Introducción a AngularJs

Anxo Fole
afole@plainconcepts.com

Desarrollo web “moderno”

¿Por qué hemos llegado a las SPA’s?

Tendencias en el desarrollo Web

- Html renderizado en servidor + algo javascript en cliente
  - Postbacks, postbacks, postbacks…
- Ajax para realizer llamadas asíncronas al servidor
  - Mejora de la experiencia de usuario
- Librerías de cliente para facilitar la manipulación dinámica del DOM
  - jQuery
- Aparece el concepto de SPA (Single Page Application)
  - Sólo una petición al servidor para renderizar la primera página
  - Uso de frameworks en cliente que gestionan la renderización de vistas parciales, la navegación y peticiones XHR
  - El backend se convierte sólo en un API. Ya no renderiza Html

Tendencias en el desarrollo web

- Incremento de la capacidad de procesamiento de los dispositivos
  - No sólo navegadores de sobremesa
  - Navegadores móviles
- Aplicaciones móviles híbridas que “parezcan” nativas
- HTML5 y las nuevas Apis dan un soporte avanzado a estas nuevas técnicas
- Las aplicaciones vez cada vez se parecen más a las de escitorio
  - Pero ¿las pueden sustituir?

Single Page Application

- La página es descargada una vez completa del servidor
  - Por eso se llaman aplicaciones de una sola página
- La navegación es gestionada en cliente (routing)
  - Integrado con el histórico del browser
  - Podemos ir hacia atrás utilizando el navegador
- En el resto del ciclo de vida de la aplicación, el servidor:
  - Proporciona plantillas parciales de páginas bajo demanda
  - Proporciona javascript bajo demanda (dependiendo de frameworks)
  - Devuelve o procesa información mediante llamadas XHR

jQuery

- jQuery como librería javascript
- Gran éxito
  - Facilita enormemente la manipulación dinámica del DOM
  - Incluye funciones para realizar llamadas AJAX
  - Facilidad de desarrollar plugins

- Pero…
  - No tiene sistema de bindings. Toda la manipulación es explícita.
  - Muy verboso

Angular, Vue.js

- Frameworks “todo incluido” con más tirón
  - Two-way databinding
  - Templating
  - Routing
  - Inyección de dependencias
  - Separación de responsabilidades

  - Desarrollo en cliente muy parecido al desarrollo en servidor
    - SOLID

¿Actualidad?

- AngularJS
  - v1.7.8
    - Long Term Support
    - Bug Fixing

- Angular (Angular 2)
  - V8.2.8
  - Desaparecen controladores y $scope => components
  - Nuevas directivas
  - Typescript

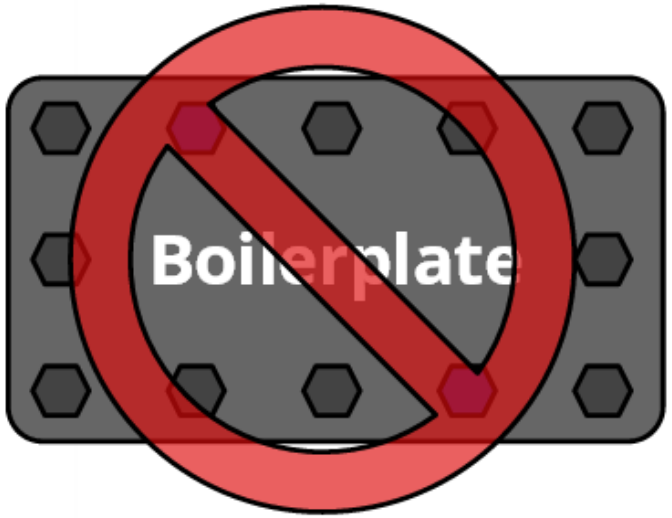
Introducción a AngularJS

AngularJs

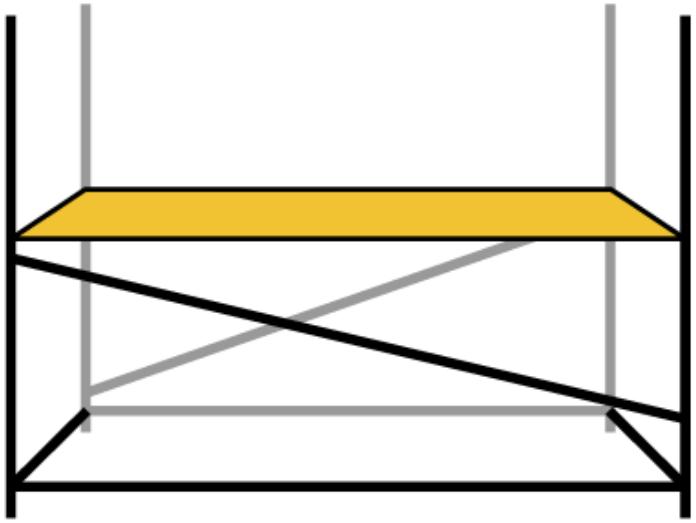
- Independiente
  - Se integra con jQuery pero no es necesario
- Modular
  - Pequeños módulos independientes que se combinan para añadir funcionalidad
- Testable
- Incluye sistema de binding
- Separación de responsabilidades

Principios de AngularJs

Opinionated Framework



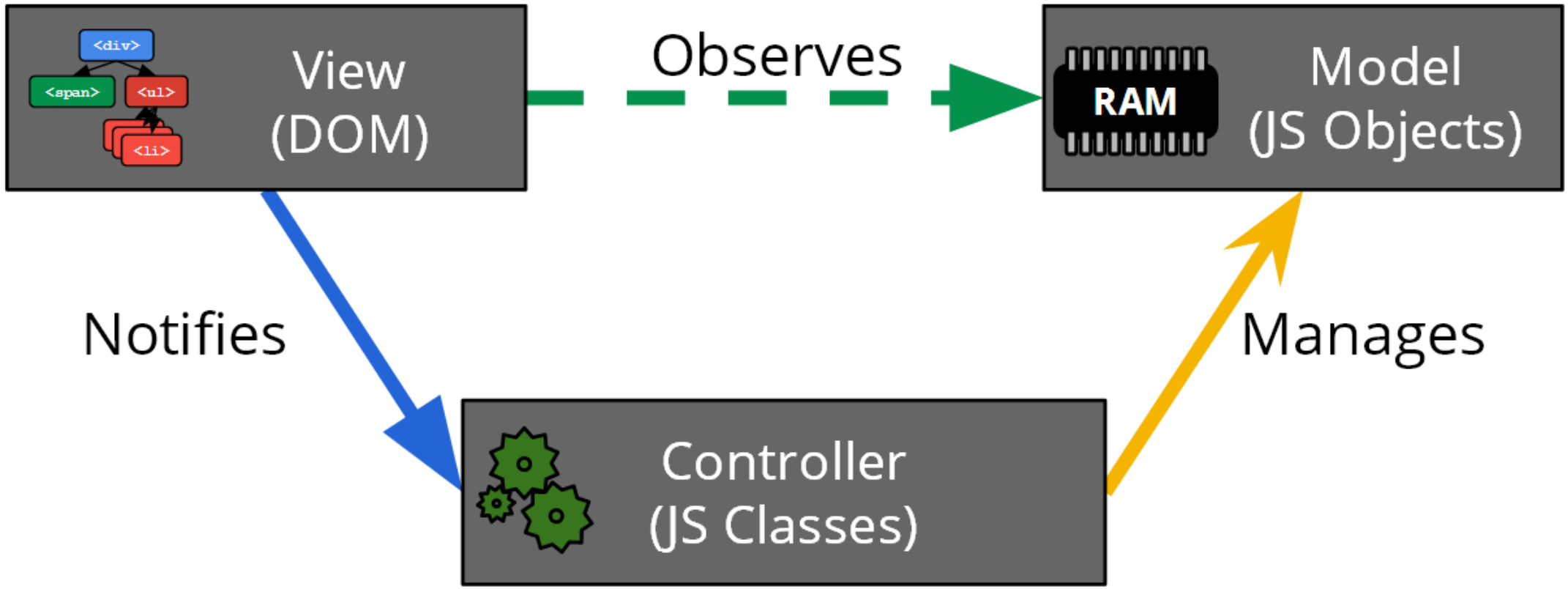
D.R.Y.



Structure

Testability

MVC



View
(DOM)
Observes
RAM
Model
(JS Objects)
Notifies
Manages
Controller
(JS Classes)

Model:

```
{  "name": "Misko"  }
```

View:

```
<p ng-controller='PersonController as person'>
    Hi, {{person.model.name}}
</p>
```

Controller:

```
myApp.controller('PersonController', function () {
    this.model = {  name: 'Misko'  };
});
```

Bootstrapping (Html)

```
<html ng-app='coolApp'>
…
<body>
…
  <script src='angular.js'></script>
  <script src='coolApp.js'></script>
</body>
</html>
```

Bootstrapping (JS)

```
var myApp = angular.module('coolApp', []);
```

$scope

You can request $scope in controllers

```
myApp.controller('GreetCtrl', function ($scope) {
    $scope.greeter = { greeting: 'Misko'};
});
```

Use in template

```
<p ng-controller='GreetCtrl'>{{greeter.greeting}}</p>
```

Data Binding

hello.js

```
this.greeting = 'Hola!';
```

hello.html

```
{{ greeter.greeting }}                    // 1-way
<input ng-model='greeter.greeting'>  // 2-way
```

Directivas

```
<div ng-repeat='item in cart.items'>
  <span ng-bind='item.name'></span>
  <button ng-click='cart.delete($index)'>
    Delete
  </button>
</div>
```

Expresiones

{{ con sintaxis de template}}

ng-bind=‘o en directivas'

- Como javascript pero se evaluan contra el **$scope**
- No se puede introducer lógica compleja (loops, throws, …)
- El operador ternario está soportado

Filtros

```
<p>{{ invoice.total | currency }}</p>
```

Inyección de dependencias

Definition

```
function MyFoo() { … };
module.service('foo', MyFoo);
```

Usage

```
function MyBar(foo) {
   Foo instanceof MyFoo; // true!
}
```

Herramientas

Angular y Visual Studio

- Angular está disponible como paquete de NuGet
  - Pero no todos lo módulos de la comunidad lo están
  - npm incluye más módulos pero no está integrado con VS

- Visual Studio Code
  - Windows, Linux, macOS
  - Completado de código
  - Plugins

Depuración

- Puntos de interrupción en Visual Studio en javascript sólo funcionan con Internet Explorer

- La mejor opción: Herramientas de desarrollador de navegadores
  - **Chome Dev Tools** es la mejor. Tiene plugins específicos para Angular
    - Batarang
    - ng-inspector

http://angular.github.io/angular-phonecat/step-11/app/#/phones

Chorme Developer Tools

- **Elements:** Inspección del DOM
- **Network:** Inspección de las peticiones
- **Sources:** Inspección y depuración de javascript
  - F8 + F10 + F11
  - Puntos de interrupción
  - Evaluación de variables
- **Timeline y Profiles:** Profiling
- **Resources:** Cookies, local storage, …
- **Audit:** Recomendaciones
- **Console:** La consola de toda la vida

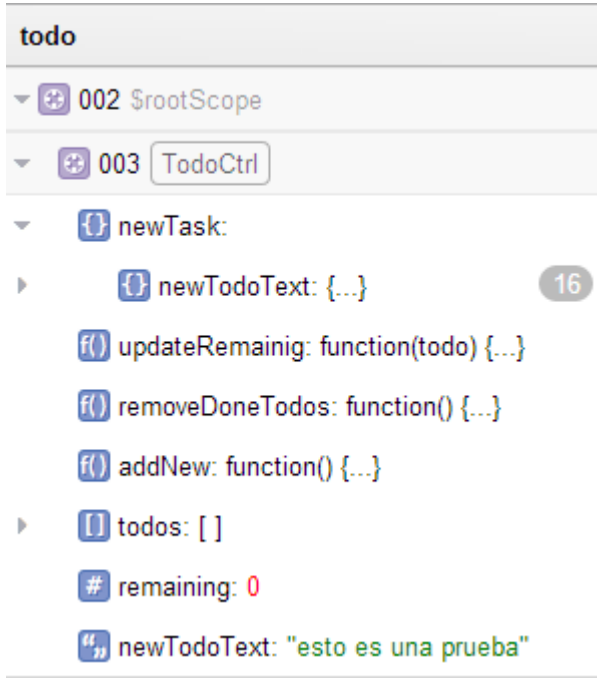
Desactivar cache en desarrollo!

Batarang

- Muestra los scopes en una página
- Mide el rendimiento
- Gráfica de dependencias
- Options: Indica la versión de Angular utilizada por el sitio

ng-inspector

- Para Crome y Safari
- Muestra los scopes, así como los controladores o directivas que los han creado
- Muestra la información de los formularios
- Al pinchar en un elemento muestra su información en la consola



http://ng-inspector.org/

Otras herramientas

- Plunker: http://plnkr.co/
  - Ejemplos
  - Los ejemplos de la web de Angular están alojados aquí
- JsFiddle: http://jsfiddle.net/
- Fiddler: Http debugger. Si los tabs de Network del browser no son suficientes…
- Postman: Permite hacer llamadas a un api Http con diferentes opciones de autenticación

AngularJs y navegadores antiguos

Soporte de navegadores

- Todas las versiones soportan:
  - Safari
  - Chrome
  - Firefox
  - Opera
  - IE*
- Soporte de Internet Explorer
  - Versión 1.2.x: soporta desde IE8. Sólo corrige bugs específicos de IE desde IE9
  - Versión 1.3.x: Elimina soporte a IE8

Soporte versiones antiguas de IE

1. Añadir polyfill para JSON.stringify (IE7 y anteriores)
2. Añadir **id="ng-app"** al elemento raíz de la aplicación junto con **ng-app** para el bootstrapping
3. No usar directivas como elementos **<ng-view>**. Usar en su lugar directivas como atributos **<div ng-view>**
4. Si se usan directivas como elementos añadir el elemento personalizado al DOM (IE8 y anteriores)

5. Usar **ng-style** en vez de **style={{cssPersonalizado}}**

https://docs.angularjs.org/guide/ie

Organización del código

Estructurar el código fuente

- Un archivo por módulo (app.js)
  - En los ejemplos de angular seed aparecía así
- Un archivo por tipo (controllers.js, services.js)
  - Aplicaciones pequeñas
  - Usar IFFE’s para el módulo principal y los componentes
- Por características
  - Agrupar áreas de la aplicación por característica
  - Agrupar todos los archivos relacionados

Estructura del código fuente

- Estructura recomendada para aplicaciones medianas/grandes