }, 1000);
};
```

# Servicios Angular

## Ejercicio

3) Mejora tu generador de números aleatorios de tal manera que al hacer click, se genere un número aleatorio cada segundo. Puedes usar $interval inyectando el servicio en tu controlador.