

Fundamentos AngularJS

¿Qué es AngularJS?

- Es un framework estructurado para la creación de aplicaciones web dinámicas
- Permite utilizar HTML para creación de plantillas
- Permite extender la sintaxis HTML

¿Qué es AngularJS?

- Es una solución completa para el cliente:
 - Se encarga de todo el “glue code” de manipulación del DOM y AJAX
 - Aporta todo lo necesario para crear apps CRUD: Data-binding, templating, validación de formularios, routing, inyección de dependencia, etc
 - Lo necesario para testeo: Unit tests, e2e tests, mocking, etc

Fundamentos: template

- Un fichero html con markup añadido se le llama "template"

```
<body ng-app="gnaApi">
  <content></content>
  <foot></foot>
  <script src="scripts/vendor/angular.js"></script>

  <!-- API MODULE→
  <script src="scripts/gnaApi.js"></script>
  <script src="scripts/directives.js"></script>
  <script src="scripts/services.js"></script>
  <!-- -->
</body>
```

Fundamentos: vistas

- Cuando Angular carga la aplicación, parsea este nuevo markup de los templates usando el “compilador” (\$compiler)
- El cargado, transformado y renderizado DOM es a lo que llamamos vista

Fundamentos: vistas

- Cuando Angular carga la aplicación, parsea este nuevo markup de los templates usando el “compilador” (\$compiler)
- El cargado, transformado y renderizado DOM es a lo que llamamos vista

Fundamentos: directivas

- El primer tipo de markup son las directivas
- Añaden un comportamiento a elementos o atributos del HTML

Fundamentos: directivas

- El atributo ng-app esta linkado a una directiva que directamente inicializa la aplicación Angular
- La directiva ng-model guarda/actualiza el valor del campo de un input en una variable

Fundamentos: directivas

```
<body ng-app>  
  <h3> Primera App Angular: Sumador </h3>  
  Numero A.<input type="number" autocomplete="off" ng-model="numberA"/>  
  Numero B.<input type="number" autocomplete="off" ng-model="numberB"/>  
  <div id="display">  
    <div id="suma"><h3> A + B: {{numberA + numberB}} </h3></div>  
  </div>  
</body>
```

Fundamentos: directivas

- Angular ofrece un montón de directivas
- ngRepeat instancia un template por elemento en una colección

```
<ul class="nav navbar-nav nav-cover">  
  <li class='{{ section.animation }}' ng-repeat="section in sections">  
    <a href="{{section.hash}}">{{ section.text }}</a>  
  </li>  
</ul>
```

Fundamentos: expresiones

- El segundo tipo de nuevo markup es la doble llave `{{}}`
- Cuando el compilador encuentra este markup, evalúa la expresión de dentro y lo reemplaza por esta

```
<div id="display">  
  <div id="suma">  
    <h3> A + B: {{numberA + numberB}} </h3>  
  </div>  
</div>
```

Fundamentos: expresiones

- Una expresión en un template en un fragmento de código javascript que permite leer y escribir en variables
- Dichas variables no son globales, sino que pertenecen a un `$scope` determinado

Fundamentos: \$scope

- Igual que las variables dentro de una función javascript viven en un scope, Angular provee un scope para las variables, accesible en las expresiones

```
<div id="display">  
  <div id="suma">  
    <h3> A + B: {{numberA + numberB}} </h3>  
  </div>  
</div>
```

Fundamentos: model

- Y nos referimos a los valores almacenados en las variables del scope como el modelo

Fundamentos: módulos

- Un módulo en angular es un contenedor para las diferentes partes de la aplicación: controladores, servicios, filtros, directivas, etc
- Angular no tiene método main

Fundamentos: módulos

- En lugar de main, Angular define como deben inicializarse o arrancarse los diferentes componentes (Bootstrapping)

Fundamentos: módulos

- El Bootstrapping tiene varias ventajas
 - Proceso declarativo más fácil de entender
 - Módulos reusables
 - Independiente orden de carga
 - Unit tests sólo cargan los módulos necesarios
 - E2E tests pueden usar módulos para sobrescribir configuración

Fundamentos: módulos

.html

```
<body ng-app="myApp">
  <div id="prime">
    <tr ng-repeat="number in numbers">
      <td>is prime? {{ prime |
isPrime }}</td>
    </tr>
  </div>
</body>
```

.js

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

Fundamentos: módulos

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

Fundamentos: módulos

- 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 él y que contiene código de inicialización

Data Binding

- El data-binding en Angular es la sincronización automática de los datos entre los componentes del modelo y la vista
- Modelo es la única fuente-de-verdad

Data Binding

- La vista es una proyección del modelo en todo momento
- Cuando el modelo cambia, la vista refleja el cambio, y viceversa

Data Binding

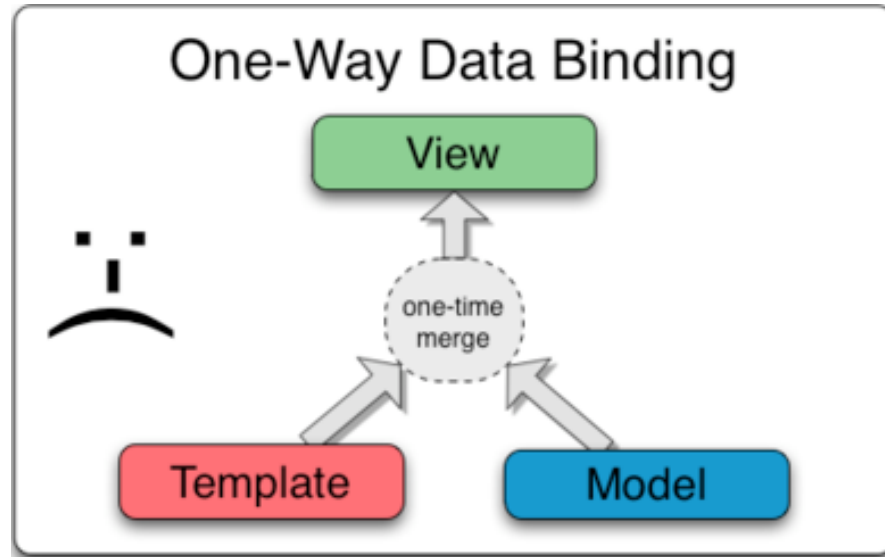
- Clásico
 - La mayoría de los sistemas de plantillas de datos se unen en una sola dirección
 - Se funde el template y el modelo en la vista
 - Los cambios en el modelo o secciones relacionadas de la vista no se reflejan automáticamente en la vista

Data Binding

- Clásico
 - Peor aún, los cambios que realice el usuario a la vista no se reflejan en el modelo
 - El desarrollador tiene que escribir el código que se sincroniza constantemente la vista con el modelo y el modelo con la vista

Data Binding

- Clásico



Data Binding

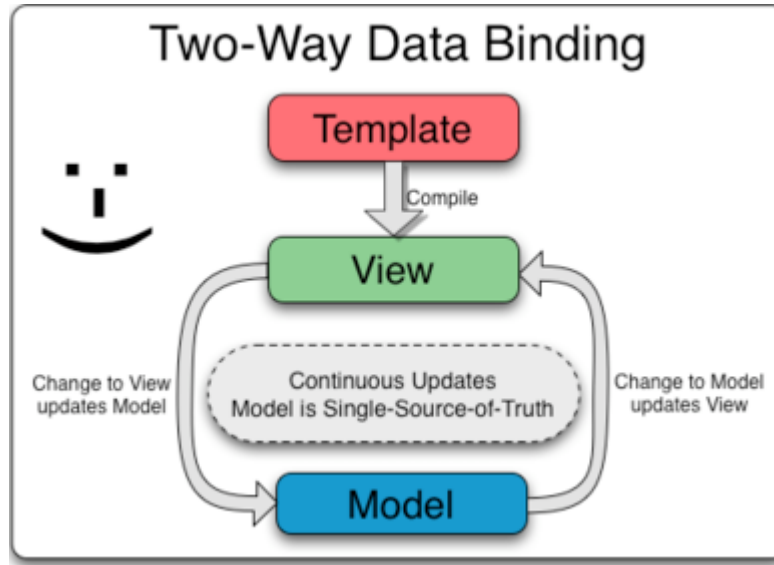
- Angular
 - El template (HTML sin compilar) se compila en el navegador y se produce una vista en vivo
 - Cualquier cambio a la vista se reflejan inmediatamente en el modelo, y cualquier cambio en el modelo se propagan a la vista

Data Binding

- Angular
 - La vista es simplemente una proyección del modelo, el controlador está completamente separado de la vista y la desconoce
 - Esto hace más fácil de testear el controlador de manera aislada sin la vista y la dependencia DOM/navegador relacionado

Data Binding

- Angular



Creando una app Angular

- Para crear una aplicación Angular necesitamos 2 cosas:
 - Tener Angular cargado en nuestro index
 - Añadir directiva ng-app

Ejercicios

- Utilizando "primera-app-mio.html", crea una app Angular
- Un módulo que depende de otros módulos por debajo de él y que contiene código de inicialización
- Añade alguna expresión que evalúe esos valores que hemos recogido gracias a ngModel